

Faculté des Sciences de Luminy  
ED 184 – Mathématiques et Informatique  
UFR  
LIF/TALEP

Thèse présentée pour obtenir le grade universitaire de docteur

Discipline : Informatique

Jérémie TAFFOREAU

Modèle joint pour le Traitement Automatique de la Langue :  
Perspectives aux travers des réseaux de neurones

Soutenue le 20/11/2017 devant le jury :

Philippe LANGLAIS	Université de Montréal	Rapporteur
Alexandre ALLAUZEN	Université Paris Sud	Rapporteur
Christophe CERISARA	CNRS LORIA, Nancy	Examineur
Yannick ESTEVE	Université du Maine	Examineur
Benoit FAVRE	Aix-Marseille Université	Examineur
Frédéric BECHET	Aix-Marseille Université	Directeur de thèse
Thierry ARTIERES	Aix-Marseille Université	Co-Directeur de thèse



Cette oeuvre est mise à disposition selon les termes de la [Licence Creative Commons Attribution - Pas d'Utilisation Commerciale - Pas de Modification 3.0 France](#).

# Résumé

Les recherches en Traitement Automatique des Langues (TAL) ont identifié différents niveaux d'analyse lexicale, syntaxique et sémantique. Il en découle un découpage hiérarchique des différentes tâches à réaliser afin d'analyser un énoncé. Les systèmes classiques du TAL reposent sur des analyseurs indépendants disposés en cascade au sein de chaînes de traitement (pipelines). Cette approche présente un certain nombre de limitations : la dépendance des modèles à la sélection empirique des traits, le cumul des erreurs dans le pipeline et la sensibilité au changement de domaine. Ces limitations peuvent conduire à des pertes de performances particulièrement importantes lorsqu'il existe un décalage entre les conditions d'apprentissage des modèles et celles d'utilisation. Un tel décalage existe lors de l'analyse de transcriptions automatiques de parole spontanée comme par exemple les conversations téléphoniques enregistrées dans des centres d'appels. En effet l'analyse d'une langue non-canonique pour laquelle il existe peu de données d'apprentissage, la présence de disfluences et de constructions syntaxiques spécifiques à l'oral ainsi que la présence d'erreurs de reconnaissance dans les transcriptions automatiques mènent à une détérioration importante des performances des systèmes d'analyse. C'est dans ce cadre que se déroule cette thèse, en visant à mettre au point des systèmes d'analyse à la fois robustes et flexibles permettant de dépasser les limitations des systèmes actuels.

Ainsi, notre travail propose de s'abstraire des limitations inhérentes aux modèles *pipelines* à l'aide de modèles issus de l'apprentissage par réseaux de neurones profonds. Nous proposons d'utiliser un modèle de type *End-to-End Bidirectional Recurrent Neural Network* (BiRNN) applicable à différentes analyses du TAL. Il s'agit d'un modèle générique qui résout indépendamment chaque tâche comme un problème d'étiquetage de séquences, en ne considérant que les mots en entrée. Plusieurs instances de cette architecture sont entraînées conjointement à l'aide d'un algorithme d'apprentissage multitâche ayant pour but d'éviter des phénomènes de cumul d'erreurs en permettant la remise en cause des prédictions relatives à chaque niveau d'analyse. Concernant les capacités d'adaptation des modèles à de nouveaux domaines ou de nouveaux registres de langue, nous proposons des stratégies d'adaptation lexicales basées sur des représentations continues des mots appelés « plongements » ou *word embeddings*. La stratégie proposée vise à fournir des représentations pertinentes pour les mots rares ou inconnus, dont les proportions sont fortement modifiées lors d'un changement de domaine.

Afin d'éprouver et de valider nos hypothèses, nous avons évalué notre modèle sur différents corpus de référence utilisés en TAL. Les performances « état-de-

l'art » obtenues valident notre approche, ce qui nous a permis de mener une évaluation comparative entre l'apprentissage multitâche d'un BiRNN et un *pipeline* de BiRNNs. On observe ainsi des gains de performances en limitant le cumul des erreurs inhérent au *pipeline*. De plus, la possibilité d'utiliser des sorties additionnelles permet de rendre le système plus robuste aux erreurs d'analyse préliminaire. Dans le cadre du projet Européen SENSEI dans lequel s'inscrit cette thèse, nous avons appliqué notre approche à l'analyse sémantique de transcriptions automatiques de conversations téléphoniques. Nous avons pu montrer que notre approche multitâche présente une meilleure robustesse aux erreurs de transcription que l'approche *pipeline* classique. Enfin, l'étude systématique de l'impact de la quantité de données disponible en entraînement sur les performances, réalisée tout au long de ces évaluations, a permis de déterminer que notre approche est particulièrement performante lorsque peu d'exemples d'apprentissage sont disponibles.

Mots clés : Traitement de la Langue, Analyse syntaxique, Analyse sémantique, Apprentissage automatique, Réseaux de neurones profonds, Multitâche.

# Abstract

NLP researchers has identified different levels of linguistic analysis. This lead to a hierarchical division of the various tasks performed in order to analyze a text statement. The traditional approach considers task-specific models which are subsequently arranged in cascade within processing chains (pipelines). This approach has a number of limitations : the empirical selection of models features, the errors accumulation in the pipeline and the lack of robustness to domain changes. These limitations lead to particularly high performance losses in the case of non-canonical language with limited data available such as transcriptions of conversations over phone. Disfluencies and speech-specific syntactic schemes, as well as transcription errors in automatic speech recognition systems, lead to a significant drop of performances. It is therefore necessary to develop robust and flexible systems. We intend to perform a syntactic and semantic analysis using a deep neural network multitask model while taking into account the variations of domain and/or language registers within the data.

Our work proposes to abstract from the pipeline limitations using Deep Learning models. We consider a Bidirectional End-to-End Recurrent Neural Network applied to various NLP tasks. This is a generic model that independently resolves each task as a sequence labeling problem, only considering input words. Several BiRNNs are trained jointly using a multitask learning algorithm in order to allow soft accesses to each level predictions. Moreover, this algorithm can consider input features as additional outputs. During the training phase, this allows the use of features that will be unavailable or strongly noisy during the evaluation phase on unknown data. Finally, as part of the application to the spoken language understanding, we propose lexical adaptation strategies based on word embeddings. This methodology aims to provide more relevant representations in the case of rare or unknown out-of-vocabulary words, whose proportions explode when a domain change occurs.

We realized a BiRNN architecture obtaining state-of-the-art performances on different benchmarks of the syntactic analysis. This allowed us to compare a multitask BiRNN and a BiRNNs pipeline. Performance gains can be observed by reducing the accumulation of pipeline errors. In addition, the ability to use additional outputs makes the system more robust to automatic features errors. We then applied our approach to the semantic analysis on automatically annotated conversations over phone. In the case of automatic transcripts with a high word error rate, automatically generated features are noisy. Their use as inputs of traditional models usually results in a significant drop of performances. Our End-to-End model, partially independent from the preliminary analyzes, presents a

better robustness to transcription errors. Finally, the systematic study of the impact of the amount of data available during training show that our approach is mostly effective when few training examples are available. Truncating the training corpus also makes it possible to artificially change the proportions of unknown words in order to study the impact of our OOV handling strategies.

Keywords : Natural Language Processing, Syntactic & Semantic Parsing, Machine Learning, Deep Neural Networks, Multitask.

# Remerciements

Avant toute chose, le moment est venu de remercier les personnes qui ont contribué à cette thèse. Mes remerciements s'adressent à Frédéric Béchet, sans qui cette aventure aurait, à coup sûr, viré à l'Odyssée. Ses attentions bienveillantes, bien souvent teintées d'un paternalisme rassurant, m'ont toujours été d'un précieux réconfort. Thierry Artières, qui a accepté de se joindre à cette laborieuse entreprise, m'a permis, par son sens de la critique et sa rigueur scientifique, de donner le meilleur de moi-même. Enfin, Benoit Favre, l'homme de l'ombre, risque d'avoir bien du mal à y rester, tant il attire naturellement la lumière. Sa saine curiosité, sa justesse d'analyse, sa prudence et sa diplomatie en font un allié inestimable pour quiconque souhaiterait plonger dans les abysses de la Recherche. A tous les trois, votre disponibilité, votre patience, la confiance et l'autonomie que vous m'avez accordées, votre enthousiasme sur les *expé*, vos conseils avisés et empreints du recul qui si souvent me manquait, tous ces moments passés à tenter de faire parler les équations, à discuter vos idées, à interpréter les résultats, à retoucher nos articles, m'ont permis de me dépasser dans la réalisation de ce travail, et même d'y prendre plaisir. Merci d'avoir su tenir le cap, surtout lorsqu'il fallait naviguer à l'aveugle.

J'adresse également mes remerciements à Messieurs Philippe Langlais, Alexandre Allauzen, Christophe Cerisara et Yannick Estève qui ont accepté de lire et d'évaluer ce travail de thèse. Vos retours et vos questions résonnent aujourd'hui encore dans mon esprit.

Je tiens finalement à remercier Fanny pour son soutien indéfectible et la force de caractère qu'elle a toujours su m'insuffler. Olivier Michalon, Jérémy Trione et bien sûr Martin, frères d'armes avec qui j'ai partagé ces quatre années de galère. Là où une seule corde aurait cédé, quatre cordes, tressées ensemble, ont résisté. Gauthier Gaide, probablement l'une des rares personnes à comprendre cette thèse sans même avoir besoin de la lire *a posteriori*. Merci pour toutes ces nuits passées à décortiquer et formaliser nos intuitions. Jeremy Auguste, qui a répliqué l'ensemble des expériences décrites dans cet ouvrage, et activement participé à l'interprétation des résultats. Jeanne, S'mi, Joris, Gwen, Théo, Lila, Alain, Charlie et toute la *clique*, pour leur accueil et leur participation à cette formidable utopie luminyenne, avec une pensée toute particulière pour Jynx, Moon, Olaf, Bakara et toute sa dynastie féline, sans lesquels ces nuits de rédaction auraient parues bien mornes.

Enfin, je souhaiterais dédier cette thèse à ma grand-mère, Eliane Tafforeau, qui en plus d'avoir relu et corrigé intégralement le manuscrit que vous tenez aujourd'hui, m'a très tôt appris que "*la vieille ne voulait pas mourir parce qu'elle apprenait chaque jour quelque chose de nouveau*".

# Table des matières

<b>Résumé</b>	<b>4</b>
<b>Abstract</b>	<b>6</b>
<b>Remerciements</b>	<b>7</b>
<b>Liste des figures</b>	<b>10</b>
<b>Liste des tableaux</b>	<b>11</b>
<b>Introduction</b>	<b>13</b>
<b>Notations</b>	<b>21</b>
<b>1 Formalisme des réseaux de neurones profonds</b>	<b>22</b>
1.1 Contexte et Motivations	22
1.2 Neurones artificiels	23
1.2.1 Définition	24
1.2.2 Perceptron pour la classification binaire	25
1.2.3 Perceptron multi-classes	27
1.3 Réseaux <i>feed-forward</i>	28
1.3.1 Perceptron multi-couches	29
1.3.2 Algorithme d'apprentissage	32
1.4 Réseaux récurrents	37
1.4.1 Structure récurrente	38
1.4.2 Extensions	39
1.4.3 Algorithme d'apprentissage	42
1.4.4 Mise en forme des exemples	46
<b>2 Application au Traitement Automatique de la Langue</b>	<b>49</b>
2.1 Contexte et Motivations	49
2.1.1 Analyse sémantique	50
2.1.2 État de l'art	51
2.2 Analyse syntaxique du texte	52
2.2.1 Étiquetage morpho-syntaxique	52
2.2.2 Analyse syntaxique de surface	54
2.2.3 Analyse en dépendances syntaxiques	55
2.3 Formalisation des tâches d'analyse	57
2.3.1 Étiquetage	57
2.3.2 Segmentation	58

2.3.3	Détection de relation	58
2.4	Représentation vectorielle de mot	59
2.4.1	Motivations	59
2.4.2	One-hot Vectors	61
2.4.3	Word Embeddings	61
2.4.4	Application aux réseaux de neurones	66
<b>3</b>	<b>Algorithme d'apprentissage multitâche pour l'analyse syntaxique</b>	<b>72</b>
3.1	État de l'Art	73
3.1.1	A Unified Architecture for NLP : DNN with MTL	74
3.1.2	Multi-Task Learning for Multiple Language Translation	76
3.1.3	Multi-task Sequence to Sequence Learning	78
3.1.4	Discussion	79
3.2	Modèle Joint	80
3.2.1	LSTM-BiRNN	81
3.2.2	Critère d'apprentissage global	86
3.2.3	Apprentissage par tronçon	87
3.3	Applications	89
3.3.1	Étiquetage morpho-syntaxique	90
3.3.2	Analyse syntaxique de surface	92
3.3.3	Analyse en dépendance syntaxique	95
<b>4</b>	<b>Analyse et Compréhension de la langue orale</b>	<b>101</b>
4.1	État de l'Art	101
4.2	Corpus de transcriptions	103
4.2.1	Annotations syntaxiques	104
4.2.2	Annotations sémantiques	104
4.3	Expérimentations	106
4.3.1	Conditions expérimentales	107
4.3.2	Évaluation sur les transcriptions manuelles	108
4.3.3	Évaluation sur les transcriptions automatiques	110
4.4	Problème des mots inconnus	114
4.4.1	Caractérisation	115
4.4.2	Généralisation	116
4.4.3	Stratégies envisagées	118
4.4.4	Application	122
	<b>Conclusion</b>	<b>128</b>
	<b>Bibliographie</b>	<b>132</b>
	<b>Index</b>	<b>139</b>
	<b>ANNEXES</b>	<b>141</b>

# Liste des figures

1.1	<i>Neurone formel</i>	24
1.2	<i>Exemple de séparateur linéaire</i>	25
1.3	<i>Exemple fictif d'exécution du Perceptron permettant d'observer l'évolution de l'hyperplan séparateur calculé à mesure que les exemples d'apprentissage sont pris en compte. - en.wikipedia.org</i>	26
1.4	<i>Recurrent Neural Network (RNN) - Guo 2013</i>	38
1.5	<i>Long Short-Term Memory cell (LSTM)</i>	39
1.6	<i>Bidirectional Recurrent Neural Network (BiRNN)</i>	41
1.7	<i>Modèle d'attention</i>	43
1.8	<i>Unfolded Recurrent Neural Network (RNN) - Guo 2013</i>	44
2.1	<i>Représentation de features dans le cas du Traitement de la Langue (a) sous forme de one-hot vectors, (b) sous forme de word embeddings. - GOLDBERG 2015</i>	60
2.2	<i>Word2Vec - MIKOLOV, I. SUTSKEVER, K. CHEN et al. 2013</i>	63
2.3	<i>GloVe - E. H. HUANG, SOCHER, C. D. MANNING et al. 2012</i>	64
2.4	<i>W2Vf - LEVY et GOLDBERG 2014</i>	65
2.5	<i>Skip-Thought Vectors - KIROS, ZHU, SALAKHUTDINOV et al. 2015</i>	65
2.6	<i>Architecture MLP prenant des embeddings en entrée</i>	67
2.7	<i>Architecture MLP permettant un raffinement des embeddings</i>	69
3.1	<i>Approche MTL envisagée par COLLOBERT et WESTON 2008 dans le cadre de l'analyse syntaxique - (a) Une architecture générique dédiée au TAL qui prédit la classe d'un mot à partir d'une fenêtre injectée dans un Time-Delay Neural Network. Les représentations vectorielles des mots et de leur features sont accessibles via des Lookup tables. - (b) Un exemple de système multitâche utilisant cette architecture pour apprendre conjointement deux tâches. Une Lookup table est partagée entre les TDNN, tous les autres paramètres étant spécifiques. Ces réseaux sont entraînés à l'aide d'un algorithme qui alterne entre les tâches.</i>	75
3.2	<i>Approche MTL envisagée par DONG, H. WU, HE et al. 2015 dans le cadre de la traduction automatique. Ils réalisent la traduction simultanée de l'anglais vers le Français, l'Espagnol l'Allemand et le Portugais. Leur modèle se compose d'un Encoder BiRNN et de quatre Decoders RNN. La communication entre Encoder et Decoder passe par un modèle d'attention spécifique au Decoder. Les quatre Sequence-to-Sequence ainsi définis sont entraînés à l'aide d'un algorithme considérant une alternance de mini-batches monolingues.</i>	77

3.3	<i>Approche MTL envisagée par LUONG, LE, Ilya SUTSKEVER et al. 2015 dans le cadre de la traduction automatique, du parsing et de l'image captioning. Ils proposent trois types de structure Sequence-to-Sequence où les encoders et les decoders sont partagés : "one-to-many", "many-to-one" et finalement, la plus générale, "many-to-many". Les couples (encoder/decoder) ainsi définis sont entraînés deux à deux à l'aide d'un algorithme considérant une alternance de mini-batches propres à une tâche. Chaque tâche est associée à un paramètre <math>\alpha</math> qui caractérise la proportion d'exemples qu'on souhaite considérer lors de l'apprentissage de cette tâche.</i>	78
3.4	<i>Scoped Back Propagation Throught Time avec <math>\tau = 10</math> et <math>s = 3</math>.</i>	88
4.1	<i>Évaluation (F-score) sur la tâche de détection de cadre sémantique en fonction du volume de l'ensemble d'apprentissage pour trois configurations : apprentissage monotâche (E2E:BiRNN), apprentissage multitâche uniforme (MT:BiRNN 1/1) et avec un poids <math>\beta</math> dédié aux frames plus important (MT:BiRNN 1/8).</i>	110
4.2	<i>Évaluation (F-score) sur la tâche de détection de cadre sémantique en fonction du taux erreurs mot WER des transcriptions automatiques, de trois modèles : les deux pipelines PL:MACAON et PL:CRF, et notre réseau multitâche (MT:BiRNN).</i>	113
4.3	<i>Classification des mots inconnus : l'ensemble de test <math>V_{test}</math> est composé de mots connus (région <math>w_1</math>), de trois types de mots inconnus (régions <math>w_2</math>, <math>w_3</math> et <math>w_4</math>)</i>	116
4.4	<i>Évaluations (F1-score et PWA restreintes aux <b>OoE</b> et aux <b>OoT</b>) en fonction du volume de données disponibles durant l'apprentissage.</i>	123

## Liste des tableaux

2.1	<i>État de l'art - étiquetage morpho-syntaxique.</i>	53
2.2	<i>État de l'art - analyse syntaxique de surface.</i>	54
2.3	<i>État de l'art - analyse en dépendance syntaxique.</i>	56
3.1	<i>Évaluation sur l'étiquetage morpho-syntaxique sur le Penn TreeBank.</i>	91
3.2	<i>Évaluation sur l'étiquetage morpho-syntaxique en fonction du volume de l'ensemble d'apprentissage sur le Penn TreeBank.</i>	91
3.3	<i>Évaluation sur l'analyse syntaxique de surface sur le CoNLL 2000.</i>	93
3.4	<i>Étude des features lexicales sur l'analyse syntaxique de surface en fonction du volume de l'ensemble d'apprentissage sur le CoNLL 2000.</i>	93
3.5	<i>Évaluation sur l'analyse syntaxique de surface en fonction du volume de l'ensemble d'apprentissage sur le CoNLL 2000.</i>	94

3.6	<i>Évaluation sur l'analyse en dépendance syntaxique en fonction du volume de l'ensemble d'apprentissage sur le CoNLL 2009.</i>	96
3.7	<i>Évaluation sur l'analyse en dépendance syntaxique sur le CoNLL 2009.</i>	97
4.1	<i>Exemple d'annotation pour la phrase : I I lost my phone in bus 38. Les étiquettes de cadre sémantique suivent une syntaxe simple : <b>position</b> (soit B pour begin, soit I pour inside); <b>frame name</b>; <b>role</b> (agent, object ou LU pour lexical unit).</i>	106
4.2	<i>Description des corpus d'apprentissage, de développement et de test du corpus RATP-DECODA</i>	107
4.3	<i>Évaluation sur les transcriptions manuelles du corpus RATP-DECODA : comparaison entre l'approche BiRNN End-to-End, Pipeline et Multi-tâche.</i>	108
4.4	<i>Évaluation sur l'étiquetage morpho-syntaxique, l'analyse en dépendance syntaxique, la détection de disfluence, la reconnaissance d'entité nommée et la détection de frame sémantique en fonction du volume de l'ensemble d'apprentissage sur les transcriptions manuelles du corpus RATP-DECODA.</i>	109
4.5	<i>Évaluation sur la détection de cadre sémantique sur les transcriptions manuelles et automatiques du corpus RATP-DECODA.</i>	111
4.6	<i>Évaluation sur la détection de cadre sémantique sur les transcriptions automatiques du corpus RATP-DECODA, en fonction de la distribution <math>\beta</math> du critère d'apprentissage multitâche. Afin de pallier les erreurs d'alignement des transcriptions automatiques, pour chaque segment détecté, on vérifie si un segment avec une étiquette de frame identique apparaît dans le même intervalle de temps <math>\pm\delta</math></i>	112
4.7	<i>Distribution des OOV dans <math>C_{test}</math> en fonction de différentes partitions de <math>C_{train}</math>, le corpus <math>C_{emb}</math> étant fixé.</i>	117

# Introduction

L'apparition des premières écritures, il y a près de 5500 ans, a révolutionné à tel point l'humanité, que celle-ci décida ultérieurement que son Histoire débiterait à cet instant. L'importance de l'écriture en tant que support de la Langue et moyen de communication n'a fait que croître depuis. L'avènement de l'ère du numérique a marqué un tournant dans l'évolution des moyens de communication. L'apparition et la démocratisation de nouveaux vecteurs d'interaction sociale ont engendré une explosion de la quantité d'information échangée. De nos jours, quelques 4 milliards d'internautes communiquent à un rythme effréné avec près de 500 millions de tweets échangés chaque jour, 400 heures de vidéos Youtube mises en ligne chaque minute et 4100 statuts Facebook partagés chaque seconde. Un des principaux enjeux de notre siècle est de parvenir à tirer profit de cette manne intarissable de ressources.

En 1950, Alan Turing publie un article intitulé *Computing Machinery and Intelligence* qui pose les bases de ce qu'on appellera par la suite le *test de Turing*. L'université de Georgetown expérimente alors en 1954 un système de traduction automatique du russe vers l'anglais. À l'époque, les auteurs estimeront que sous 3 à 5 ans, le problème de la traduction automatique devrait être résolu. Le Traitement Automatique de la Langue (TAL) est ainsi né du besoin d'appliquer des méthodes et techniques informatiques à la réalisation automatique de l'ensemble des tâches faisant intervenir le langage humain. Un énoncé est généralement ambiguë et sa structure linguistique peut dépendre de nombreuses variables, incluant le registre de langue, des expressions issues de dialectes ou propres à un contexte social. Le saint Graal du TAL pourrait être défini comme une interface conversationnelle gérant les interactions *homme-machine* en langage naturel.

Ainsi, ce domaine vise à produire des systèmes de reconnaissance, d'analyse et de synthèse de texte appliqués par exemple à la correction, à la traduction, au résumé automatique ou encore à l'analyse d'opinion. Ces outils sont par la suite appliqués dans de nombreux domaines et à diverses fins. La traduction automatique permet de faciliter la communication entre individus de cultures différentes sans passer par l'apprentissage d'une langue commune. L'ensemble des outils d'aide à la saisie tels que la correction et la complétion automatiques, permettent à la fois d'accélérer la rédaction des messages échangés tout en s'assurant de leur compréhension, que ce soit par confort ou par nécessité (dans le cas d'utilisateurs handicapés). Le résumé automatique couplé à l'analyse d'opinion permet de synthétiser les avis et réflexions qui émergent d'une masse de commentaires à propos d'un fait d'actualité. Autant de domaines d'application

importants pour notre société en offrant à chaque individu une expérience plus confortable de la communication.

Le projet Européen SENSEI<sup>a</sup> dans le cadre duquel cette thèse a été réalisée s'inscrit dans cette tradition du TAL visant à développer des outils permettant d'analyser le langage humain afin de faciliter la découverte et l'accès à l'information stockée dans de grandes collections de données langagières. Ce projet, sous-titré "*Making Sense of Human Interaction*", consiste à mettre en place des outils d'aide à l'analyse de très vastes corpus de conversations collectées, soit dans des centres d'appels téléphoniques (conversations orales) soit sur des réseaux sociaux (conversations textuelles). Ce projet européen s'est fixé pour objectif de développer de nouvelles méthodes d'accès au *sens* à travers les aspects suivants :

- Analyser des conversations, à la fois sur leur contenu sémantique, mais aussi sur les dimensions dialogiques et comportementales des participants ;
- Développer des méthodes permettant d'adapter les modèles d'analyse rapidement à la diversité des contenus et des médias véhiculant de nouveaux types de conversations ;
- Générer des rapports de résumé permettant de présenter à un utilisateur, sous une forme synthétique, une collection de conversations entre deux ou plusieurs participants ;
- Évaluer de façon "écologique" les technologies développées en concertation avec les utilisateurs finaux dans les différents cadres d'études.

Dans le cadre cette thèse nous nous focaliserons uniquement sur les tâches d'analyse syntaxique et sémantique des transcriptions de conversations, qu'elles soient réalisées manuellement ou générées automatiquement par un système de reconnaissance automatique de la parole.

Les recherches en TAL ont identifié différents niveaux d'analyse lexicale, syntaxique et sémantique. Il en découle un découpage hiérarchique des différentes tâches à réaliser afin d'analyser un énoncé. Si on considère un texte comme étant une succession de caractères, il est premièrement nécessaire de découper ce flux en mots puis en phrases. Cette analyse lexicale est suivie d'une analyse syntaxique qui se décompose en un ensemble de traitements dont on peut citer les plus fréquemment étudiées. Tout d'abord, l'*étiquetage morpho-syntaxique* détermine la classe grammaticale des mots d'une phrase considérée. On peut ensuite identifier pour chacun des mots leur "lemme", c'est à dire leur forme canonique sans flexion (accord en genre, nombre, temps, etc.). Puis, l'*analyse en syntagme* identifie les éléments constituant cette phrase tels que les groupes nominaux ou

---

a. <http://www.sensei-conversation.eu/>

verbaux à partir des mots annotés grammaticalement. Enfin, l'*analyse syntaxiques* détecte les relations syntaxiques existant entre les syntagmes identifiés précédemment. L'analyse sémantique consiste à interpréter et représenter sous forme abstraite les concepts sémantiques auxquels une phrase fait référence. Cependant, même si la représentation sémantique s'abstrait de la structure syntaxique de la phrase, l'analyse sémantique prédisant cette structure est souvent basée sur des informations syntaxiques issues des analyses préalables. C'est la raison pour laquelle la plupart des modèles d'analyse sémantique actuels prennent tous en considération la structure syntaxique de l'énoncé car cette dernière permet d'explicitier les relations de dépendance entre les différents constituants de la phrase. De cette formalisation sous forme d'une hiérarchie de sous-problèmes découle un découpage modulaire des traitements d'analyse.

Depuis les années 90, les systèmes de Traitement Automatique de la Langue reposent principalement sur des modèles d'apprentissage automatique (*Machine Learning - ML*). De nombreux algorithmes d'apprentissage automatique ont été appliqués afin de réaliser des analyseurs linguistiques. Ces algorithmes (CRF, SVM, DNN) prennent en entrée des *traits* (ou *features*) qui sont des attributs décrivant les données d'entrée. Les modèles statistiques résultant prennent des décisions probabilistes basées sur des valeurs associées à chaque *features*. De tels modèles ont l'avantage de pouvoir exprimer leurs incertitudes vis-à-vis d'un certain nombre de prédictions possibles. Considérer ainsi simultanément plusieurs hypothèses, produit généralement de meilleurs résultats lorsque ces modèles font partie intégrante de systèmes plus complexes. Le *Machine Learning* appliqué au TAL considère que la connaissance linguistique reste implicitement représentée dans le corpus accumulé. Les performances des modèles qui en découlent sont donc fortement impactées par la quantité de données disponibles en entraînement. Ces procédures d'apprentissage tirent profit d'algorithmes d'inférence statistique afin de produire des modèles robustes aux entrées erronées ou contenant des mots ou des structures inconnus (non-observés en entraînement). Les algorithmes d'apprentissage automatique observent les caractéristiques d'un grand nombre d'exemples d'apprentissage supposés indépendants. Parmi les redondances dans la distribution des *features* associés aux exemples, ils détectent des motifs suffisamment discriminants pour guider des prédictions automatiques. Ils sont donc basés sur l'hypothèse que le corpus d'apprentissage est représentatif de l'ensemble des exemples qu'on serait amené à analyser. En d'autres termes, ils sont sensibles aux variations de distribution statistique entre ensembles d'entraînement et d'évaluation.

Depuis 2010, avec la démocratisation du calcul parallèle sur GPU, le *Machine Learning* et par conséquent le TAL ont été bouleversés par l'arrivée des architectures issues de l'apprentissage par réseaux de neurones profonds ou *Deep Learning* (G. E. HINTON, OSINDERO et TEH 2006). Ce domaine s'intéresse à l'en-

traînement de modèles composés de nombreuses couches de neurones formant une fonction paramétrique *i.e.* dont les sorties dépendent à la fois des entrées et de paramètres internes. Ils sont utilisés afin de résoudre des problèmes de classification. Tandis que des exemples d'apprentissage sont présentés au réseau, un algorithme d'optimisation mesure son erreur de classification. Le gradient de l'erreur vis-à-vis des paramètres internes permet d'indiquer au modèle comment modifier ces derniers afin de minimiser l'erreur qu'il commet. Les réseaux de neurones profonds ont prouvé leur efficacité sur des tâches complexes notamment dans les domaines du traitement automatique de la langue (COLLOBERT, WESTON, BOTTOU et al. 2011), de la classification d'images (CIRESAN, MEIER, MASCI et al. 2011), et du traitement acoustique (HINTON, DENG, D. YU et al. 2012). Grâce à l'augmentation de la puissance de calcul et de la quantité de données disponibles, il est désormais possible d'entraîner efficacement des architectures neuronales de plus en plus complexes (SZEGEDY, LIU, JIA et al. 2015). De nombreuses études misent sur la flexibilité et l'adaptabilité de tels systèmes dans le cas du traitement de données textuelles. Cet engouement s'est fortement accéléré ces dernières années, de nouvelles approches sont constamment publiées, et toutes n'ont pas encore prouvé leur supériorité. C'est dans ce contexte qu'on a été réalisé ces travaux qui tentent d'adapter et d'appliquer ces avancées dans le cadre de l'analyse syntaxique et sémantique du texte.

**Problématiques** Les systèmes classiques du TAL reposent sur des analyseurs indépendants disposés en cascade. En effet, on souhaite que l'analyse produite par un modèle dépende des analyses issues des niveaux inférieurs. C'est pourquoi l'approche traditionnelle consiste à développer des modèles spécifiques à chacune de ces analyses qui sont par la suite agencés successivement au sein de chaînes de traitement (*pipelines*). Mais cette approche présente un certain nombre de limitations :

- *La sélection empirique de features* : Les modèles traditionnellement utilisés sont des modèles linéaires s'appuyant sur une description des meilleurs éléments à prendre en compte dans la réalisation d'une tâche, les *features*. Afin que ces modèles soient performants, il est nécessaire de croiser certaines caractéristiques simples afin de former des *features* complexes. Cette étape de sélection des *features* maximisant les performances des algorithmes est guidée par l'expertise linguistique et l'ajustement empirique sur corpus, ce qui en complexifie le développement et risque de produire des résultats sous-optimaux.
- *Le cumul des erreurs* : Il s'agit d'un problème inhérent à la résolution séquentielle d'un ensemble de tâches inter-dépendantes. Considérer une chaîne de traitements indépendants, les sorties des uns servant d'entrées aux autres, entraîne un phénomène de cumul des erreurs. L'analyse produite par un

module dépend des symboles découverts par les modules précédents sans remise en cause de ces derniers. Il en résulte qu'une analyse erronée dès les premiers niveaux du *pipeline* se répercute implacablement dans les niveaux supérieurs entraînant une détérioration des performances du système d'analyse dans les tâches de haut niveau.

- *Le changement de domaine* : Les données *in-domain* annotées manuellement étant naturellement rares et coûteuses, les ressources disponibles dans un domaine spécifique sont généralement de taille modeste. La plupart des modèles sont donc développés à partir de corpus de référence composés souvent d'écrits journalistiques. Ces modèles sont ensuite appliqués à leur cas d'utilisation sans adaptation, c'est-à-dire en négligeant les variations de lexique et de constructions syntaxiques employées, causées par un changement du registre de langue ou des thèmes abordés lorsqu'on passe d'un domaine à l'autre. Une des hypothèses fondatrices du *Machine Learning* est la conservation de distribution entre corpus d'apprentissage et données d'évaluation car les *classifieurs* apprennent aussi bien des invariants du domaine que du langage. Lorsque le changement de domaine est trop brutal, l'absence d'adaptation détériore les performances du système d'analyse dans son ensemble, et enclenche donc les phénomènes de cumul d'erreurs.

Ces limitations conduisent à des pertes de performances particulièrement importantes (GALLIANO, GRAVIER et CHAUBARD 2009) dans le cas qui nous intéresse au travers du projet européen SENSEI, l'analyse de la langue non-canonique telle que les transcriptions de conversations téléphoniques. La présence de disfluences et de constructions syntaxiques spécifiques à la langue orale ainsi que les erreurs de reconnaissance dans les transcriptions automatiques mènent à une détérioration importante des performances des systèmes d'analyse lorsqu'on les applique à la langue spontanée (Alexis NASR, BECHET, FAVRE et al. 2014). Il est nécessaire de mettre au point des systèmes d'analyse à la fois robustes et flexibles à travers les aspects suivants :

- *Système global* : L'approche *pipeline* envisage une résolution séquentielle de ces différentes tâches d'intérêt. On préférerait produire une analyse plus globale, considérant simultanément un ensemble de tâches. Un système multitâche applicable à l'ensemble des analyses du TAL permettrait de limiter la propagation des erreurs d'analyses en s'axant sur une résolution conjointe des différentes tâches.
- *Modèle générique* : L'approche *pipeline* considère des modules spécifiques à chacune des analyses. On souhaite uniformiser les modèles afin de plus facilement les entremêler au sein d'un système d'analyse globale.
- *Adaptation de données* : Afin de pallier le manque de données *in-domain* annotées manuellement, on désirerait recourir à un modèle capable de ti-

rer profit de vastes ressources de référence tout en prenant en compte les spécificités d'un domaine d'application, notamment par le biais de corpus annotés automatiquement.

Afin de répondre à ces problématiques, nous nous basons sur les méthodes d'apprentissage automatique de réseaux de neurones profonds. Nous nous intéressons notamment aux modèles *End-to-End* i.e utilisant des données avec le moins de prétraitement possible en entrée, sans recourir à des représentations intermédiaires explicites telles que celles produites par la sélection de *features*. De plus, étant donné la nature séquentielle de la langue, nous nous sommes focalisé sur des architectures récurrentes (RNN) spécifiques à l'analyse de séquences. Enfin, nous utilisons l'extraction automatique de représentations vectorielles de mots (*word embedding*) afin de proposer des méthodes d'adaptation permettant de traiter les mots inconnus ou rares.

**Contributions** Notre travail propose de s'abstraire des limitations inhérentes au *pipeline* à l'aide de modèles issus du *Deep Learning*. La mise au point d'un système unifié, multitâche et adaptable aux spécificités d'un domaine d'application, tout en restant efficace, nécessite :

- *Un modèle RNN End-to-End* (cf. 3.2.1) applicable à différentes analyses du TAL formalisées comme autant de problèmes de classification. Au prix d'hypothèses simplificatrices, les tâches d'analyse syntaxique et sémantique peuvent se réduire à des combinaisons d'opérations d'étiquetage, de segmentation et de détection de relation. Et au prix d'abstractions supplémentaires dans les annotations, on peut ramener l'ensemble de ces opérations à des problèmes de classification. Ce modèle générique résout indépendamment chaque tâche, en ne considérant que des séquences de mots en entrées.
- *Une prise en compte du contexte allant au-delà de la phrase* (cf. 3.2.3) afin d'éviter tout *a priori* sur la tâche. Cette approche permet la possibilité de gérer les phénomènes dépassant les frontières de la phrase, tel que les anaphores pronominales par exemple. Le corpus accumulé est considéré comme un flux continu de mots et on entraîne une structure récurrente qu'on pourrait déplier (*unfold*) sur l'intégralité du corpus. Cette caractéristique favorise la robustesse de l'algorithme d'apprentissage.
- *Un algorithme d'apprentissage multitâche* (cf. 3.2.2) basé sur l'entraînement joint de plusieurs instances de cette architecture générique. Le système qui en découle réalise l'ensemble des analyses en partageant des représentations cachées. Les couches cachées partagées produisent des représentations optimisées pour l'ensemble des tâches qui sont classées indépendamment par chaque couche de sortie spécifique. Ceci permet la remise en

cause des prédictions relatives à chaque niveau d'analyse, et évite des phénomènes de cumul des erreurs.

- Ce même algorithme permet de transformer des *features* d'entrée en sortie additionnelle. Ceci permet l'utilisation en entraînement de *features* qui seront indisponibles ou fortement bruitées lors de la phase d'évaluation sur des données inconnues. Cette approche permet de rendre les prédictions indépendantes des modèles d'analyses préliminaires, ou plus précisément, de la qualité des annotations produites par ces analyses. Cette caractéristique favorise également la robustesse de l'algorithme d'apprentissage. On montre notamment qu'ajouter un grand nombre de sorties additionnelles ne dénature pas les prédictions du système.
- *Des stratégies d'adaptation lexicale* (cf. 4.4) basées sur la représentations des mots à l'aide de *word embeddings*. On tire profit de leur nature distributionnelle afin de proposer des adaptations d'espaces de représentations. Cette méthodologie permet notamment de fournir des représentations plus pertinentes dans le cas de mots rares ou inconnus, dont les proportions peuvent être fortement modifiées lors d'un changement de domaine. On évalue nos stratégies en étudiant spécifiquement leur impact sur la gestion des mots hors vocabulaires (*OOV handling*).

Afin d'éprouver et de valider nos hypothèses, nous avons réalisé, une architecture RNN *End-to-End* évaluée sur différents corpus et tâches de référence (*benchmarks*) du TAL. Les performances "état-de-l'art" obtenues valident les choix effectués. Ceci nous a permis de mener une évaluation comparative entre l'apprentissage multitâche de RNNs et un *pipeline* de RNNs. Cette évaluation est réalisée sur le même ensemble de corpus de référence. On observe ainsi des gains de performances en limitant le cumul des erreurs inhérent au *pipeline* (cf. 3.3). De plus, la possibilité d'utiliser des sorties additionnelles permet de rendre le système plus robuste aux erreurs d'analyse préliminaire. Nous avons par la suite, appliquée notre approche à l'analyse sémantique de transcriptions de conversations téléphoniques annotées automatiquement par un système *pipeline* classique (cf. 4.2). En entraînant un système réalisant l'ensemble de ces analyses syntaxique et sémantique directement à partir des séquences de mot, nous obtenons de meilleures performances que le *pipeline* ayant annoté les exemples d'apprentissage (cf. 4.3.2). Ces résultats nous ont permis de considérer l'analyse de transcriptions automatiques contenant des erreurs de reconnaissance. Les erreurs commises par les systèmes de reconnaissances de la parole (*ASR*) entraînent des analyses erronées dès les premiers niveaux d'analyse qui se répercutent dans les niveaux supérieurs. Dans le cas de transcriptions automatiques présentant un taux d'erreur mot (*WER*) élevé, les *features* produites automatiquement sont inévitablement de mauvaise qualité. Leur utilisation en tant qu'entrées des modèles traditionnels, fortement dépendants de la bonne qualité de leur *fea-*

tures, se solde généralement par une détérioration importante des performances de ces derniers. Notre modèle, s'abstrayant en partie des analyses préliminaires, présente une meilleure robustesse aux erreurs de transcription (cf. 4.3.3). Enfin, l'étude systématique de l'impact de la quantité de données disponibles en entraînement sur les performances que nous avons réalisées tout au long de ces évaluations a permis de déterminer que notre approche est particulièrement performante lorsque peu d'exemples d'apprentissage sont disponibles. Tronquer le corpus d'apprentissage permet également de faire varier artificiellement les proportions de mots inconnus afin d'étudier l'impact de nos stratégies d'adaptation (cf. 4.4).

Le présent document s'organise comme suit. Nous décrivons dans le chapitre 1 le formalisme associé aux modèles du *Deep Learning* afin de présenter la théorie des réseaux de neurones et de fournir quelques clefs de compréhension. Nous nous pencherons successivement sur le neurone formel ou *Perceptron*, le *Perceptron* multi-couche, les réseaux de neurones récurrents et leurs algorithmes d'apprentissage respectifs.

Dans un second temps, nous aborderons dans le chapitre 2 les hypothèses et spécificités nécessaires à leur application au traitement automatique de la langue. Après avoir présenté les tâches d'analyse syntaxique auxquelles on s'intéressera, nous montrerons comment ramener la réalisation de ces tâches à la résolution d'un problème de classification. La fin de ce chapitre est consacrée à la présentation des *word embeddings* et des algorithmes non-supervisés dont ils sont issus.

Le chapitre 3 concerne notre modèle *End-to-End* multitâche. On y présente une architecture neuronale générique, dédiée à l'ensemble des tâches du traitement automatique de la langue, permettant une prise en compte du contexte allant au-delà des frontières de la phrase et un algorithme d'apprentissage multitâche supervisé responsable de l'entraînement conjoint de plusieurs instances de cette architecture. Les performances de notre modèle seront évaluées sur des corpus de références de l'analyse syntaxique.

Enfin, le chapitre 4 présente l'application de notre modèle à la compréhension de la langue orale et plus précisément à la détection de cadres sémantiques (*semantic frames*) dans des transcriptions manuelles et automatiques de conversations téléphoniques rassemblées au sein du corpus DECODA. Alors, des stratégies d'adaptation lexicale à partir de méthodes basées sur les *word embedding* seront proposées et leur impact sera étudié sur la gestion des mots hors vocabulaires.

# Notations

- $x$  représente un scalaire ;
- $\mathbf{x}$  représente un vecteur *i.e.*  $\mathbf{x} = (x_1, \dots, x_n, \dots, x_N)$  ;
- $X$  représente une matrice *i.e.*  $X = (\mathbf{x}_1, \dots, \mathbf{x}_m, \dots, \mathbf{x}_M)$  ;
- $x^{1:T}$  représente une séquence de taille  $T : x^1, \dots, x^t, \dots, x^T$  ;
- $\frac{\partial f}{\partial \theta}$  représente la dérivée partielle de  $f$  par rapport à  $\theta$  ;
- $\cdot$  représente le produit matriciel ;
- $\odot$  représente le produit terme-à-terme ;
- $\bullet$  représente la concaténation ;
- $\circ$  représente la composition de fonction.

# 1 Formalisme des réseaux de neurones profonds

Les objectifs de cette thèse sont l'analyse, la conception, le développement et l'implémentation de modèles issus de l'apprentissage automatique et plus spécifiquement du *Deep Learning*, appliqués à la résolution de tâches d'analyse linguistique du Traitement Automatique de la Langue. Ce chapitre est consacré à une présentation de la théorie des réseaux de neurones et sera l'occasion de proposer quelques clefs de compréhension. Il débute par la définition du neurone formel ou *Perceptron* (section 1.2), la plus simple instance des réseaux de neurones. Cette définition sera étendue afin de construire un Perceptron multi-couche (section 1.3) qui permettra de présenter les réseaux sans rétro-action composés de couches denses. Il se clôture par la présentation des réseaux de neurones récurrents (section 1.4) et leurs algorithmes d'apprentissage respectifs.

## 1.1 Contexte et Motivations

L'apprentissage automatique (ou *Machine Learning*) est un champ d'étude de l'intelligence artificielle, qui concerne la conception et le développement de méthodes permettant d'entraîner une *machine* à réaliser une tâche de prédiction. Considérons un ensemble d'observations annotées, rassemblé au sein d'un *corpus*  $C_{\text{task}}$ . Il peut s'agir de phrases où chaque mot est étiqueté par sa classe grammaticale (nom, verbe, etc.), d'un enregistrement audio d'une allocution accompagné de sa transcription écrite, ou encore d'images pour lesquelles on a renseigné si elles contenaient, ou non, un type défini d'objet (une voiture, un visage, etc.). Une *machine*  $\mathcal{F}_\theta$  est une fonction paramétrique<sup>a</sup> qui prend en entrée l'observation  $\mathbf{x}$  et produit en sortie l'annotation  $y$ . On souhaite modifier par un processus systématique les paramètres  $\theta$  de  $\mathcal{F}_\theta$  afin de minimiser l'erreur de prédiction sur le *corpus* d'apprentissage. Autrement dit, on souhaite trouver les valeurs de  $\theta$  tel que  $\forall (\mathbf{x}, y) \in C_{\text{task}} \mathcal{F}_\theta(\mathbf{x}) = y$ .

Un modèle est entraîné sur une base d'apprentissage en minimisant un critère de performance permettant d'estimer la qualité des prédictions du modèle sur les exemples d'apprentissage. Cette approche est valide si on suppose l'hypothèse que la distribution des exemples dans  $C_{\text{task}}$  est représentative de l'ensemble

---

a. Cela signifie que l'image  $\mathcal{F}_\theta(\mathbf{x})$  dépend de la variable  $\mathbf{x}$  et de paramètres internes  $\theta$ .

des observations qu'on pourrait être amené à analyser. On espère ainsi que le modèle entraîné soit capable de généraliser ses prédictions sur des observations non-fournies en tant qu'exemple d'apprentissage.

Plus spécifiquement, l'apprentissage automatique à l'aide de réseaux de neurones profonds (ou *Deep Learning*) se base sur l'entraînement de modèles composés de nombreuses couches de neurones formant une fonction paramétrique. Ils sont notamment appliqués à la réalisation de tâches de classifications (étiquetage, reconnaissance de motif, prévision de séries temporelles, etc.). Le *Deep Learning* utilise un algorithme d'optimisation par rétro-propagation du gradient de l'erreur commise par le modèle afin d'estimer comment changer ses paramètres internes en vue de minimiser cette erreur. Grâce à l'augmentation de la puissance de calcul et de la quantité de données disponibles, il est désormais possible d'entraîner efficacement des architectures neuronales de plus en plus complexes. L'ensemble de ces méthodes a significativement amélioré les performances "état-de-l'art" dans des domaines tels que la transcription automatique de l'oral, le traitement automatique de l'écrit, la reconnaissance de visage ou de façon générale l'analyse de texte, d'image, de son et de vidéo.

## 1.2 Neurones artificiels

Dès le milieu du siècle dernier, à la suite des pionniers de l'informatique, l'objectif des chercheurs était de construire un modèle capable de reproduire certains aspects de l'intelligence humaine. Cette problématique de modélisation inspire en 1943 la première définition du neurone formel. Il s'agit d'un neurone binaire, c'est-à-dire dont la sortie vaut  $+1$  ou  $-1$ . Pour calculer cette sortie, le neurone effectue une somme pondérée de ses entrées puis applique une fonction d'activation à seuil : la sortie du neurone est  $+1$  si et seulement si la somme pondérée dépasse une certaine valeur. Malgré l'apparente simplicité de cette modélisation, ou peut-être grâce à elle, le neurone formel de McCulloch et Pitts reste aujourd'hui encore un élément de base des réseaux de neurones artificiels.

## 1.2.1 Définition

Un neurone artificiel est une fonction paramétrique  $\mathcal{F}_\theta$  de  $\mathbb{R}^N$  dans  $\mathbb{R}$  (éventuellement ramené à  $[-1; +1]$ <sup>b</sup> suivant la fonction d'activation considérée). Ce neurone calcule un potentiel défini comme la somme des signaux d'entrée  $\mathbf{x} = (x_1, \dots, x_n, \dots, x_N)$  pondérés par des poids  $\mathbf{w} = (w_1, \dots, w_n, \dots, w_N)$  associés aux connexions. La sortie du neurone en fonction de son potentiel est :

$$\mathcal{F}_\theta(\mathbf{x}) = \varphi \left( \sum_{n=1}^N w_n x_n + b \right) \quad (1.1)$$

où  $\varphi(\cdot)$  est la fonction d'activation<sup>c</sup> du neurone. Plusieurs choix sont possibles parmi lesquels, par exemple, la fonction tangente hyperbolique  $\tanh$ . Cette fonction est strictement croissante, à valeur dans  $[-1; +1]$  et telle que  $\varphi(0) = 0$ . Si le produit scalaire  $\mathbf{w} \cdot \mathbf{x}$  est plus grand qu'un certain seuil  $b$  (ou biais), le neurone est actif *i.e.*  $\mathcal{F}_\theta(\mathbf{x}) > 0$ . Les poids  $w_i$  peuvent prendre des valeurs positives ou négatives et font partie (avec le biais  $b$ ) de l'ensemble des paramètres  $\theta$  du neurone.

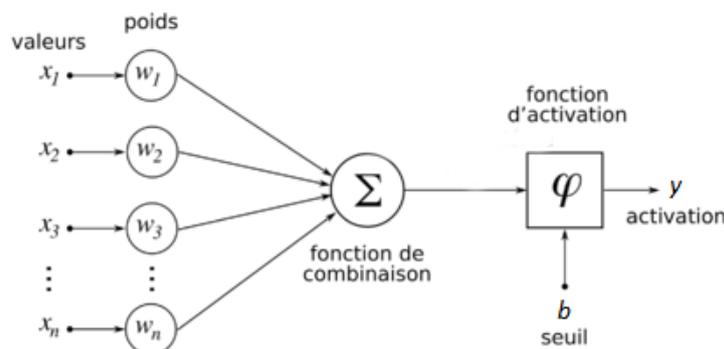


Figure 1.1 – Neurone formel

Il est possible d'interpréter géométriquement l'équation de sortie du neurone artificiel en considérant, dans un espace de dimension  $N$ , le vecteur d'entrée  $\mathbf{x} = (x_1, \dots, x_N)$  et le vecteur de poids  $\mathbf{w} = (w_1, \dots, w_N)$ . Soit l'hyperplan défini par sa normale  $\mathbf{w}$  et sa distance à l'origine  $b$ . Si  $\mathbf{w} \cdot \mathbf{x} = 0$  alors  $\mathbf{x}$  est sur l'hyperplan. Et plus généralement, le signe de ce produit scalaire indique de quel côté de l'hyperplan se trouve  $\mathbf{x}$ . Donc un neurone sépare en deux l'espace des données à l'aide de cet hyperplan.

b. ou de façon équivalente  $[0; 1]$  *via* le changement de variables :  $j = 2i - 1$ .

c. Plusieurs choix de fonction d'activation sont possibles et seront présentés dans la *section 1.3*

## 1.2.2 Perceptron pour la classification binaire

Ce modèle de neurone artificiel peut-être utilisé afin de réaliser un système de classification. L'algorithme du *Perceptron* de Rosenblatt (1957) est la première et la plus simple implémentation des réseaux de neurones. Il s'agit d'un algorithme d'apprentissage supervisé, dédié à la classification binaire *i.e.* dont le but est d'étiqueter une entrée en l'associant ou non à une classe d'intérêt. On utilise dans ce cas la fonction d'activation  $\varphi(x) = \text{signe}(x)$ . Les sorties attendues  $y$  et calculées  $\mathcal{F}_\theta(\mathbf{x})$  sont à valeur dans  $\{-1; +1\}$ . On peut voir les entrées comme un espace de description (attributs binaires ou réels), formant un nuage de points, et le Perceptron comme une procédure de séparation linéaire de cet espace selon les sorties attendues. Cet algorithme requiert des données d'apprentissage, qu'on peut écrire sous la forme  $C_{\text{train}} = \{(\mathbf{x}, y)\}$  avec  $\mathbf{x} \in \mathbb{R}^N$  (où  $N$  est le nombre d'attributs des données d'entrée) et  $y \in \{-1; +1\}$ .

$$\mathcal{F}_\theta(\mathbf{x}) = \text{signe}(\mathbf{w} \cdot \mathbf{x} + b) \quad (1.2)$$

Un Perceptron binaire est un *classifieur* linéaire. À partir des données d'entraînement  $(\mathbf{x}, y)$ , l'objectif est d'apprendre un vecteur  $\mathbf{w} \in \mathbb{R}^N$  et un scalaire  $b$  tels que  $\mathbf{w} \cdot \mathbf{x} + b > 0$  pour tout  $\mathbf{x}$  appartenant à la classe  $y = +1$  et que  $\mathbf{w} \cdot \mathbf{x} + b \leq 0$  pour tout  $\mathbf{x}$  appartenant à la classe  $y = -1$ . Autrement dit, on souhaite que tout point  $\mathbf{x}$  associé à la classe  $+1$  dans  $C_{\text{train}}$  soit "au-dessus" de l'hyperplan défini par sa normale  $\mathbf{w}$  et sa distance à l'origine  $b$ . Réciproquement, on désire que tout point  $\mathbf{x}$  associé à la classe  $-1$  soit "en-dessous" de cet hyperplan.

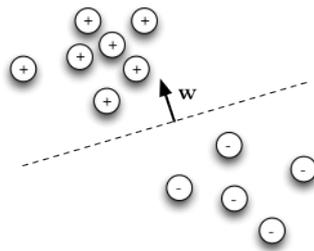


Figure 1.2 – Exemple de séparateur linéaire

Considérons  $C_{\text{train}} = \{(\mathbf{x}, y)\}$  un sous ensemble de  $\mathbb{R}^N \times \{-1; +1\}$  correspondant à des exemples d'apprentissage. On note  $S_+ = \{\mathbf{x} \in \mathbb{R}^N \mid (\mathbf{x}, +1) \in C_{\text{train}}\}$  les exemples positifs et  $S_- = \{\mathbf{x} \in \mathbb{R}^N \mid (\mathbf{x}, -1) \in C_{\text{train}}\}$  les exemples négatifs. On dit que  $S$  est *linéairement séparable* s'il existe un hyperplan  $H$  de  $\mathbb{R}^N$  tel que les ensembles  $S_+$  et  $S_-$  soient situés de part et d'autre de cet hyperplan.

---

**Algorithm 1** Perceptron Binaire

---

**Require:** Poids  $\mathbf{w} \in \mathbb{R}^N$

**Require:** Biais  $b \in \mathbb{R}$

**Require:** Ensemble d'apprentissage  $C_{\text{train}} = (\mathbf{x}_i, y_i)_{1..n}$

**while**  $\exists (\mathbf{x}, y) \in C_{\text{train}}$  tq.  $\mathcal{F}_\theta(\mathbf{x}) \neq y$  **do**

    Tirer aléatoirement un exemple  $\mathbf{x}_i, y_i$

$\mathcal{F}_\theta(\mathbf{x}_i) \leftarrow \text{signe}(\mathbf{w} \cdot \mathbf{x}_i + b)$

**if**  $\mathcal{F}_\theta(\mathbf{x}_i) \neq y_i$  **then**

$\mathbf{w} \leftarrow \mathbf{w} + y_i \mathbf{x}_i$

**end if**

**end while**

**return**  $\mathbf{w}, b$

---

L'algorithme d'apprentissage peut être décrit succinctement comme une procédure itérative de correction d'erreur. Le modèle ajuste automatiquement ses paramètres internes  $(w, b)$  à chaque prédiction (la sortie calculée) afin de minimiser l'erreur de classification sur un ensemble d'exemples {entrée, sortie attendue} d'*entraînement*. Autrement dit, on initialise les poids du Perceptron à des valeurs quelconques et à chaque fois que l'on présente un nouvel exemple, on ajuste les poids si la sortie calculée diffère de la sortie attendue.

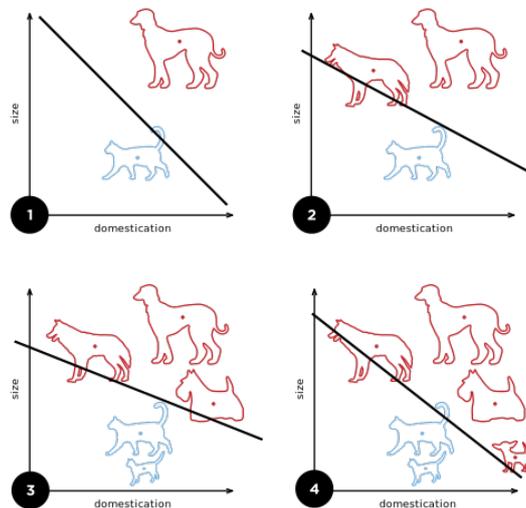


Figure 1.3 – Exemple fictif d'exécution du Perceptron permettant d'observer l'évolution de l'hyperplan séparateur calculé à mesure que les exemples d'apprentissage sont pris en compte. - [en.wikipedia.org](http://en.wikipedia.org)

Soit un exemple  $(\mathbf{x}, y) \in S$  mal classé. Si la sortie attendue  $y = +1$  alors on modifie  $\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}$  et réciproquement, si  $y = -1$  alors  $\mathbf{w} \leftarrow \mathbf{w} - \mathbf{x}$ . En modifiant ainsi sa normale, l'hyperplan s'oriente pour corriger les erreurs de classification détectées. Si  $S$  est linéairement séparable, l'algorithme finit par converger et produit un hyperplan discriminant parfaitement les sous-ensembles  $S_+$  et  $S_-$ . L'algorithme s'arrête donc lorsque tous les exemples ont été présentés et classés sans erreur.

### 1.2.3 Perceptron multi-classes

Un Perceptron multi-classes généralise le principe de classification linéaire du Perceptron binaire au cas où le nombre de classes est supérieur à deux. Afin d'étendre la classification binaire à la classification à  $M$  classes, il faut considérer l'entraînement d'une couche de  $M$  neurones. Chaque neurone est spécifique à la détection d'une classe en particulier. Une couche de neurones est représentée par une matrice  $W$  où la chaque ligne correspond à un vecteur  $\mathbf{w}_m$  associé à chacune des classes  $c_m$  possibles.

On souhaite prédire  $\mathcal{F}_\theta(\mathbf{x}) = c_m$  si et seulement si  $\forall k \neq m \mathbf{w}_k \cdot \mathbf{x} + b_k < \mathbf{w}_m \cdot \mathbf{x} + b_m$ . Autrement dit, afin de réaliser une prédiction, on s'intéresse au neurone d'activation maximale. De ce point de vue, il est possible d'interpréter les termes de biais  $\mathbf{b}$  comme une distribution *a priori* des classes. À partir des données d'entraînement  $(\mathbf{x}, y)$  où désormais  $y \in \{c_1, \dots, c_M\}$ , l'objectif est d'apprendre une matrice  $W \in \mathbb{R}^{M \times N}$  et un vecteur  $\mathbf{b} \in \mathbb{R}^M$  tels que pour tout exemple  $(\mathbf{x}, y)$ ,  $\underset{m=1..M}{\operatorname{argmax}}(\mathbf{w}_m \cdot \mathbf{x} + b_m) = y$ .

$$\mathcal{F}_\theta(\mathbf{x}) = \underset{m=1..M}{\operatorname{argmax}}(W \cdot \mathbf{x} + \mathbf{b}) \quad (1.3)$$

Tout comme le Perceptron binaire, il existe un algorithme capable d'ajuster automatiquement ces paramètres  $\theta = W, \mathbf{b}$  relatifs aux prédictions  $\mathcal{F}_\theta(\mathbf{x})$  afin de minimiser l'erreur de classification sur un ensemble d'exemples ( entrée  $\mathbf{x}$ , sortie attendue  $y$  ). En cas de prédiction erronée, on ajuste les neurones relatifs à la classe attendue  $\mathbf{w}_y$  et à la classe prédite  $\mathbf{w}_{\mathcal{F}_\theta(\mathbf{x})}$ .

Le neurone artificiel et les Perceptrons qui en découlent sont des modèles linéaires. Cela provient du fait qu'un neurone à  $N$  entrées divise  $\mathbb{R}^N$  en deux sous-espaces délimités par un hyperplan. Cependant, cela signifie également qu'ils peuvent uniquement résoudre les problèmes dont les exemples d'entrée sont tels

---

**Algorithm 2** Perceptron Multi-classes

---

**Require:** Poids  $W \in \mathbb{R}^K \times N$   
**Require:** Biais  $\mathbf{b} \in \mathbb{R}^K$   
**Require:** Ensemble d'apprentissage  $C_{\text{train}} = (\mathbf{x}_i, y_i)_{1..n}$   
  **while**  $\exists (\mathbf{x}, y) \in C_{\text{train}}$  tq.  $\mathcal{F}_\theta(\mathbf{x}) \neq y$  **do**  
    Tirer aléatoirement un exemple  $(\mathbf{x}, y)$   
     $\mathcal{F}_\theta(\mathbf{x}) \leftarrow \underset{k \in [1, K]}{\text{argmax}} (\mathbf{w}_k \cdot \mathbf{x} + b_k)$   
    **if**  $\mathcal{F}_\theta(\mathbf{x}) \neq y$  **then**  
       $\mathbf{w}_y \leftarrow \mathbf{w}_y + \mathbf{x}$   
       $\mathbf{w}_{\mathcal{F}_\theta(\mathbf{x})} \leftarrow \mathbf{w}_{\mathcal{F}_\theta(\mathbf{x})} - \mathbf{x}$   
    **end if**  
  **end while**  
**return**  $W, \mathbf{b}$

---

qu'il existe un hyperplan séparateur. Pour certaines tâches de classification, il est nécessaire d'utiliser des réseaux comportant plusieurs couches d'unités interconnectées, entrecoupées de fonctions d'activation non-linéaires afin d'augmenter grandement l'expressivité<sup>d</sup> du modèle. C'est ce qu'on appelle des Perceptron multi-couches ou des réseaux *feed-forward*. Ils seront l'objet de la section suivante.

## 1.3 Réseaux *feed-forward*

Un réseau de neurones est défini par son architecture : le nombre de neurones, le nombre de poids, la topologie du graphe des connexions et la disposition des entrées et sorties. Une architecture particulièrement utilisée est celle des réseaux sans boucles ou rétroactions (*feed-forward networks*). Dans cette architecture, les neurones sont agencés en couches successives avec des connexions allant uniquement d'une couche à la suivante.

Ces réseaux de neurones *feed-forward* sont des approximateurs universels parcimonieux (BALDI et HORNIK 1989). La propriété d'approximation peut être énoncée de la façon suivante : toute fonction bornée suffisamment régulière peut être approchée avec une précision arbitraire par un réseau de neurones comportant une couche de neurones cachés en nombre fini, possédant tous la même fonction d'activation, et un neurone de sortie linéaire. En d'autres termes,

---

d. L'ensemble des fonctions atteignables *i.e.* qu'il peut simuler en faisant varier ses paramètres

une seule couche cachée suffit pour réaliser n'importe quelle fonction des entrées même si la taille<sup>e</sup> optimale de la couche cachée reste théoriquement inconnue. Cette propriété n'est pas spécifique aux réseaux de neurones, d'autres familles de fonctions paramétrées possèdent cette propriété. La spécificité des réseaux de neurones réside dans le caractère « parcimonieux » de l'approximation : à précision égale, les réseaux de neurones nécessitent moins de paramètres ajustables que les autres approximateurs connus.

### 1.3.1 Perceptron multi-couches

Les Perceptrons multi-couches sont une application des réseaux *feed-forward* à la résolution de tâches de classification et à l'approximation de fonctions non-linéaires. Ce sont des modèles non-linéaires qui peuvent être généralement utilisés en lieu et place d'un modèle linéaire traditionnel. Leur faculté de simuler des fonctions non-linéaires appliquées à des entrées plongées dans des espaces de représentation de très grandes dimensions, permet d'obtenir des performances comparables aux modèles qu'ils remplacent. De nombreux travaux ont notamment prouvé l'utilité de tels réseaux dans le cadre de l'analyse d'image, du signal ou encore de la langue écrite et orale.

Un Perceptron multi-couches est composé d'une succession de couches de neurones artificiels analogues à celles introduites dans l'équation 1.3. Les données encodées numériquement sont injectées dans la couche d'entrée puis transformées successivement par les couches intermédiaires, dites cachées. La dernière couche classe la représentation calculée dans la dernière couche cachée afin d'effectuer une prédiction. Cette couche de sortie est composée d'autant de neurones que le problème compte de classes.

#### 1.3.1.1 Définition

Considérons un réseau de neurones  $\mathcal{F}_\theta(\cdot)$  de paramètres  $\theta$ . Tout réseau de neurones *feed-forward* à  $L$  couches peut être décrit comme une composition de fonctions  $\mathcal{F}_{\theta_i}(\cdot)$  :

$$\mathcal{F}_\theta(\mathbf{x}) = \mathcal{F}_{\theta_L} \circ \mathcal{F}_{\theta_{L-1}} \circ \dots \circ \mathcal{F}_{\theta_1}(\mathbf{x}) = \hat{\mathbf{y}} \quad (1.4)$$

---

e. Le nombre de neurones qu'elle contient

où  $\mathcal{F}_{\theta_l}(\cdot)$  représente la fonction associée à la  $l^{\text{ième}}$  couche de neurones cachés. Admettons que cette couche contienne  $K$  neurones auxquels sont envoyés le même signal d'entrée  $\mathbf{x} \in \mathbb{R}^N$ . À chacun de ces neurones est associé un vecteur de poids  $\mathbf{w}_k$  formant la matrice de poids  $W = (\mathbf{w}_1 \cdots \mathbf{w}_k \cdots \mathbf{w}_K) \in \mathbb{R}^{K \times N}$  associée à cette couche. Appelons  $\mathbf{h}_l$  le vecteur des activations de la  $l^{\text{ième}}$  couche cachée.

$$\mathbf{h}_l = \varphi(W_l \cdot \mathcal{F}_{\theta_{l-1}}(\mathbf{x}) + \mathbf{b}_l) \in \mathbb{R}^K \quad (1.5)$$

La première couche de connexions injecte les entrées  $\mathbf{x}$  dans la première couche cachée. Les activations cachées  $\mathbf{h}$  remontent la ou les couches cachées non-linéaires jusqu'à la couche de sortie pour produire  $\hat{\mathbf{y}}$ . Dans le cas d'un réseau à une seule couche cachée, on obtient :

$$\mathbf{h} = \mathcal{F}_{\text{HIDDEN}}(\mathbf{x}) = \varphi(W_1 \cdot \mathbf{x} + \mathbf{b}_1) \quad (1.6)$$

$$\hat{\mathbf{y}} = \mathcal{F}_{\text{OUTPUT}}(\mathbf{h}) = W_2 \cdot \mathbf{h} + \mathbf{b}_2 \quad (1.7)$$

### 1.3.1.2 Fonctions non-linéaires

Dans les équations précédentes, la fonction  $\varphi$  est la fonction d'activation des neurones artificiels. Différents choix sont possibles. Les neurones binaires ont des fonctions d'activation discontinues. Suivant que les états soient codés sur  $\{-1; +1\}$  ou  $\{0; 1\}$ , on a la fonction *signe* ou la fonction échelon  $\Theta(x)$  :

$$\varphi(x) = \text{signe}(x) \equiv \begin{cases} +1 & \text{si } x \geq 0 \\ -1 & \text{autrement} \end{cases} \quad (1.8)$$

$$\varphi(x) = \Theta(x) \equiv \begin{cases} 1 & \text{si } x \geq 0 \\ 0 & \text{autrement} \end{cases} \quad (1.9)$$

Suivant les cas, on utilise soit la fonction logistique  $\sigma$  à valeur dans  $[0; 1]$ , soit la tangente hyperbolique  $\tanh$  à valeur dans  $[-1; +1]$  ou encore la fonction *ReLU* (Unité Rectifié linéaire) à valeur dans  $\mathbb{R}^+$  :

$$\varphi(x) = \sigma(x) \equiv \frac{1}{1 + e^{-x}} \quad (1.10)$$

$$\varphi(x) = \tanh(x) \equiv \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (1.11)$$

$$\varphi(x) = ReLU(x) \equiv \max(0, x) \quad (1.12)$$

Si on souhaite utiliser plusieurs couches de neurones, on utilise ces fonctions non-linéaires afin d'accroître la capacité d'approximation des modèles considérés. Il n'existe pour autant aucune preuve théorique qu'une fonction non-linéaire en particulier soit meilleure que les autres.

### 1.3.1.3 Normalisation

Dans la plupart des cas d'apprentissage multi-classes, plutôt que de prédire *stricto sensu* la classe d'un exemple, on désire prédire sa probabilité d'appartenance à chacune des classes possibles. Les vecteurs de sortie du réseaux sont alors normalisés par application d'une transformation *softmax* :

$$\hat{\mathbf{y}} = y_1, \dots, y_k \quad (1.13)$$

$$softmax(y_i) \equiv \frac{e^{y_i}}{\sum_{j=1}^k e^{y_j}}$$

où  $\hat{\mathbf{y}}$  est défini par l'équation 1.7. Il en résulte un vecteur de  $\mathbb{R}_+^*$  dont les composantes somment à 1. Ce vecteur peut ainsi être interprété comme une distribution de probabilité sur  $k$  résultats possibles. La fonction *softmax* est utilisée lorsqu'on désire modéliser la distribution de probabilité d'émission des différentes classes de sorties. Elle est généralement couplée avec un apprentissage minimisant une fonction de coût probabiliste telle que la *cross-entropy* qui sera définie dans la section suivante.

## 1.3.2 Algorithme d'apprentissage

L'entraînement d'un réseau de neurones est réalisé par minimisation d'une fonction de coût sur l'ensemble des exemples d'apprentissage par descente de gradient. Cette fonction de coût représente la qualité des prédictions du réseau de neurones sur les données d'apprentissage. Cette section est consacrée à la description de l'algorithme d'apprentissage par rétro-propagation du gradient de la *Stochastic Gradient Descent* (LECUN, BOTTOU, ORR et al. 1998), élément essentiel à l'entraînement des réseaux de neurones composés de plusieurs couches cachées (Y. LECUN, L. BOTTOU, Y. BENGIO et al. 2001).

### 1.3.2.1 Fonction de coût

L'entraînement d'un réseau de neurones passe par l'optimisation d'une fonction de coût définie par  $\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})$  à valeur dans  $\mathbb{R}_+$ , estimateur de l'erreur commise en prédisant  $\hat{\mathbf{y}} = \mathcal{F}_\theta(\mathbf{x})$  lorsque la sortie attendue est  $\mathbf{y}$ . L'objectif de l'entraînement est de minimiser l'erreur sur l'ensemble des exemples d'apprentissage  $C_{\text{train}}$ . La fonction de coût est positive et ne s'annule que lorsque la prédiction est correcte. Comme nous le verrons, les paramètres internes du réseau, responsables de cette prédiction, seront automatiquement modifiés afin de minimiser  $\mathcal{L}$ . L'objectif de l'entraînement est de trouver l'état paramétrique  $\theta^*$  tel que :

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \sum_{C_{\text{train}}} \mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) \quad (1.14)$$

où les exemples d'entrée/sortie désirés sont fixés, et la fonction de coût  $\mathcal{L}$  est une fonction de paramètres  $\theta$ .

**Moindres carrés** La fonction d'erreur quadratique moyenne (*Mean Square Error*, MSE) est définie pour un exemple  $(\mathbf{x}, \mathbf{y})$  comme la déviation entre l'image calculée  $\hat{\mathbf{y}} = \mathcal{F}_\theta(\mathbf{x})$  et le vecteur  $\mathbf{y}$  attendu.

$$\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) = \|\hat{\mathbf{y}} - \mathbf{y}\|^2 \quad (1.15)$$

**Entropie croisée** La fonction *cross-entropy* est utilisée lorsqu'on désire interpréter les sorties du réseau comme des distributions de probabilités sur les différentes classes. Soient  $\mathbf{y} = y_1, \dots, y_n$  un vecteur représentant la vraie distribution de probabilité sur les classes  $1, \dots, n$  et  $\hat{\mathbf{y}} = \hat{y}_1, \dots, \hat{y}_n$  la sortie du réseau après normalisation par un *softmax* représentant la distribution de probabilités

$\hat{y}_i = P(y = i|\mathbf{x})$ . La *cross-entropy* mesure la similarité entre deux distributions de probabilités.

$$\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) = - \sum_i y_i \log(\hat{y}_i) \quad (1.16)$$

Lorsqu'on l'utilise dans le cadre de la classification, chaque exemple d'apprentissage appartient à une et une seule classe,  $\mathbf{y}$  est donc un vecteur *1-hot* représentant la classe correspondante. On obtient par simplification :

$$\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) = - \log(\hat{y}_t) \quad (1.17)$$

où  $t$  représente la classe attendue. On désire prédire une probabilité de 1 pour la classe  $t$ . Comme les sorties  $\hat{\mathbf{y}}$  ont été transformées par un *softmax* et représente des probabilités conditionnelles, alors augmenter la probabilité de la bonne classe signifie diminuer la probabilité assignée à l'ensemble des autres classes. Ce critère d'apprentissage de *cross-entropy* permet de produire des *classifiers* multi-classes ne prédisant pas qu'une hypothèse d'étiquette mais une distribution de probabilités sur l'ensemble des étiquettes possibles.

**Hinge & Ranking Losses** Un autre critère classique, hérité des machines à vecteurs de support (CORTES et VAPNIK 1995) est la fonction de Hinge. Si on considère un problème de classification binaire, la sortie d'un réseau est un scalaire  $\hat{y}$  et la sortie attendue  $y$  est à valeur dans  $\{-1, +1\}$ . La classification est considérée correcte si  $\hat{y}$  et  $y$  sont de même signe, autrement dit si  $\hat{y}y > 0$ .

$$\mathcal{L}(\hat{y}, y) = \max(0, 1 - \hat{y}y) \quad (1.18)$$

Cette fonction a été étendue à l'apprentissage multi-classes par (GRAMMER et SINGER 2002) en considérant  $\hat{\mathbf{y}}$  le vecteur de sortie du réseau et  $\mathbf{y}$  le vecteur *one-hot* de la classe correcte. Soient  $t = \underset{i}{\operatorname{argmax}}(\mathbf{y}_i)$  et  $k = \underset{i \neq t}{\operatorname{argmax}}(\mathbf{y}_i)$ .

$$\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) = \max(0, 1 - (\hat{\mathbf{y}}_t - \hat{\mathbf{y}}_k)) \quad (1.19)$$

Dans certaines conditions expérimentales, on n'a pas accès à une supervision sous forme de *labels* mais plutôt de paires de prédictions, correctes  $\hat{\mathbf{y}}_+$  et incorrectes  $\hat{\mathbf{y}}_-$ . Le but de l'apprentissage est de prédire un meilleur score pour une sortie correcte. On parle de *Ranking Hinge Loss* :

$$\mathcal{L}(\hat{\mathbf{y}}_+, \hat{\mathbf{y}}_-, \mathbf{y}) = \max(0, \|\hat{\mathbf{y}}_+ - \mathbf{y}\|^2 - \|\hat{\mathbf{y}}_- - \mathbf{y}\|^2) \quad (1.20)$$

### 1.3.2.2 Rétro-propagation du gradient

On a besoin de minimiser une fonction non-linéaire et non-convexe. Cela introduit la présence de minima locaux dans la fonction de coût qu'on explore à l'aide d'un algorithme par descente de gradient. Cette méthode consiste à considérer un état paramétrique  $\theta$  d'une fonction  $\mathcal{F}_\theta$ . Lorsqu'utilisée pour entraîner un réseau de neurones, la fonction paramétrique  $\mathcal{F}_\theta$  est le réseau, et les paramètres  $\theta$  sont les matrices de poids ainsi que les vecteurs de biais des couches de neurones.

Pour chaque exemple d'apprentissage  $(\mathbf{x}, \mathbf{y})$  on calcule le gradient de la fonction de coût vis-à-vis de chaque paramètre. Puis, on modifie les paramètres courants de sorte à effectuer un "pas" dans la direction inverse du gradient. On cherche à minimiser l'équation 1.14. L'hyper-paramètre  $\varepsilon$  représente le taux d'apprentissage. Géométriquement, si on considère que l'algorithme explore pas-à-pas la surface d'erreur  $\mathcal{L}_\theta$ , alors le gradient  $\frac{\partial \mathcal{L}}{\partial \theta}$  indique le sens de la pente et  $\varepsilon$  la longueur du pas qu'on réalise dans cette direction.

$$\theta \leftarrow \theta - \varepsilon \frac{\partial \mathcal{L}(\mathcal{F}_\theta(\mathbf{x}), \mathbf{y})}{\partial \theta} \quad (1.21)$$

Le gradient de la fonction  $\mathcal{L}$  par rapport à l'ensemble des paramètres  $\theta$  est obtenu à l'aide de la *Chain Rule*, un algorithme connu sous le nom de rétro-propagation du gradient.

### 1.3.2.3 Gradient Stochastique et Batch

L'approche traditionnelle pour entraîner des réseaux de neurones est d'utiliser l'algorithme de descente de gradient stochastique (*SGD*). L'objectif de cet algorithme est de trouver les valeurs de  $\theta$  qui minimisent le coût total  $\mathcal{L}_\theta = \sum_{i=1}^n \mathcal{L}(\mathcal{F}_\theta(\mathbf{x}_i), \mathbf{y}_i)$  sur l'ensemble des exemples d'apprentissage. L'algorithme est itératif. Il sélectionne aléatoirement un exemple d'apprentissage, et calcule le gradient de l'erreur commise par  $\mathcal{F}_\theta$  sur cet exemple vis-à-vis des paramètres  $\theta$ .

---

**Algorithm 3** Stochastic Gradient Descent Training

---

**Require:** Function  $\mathcal{F}_\theta(\mathbf{x})$  parameterized with parameters  $\theta$

**Require:** Training set of inputs  $\mathbf{x}_1, \dots, \mathbf{x}_n$  and outputs  $\mathbf{y}_1, \dots, \mathbf{y}_n$

**Require:** Loss function  $\mathcal{L}$

**while** stopping criteria not met **do**

    Sample a training example  $\mathbf{x}_i, \mathbf{y}_i$

    Compute the loss  $\mathcal{L}(\mathcal{F}_\theta(\mathbf{x}_i), \mathbf{y}_i)$

$\theta \leftarrow \theta + \varepsilon \frac{\partial \mathcal{L}(\mathcal{F}_\theta(\mathbf{x}_i), \mathbf{y}_i)}{\partial \theta}$

**end while**

**return**  $\theta$

---

L'ensemble de paramètres  $\theta$  est enfin mis à jour dans la direction inverse du gradient avec un taux d'apprentissage  $\varepsilon$ .

Dans cette version de l'algorithme, l'erreur et le gradient sont calculés pour un seul exemple d'apprentissage. Cette approche est appelée descente de gradient stochastique. Elle est connue pour permettre une convergence plus rapide, et introduit un comportement chaotique. Une approche standard pour minimiser ce phénomène est d'estimer l'erreur et le gradient sur un ensemble de  $m$  exemples. On parle d'apprentissage par *minibatch*. La taille du *minibatch* peut varier de  $m = 1$  à  $m = n$ . Une taille importante permet une meilleure estimation de l'erreur commise par  $\mathcal{F}_\theta$  sur l'ensemble d'apprentissage tandis qu'une plus petite permet d'effectuer davantage de mises à jour de  $\theta$  et accélère la convergence dans le cas de problèmes convexes. Lorsque ce n'est pas le cas, on observe que les *minibatches* de taille importante ont un effet sur la chute dans les minima locaux.

Avec un taux d'apprentissage suffisamment faible, la *SGD* garantit une convergence vers un minimum global dans le cas où la fonction  $\mathcal{F}_\theta$  est convexe. Cependant, il peut également être utilisé pour optimiser des fonctions non-convexes telles que celles produites par des réseaux de neurones. Bien qu'on n'ait aucune garantie théorique de converger vers un minimum global, cet algorithme s'est révélé robuste et efficace en pratique. Il existe de nombreuses variations de la *SGD* (*AdaGrad* : DUCHI, HAZAN et SINGER 2011 ; *AdaM* : KINGMA et BA 2014), utilisant notamment la notion de *momentum* qui confère une forme d'inertie à la rétro-propagation de l'erreur (RUMELHART, G. E. HINTON et WILLIAMS 1988).

---

**Algorithm 4** Minibatch Gradient Descent Training

---

**Require:** Function  $\mathcal{F}_\theta(\mathbf{x})$  parameterized with parameters  $\theta$

**Require:** Training set of inputs  $\mathbf{x}_1, \dots, \mathbf{x}_n$  and outputs  $\mathbf{y}_1, \dots, \mathbf{y}_n$

**Require:** Loss function  $\mathcal{L}$

**while** stopping criteria not met **do**

    Sample a minibatch of  $m$  examples  $\{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_m, \mathbf{y}_m)\}$

$\hat{\mathbf{g}} \leftarrow 0$

**for**  $i = 1$  to  $m$  **do**

        Compute the loss  $\mathcal{L}(\mathcal{F}_\theta(\mathbf{x}_i), \mathbf{y}_i)$

$\hat{\mathbf{g}} \leftarrow \hat{\mathbf{g}} + \frac{1}{m} \frac{\partial \mathcal{L}(\mathcal{F}_\theta(\mathbf{x}_i), \mathbf{y}_i)}{\partial \theta}$

**end for**

$\theta \leftarrow \theta + \varepsilon \hat{\mathbf{g}}$

**end while**

**return**  $\theta$

---

### 1.3.2.4 Régularisation

Les réseaux de neurones possèdent de très nombreux paramètres internes et sont donc particulièrement sensibles à ce qu'on appelle le sur-apprentissage. Généralement, si on entraîne un modèle suffisamment longtemps, on peut alors observer que l'erreur qu'il commet sur l'ensemble d'apprentissage continue de diminuer tandis que celle qu'il commet sur l'ensemble d'évaluation augmente. Le modèle a cessé de généraliser. On obtient ainsi un *classifier* qui reconnaît parfaitement ses données d'apprentissage mais dont la précision des prédictions est très pauvre lorsqu'on l'applique à de nouvelles données. Le sur-apprentissage s'interprète comme un apprentissage "par cœur" des données. Le modèle se comporte alors comme une table contenant tous les exemples utilisés lors de l'apprentissage et perd ses pouvoirs de prédiction sur de nouveaux exemples. Ainsi, de par sa trop grande capacité à stocker des informations, un modèle dans une situation de sur-apprentissage aura de la peine à généraliser sur des entrées inconnues.

Les méthodes de régularisation des paramètres du réseau permettent, dans une certaine mesure, d'éviter le sur-apprentissage. Une méthode classique est la régularisation  $L_2$  consistant en l'addition d'un terme  $\frac{\lambda}{2} \|\theta\|^2$  dans la fonction de coût à minimiser.

$$\mathcal{L}_r(\hat{\mathbf{y}}, \mathbf{y}) = \mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) + \frac{\lambda}{2} \|\theta\|^2 \quad (1.22)$$

On favorise ainsi les états paramétriques  $\theta$  dont les normes  $L_2$  des composantes sont petites. Autrement dit, on pénalise les valeurs extrêmes des paramètres qui correspondent souvent à une situation de sur-apprentissage. Dans ce cas, le coefficient  $\lambda$  est un hyper-paramètre contrôlant la puissance de la régularisation. Notez qu'il est possible d'appliquer cette régularisation sur tout ou seulement sur une partie de l'ensemble de paramètres  $\theta$ . Notez également qu'il est possible de réécrire le terme de régularisation sous la forme  $\frac{1}{2} \|\theta - \theta_0\|^2$ . Cette réécriture permet d'envisager de régulariser  $\theta$  vers un état paramétrique  $\theta_0$  fixé<sup>f</sup>.

Plus récemment, le *dropout* a été proposé comme une méthode alternative de régularisation. Cette méthode est motivée par la volonté d'empêcher les prédictions du réseau de reposer sur un sous-ensemble de paramètres spécifiques. Elle consiste à *éteindre*<sup>g</sup> aléatoirement une partie des neurones d'une couche cachée à chaque exemple d'apprentissage. Il en découle qu'au cours de l'entraînement, le réseau n'aura accès qu'à une partie des activations cachées. Seuls les neurones *allumés* seront modifiés par la SGD, ce qui favorise les configurations  $\theta$  distribuant leur calcul sur l'ensemble des neurones à disposition dans la couche cachée. La technique du *dropout* fait partie des éléments décisifs ayant conduit à des améliorations significatives des performances des réseaux de neurones, et tout particulièrement lorsque combiné avec des fonctions d'activation ReLU (KRIZHEVSKY, I. SUTSKEVER et HINTON 2012). La probabilité  $p$  qu'un neurone a de s'éteindre spontanément est un hyper-paramètre déterminant l'impact de cette régularisation sur l'entraînement.

## 1.4 Réseaux récurrents

Une part importante des problèmes naturels nécessitent de considérer des données d'entrée organisées en séquences. Cette architecture est particulièrement adaptée au traitement automatique de la langue. Dans un réseau récurrent ou *Recurrent Neural Network* (SCHMIDHUBER 1990), le contexte est représenté par des connexions récurrentes. Le nombre de mots du contexte pris en compte devient variable et l'ordre des entrées  $y$  est préservé. Les RNN compressent le contexte d'un exemple dans un espace de faible dimension. Ils ont la possibilité de gérer une forme de mémoire à court terme, ce qui leur permet de mieux appréhender les invariances de positions. Il s'agit des caractéristiques majeures permettant aux RNN de surpasser les réseaux *feed-forward* dans le cadre de la

---

f. Cette méthode sera appliquée afin de régulariser sur ses valeurs initiales la couche d'entrée initialisé à l'aide de *word embedding*.

g. forcer leur sortie à 0.

classification et l'étiquetage de séquences.

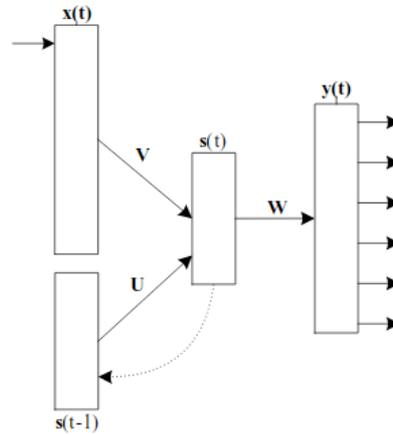


Figure 1.4 – *Recurrent Neural Network (RNN)* - Guo [2013](#)

### 1.4.1 Structure récurrente

On note  $\mathbf{x}^{i:j}$  la séquence de vecteurs  $\mathbf{x}^i, \dots, \mathbf{x}^j$ . Un RNN considère en entrée une séquence  $\mathbf{x}^{1:n}$  de vecteurs de *features* ainsi qu'un état initial noté  $\mathbf{h}^0$ . Il retourne une séquence d'états  $\mathbf{h}^{1:n}$  ainsi qu'une séquence de vecteurs de sortie  $\mathbf{y}^{1:n}$ . L'état  $\mathbf{h}^t$  est une combinaison de l'entrée  $\mathbf{x}^t$  et de l'état précédent  $\mathbf{h}^{t-1}$ . Cet état récurrent  $\mathbf{h}^{t-1}$  est interprété comme une représentation de taille fixe du contexte (ou l'historique) ayant donné lieu à  $\mathbf{x}^t$ .

Un réseau récurrent est composé d'une couche d'entrée dense, une couche cachée récurrente et une couche linéaire de sortie. La couche cachée représente le contexte de l'élément courant calculé récursivement par agrégation. Les vecteurs d'entrée de cette couche récurrente sont la concaténation (notée  $\bullet$ ) d'un vecteur  $\mathbf{x}^t$  issu de la couche d'entrée au temps  $t$  représentatif du mot courant, et des sorties de la couche cachée  $\mathbf{h}^{t-1}$  au pas de temps précédent.

$$\begin{aligned} \mathbf{h}^t &= \mathcal{F}_{\text{RNN}}(\mathbf{x}^t, \mathbf{h}^{t-1}) = \varphi(V \cdot \mathbf{x}^t + U \cdot \mathbf{h}^{t-1} + \mathbf{b}_h) \\ &= \varphi(W_h \cdot (\mathbf{x}^t \bullet \mathbf{h}^{t-1}) + \mathbf{b}_h) \end{aligned} \quad (1.23)$$

$$\hat{\mathbf{y}}^t = \mathcal{F}_{\text{OUTPUT}}(\mathbf{h}^t) = W_o \cdot \mathbf{h}^t + \mathbf{b}_o \quad (1.24)$$

## 1.4.2 Extensions

### 1.4.2.1 Neurones munis d'un état interne

Les réseaux de neurones particulièrement profonds souffrent d'un problème de *vanishing gradient* (Yoshua BENGIO, SIMARD et FRASCONI 1994 ; HOCHREITER 1998). Considérons un réseau *feed-forward* à  $N$  couches. Les gradients des fonctions d'activation standards sont généralement à valeur dans  $[-1, 1]$ . Lorsque l'algorithme de rétro-propagation propage l'erreur de classification dans les couches inférieures, le gradient effectivement propagé jusqu'à la première couche est atténué par un facteur  $\prod_{k=1}^N \alpha_k$  avec  $\forall k \alpha_k \in [-1, 1]$ . Lorsque  $N$  devient important, alors l'algorithme de rétro-propagation modifie de moins en moins les premières couches du réseau. Étant donné qu'on entraîne les réseaux récurrents en les ramenant à des réseaux *feed-forward* possédant autant de couches cachées que la séquence d'entrée compte d'éléments, cet effet indésirable rend difficile l'apprentissage sur des séquences de longueur importante.

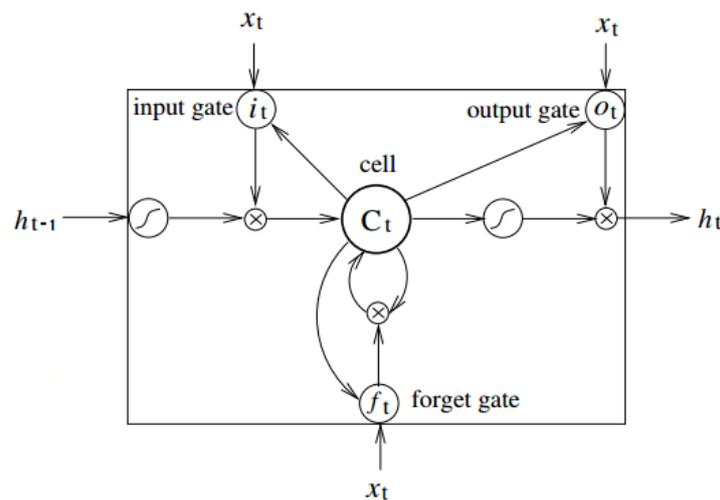


Figure 1.5 – *Long Short-Term Memory cell (LSTM)*

Les cellules *Long Short-Term Memory* (HOCHREITER et J. SCHMIDHUBER 1997) furent proposées afin de pallier cette difficulté et sont toujours utilisées de nos jours (SUNDERMEYER, NEY et SCHLUTER 2015). Elles utilisent un système de portes (*gates*) permettant au modèle de gérer explicitement la mise à jour et la remise à zéro d'une mémoire interne. De plus, ces cellules peuvent s'avérer plus efficaces pour extraire et exploiter les régularités et dépendances de longue portée observables au sein des données. Les portes sont trois vecteurs  $\mathbf{i}$ ,  $\mathbf{f}$ ,  $\mathbf{o}$  paramétrant les transformations de l'état interne  $\mathbf{c}$  de la cellule. Toutes ces quantités

correspondent à des vecteurs de la même taille que le vecteur  $\mathbf{h}$ . Les différentes *gates* sont des vecteurs à valeur dans  $[0; 1]$  pouvant être interprétées comme des facteurs d'atténuation sélectifs. Sachant l'entrée courante  $\mathbf{x}^t$ , l'historique  $\mathbf{h}^{t-1}$  et l'état  $\mathbf{c}^{t-1}$  précédents, l'*input gate*  $\mathbf{i}^t$  détermine quelle part des entrées courantes ( $\mathbf{x}^t, \mathbf{h}^{t-1}$ ) est discriminante pour la prédiction courante, la *forget gate* détermine à quel point l'état précédent de la cellule  $\mathbf{c}^{t-1}$  est toujours pertinent pour la prédiction courante, et l'*output gate* détermine quelle part de l'état courant  $\mathbf{c}^t$  contient la prédiction courante tout en étant *a priori* utile pour la prédiction suivante.

$$\begin{cases} \mathbf{i}_t = \sigma(W_{xi} \cdot \mathbf{x}^t + W_{hi} \cdot \mathbf{h}^{t-1} + W_{ci} \cdot \mathbf{c}^{t-1} + b_i) \\ \mathbf{f}_t = \sigma(W_{xf} \cdot \mathbf{x}^t + W_{hf} \cdot \mathbf{h}^{t-1} + W_{cf} \cdot \mathbf{c}^{t-1} + b_f) \\ \mathbf{o}_t = \sigma(W_{xo} \cdot \mathbf{x}^t + W_{ho} \cdot \mathbf{h}^{t-1} + W_{co} \cdot \mathbf{c}^{t-1} + b_o) \\ \mathbf{c}^t = \mathbf{f}_t \odot \mathbf{c}^{t-1} + \mathbf{i}_t \odot \tanh(W_{xc} \cdot \mathbf{x}^t + W_{hc} \cdot \mathbf{h}^{t-1} + b_c) \end{cases} \quad (1.25)$$

$$\mathbf{h}^t = \mathcal{F}_{\text{LSTM}}(\mathbf{x}^t, \mathbf{h}^{t-1}) = \mathbf{o}_t \odot \tanh(\mathbf{c}^t) \quad (1.26)$$

$$\hat{\mathbf{y}}^t = \mathcal{F}_{\text{OUTPUT}}(\mathbf{h}^t) = W_o \cdot \mathbf{h}^t \quad (1.27)$$

Dans l'équation 1.25,  $\sigma(\cdot)$  est une fonction sigmoïde logistique, ce qui permet aux *gates* d'être à valeur dans  $[0, 1]$ . Ces vecteurs sont appliqués terme à terme (notée  $\odot$ ) sur les quantités qu'ils régulent. Respectivement, l'*input gate* supervise les entrées courantes  $\mathbf{x}^t \bullet \mathbf{h}^{t-1}$ , la *forget gate* l'état interne  $\mathbf{c}^{t-1}$ , et l'*output gate* la sorties  $\mathbf{h}^t$ .

### 1.4.2.2 Architecture Bidirectionnelle

Dans le cadre de l'étiquetage de séquences, on a généralement accès à la totalité d'une séquence dont on souhaite annoter chaque constituant. Cela signifie qu'à tout instant  $t$ , on connaît les entrées passées ( $t - \delta$ ) et futures ( $t + \delta$ ). Un RNN classique ne considère cependant que les entrées passées. Si on souhaite accéder aussi bien aux *features* passées et futures pour un instant  $t$  donné, il est possible d'utiliser un réseau bidirectionnel (SCHUSTER et PALIWAL 1997).

Le principe fondamental des réseaux récurrents bidirectionnels est de présenter chaque séquence *de gauche à droite* et *de droite à gauche* à deux RNN distincts connectés à la même couche de sortie. Cela signifie qu'en tout point d'une séquence donnée, le BiRNN a accès à la totalité de l'information séquentielle en aval et en amont. Ce modèle est capable de capturer l'information contenue dans des données séquentielles et maintient des *features* du contexte passé et

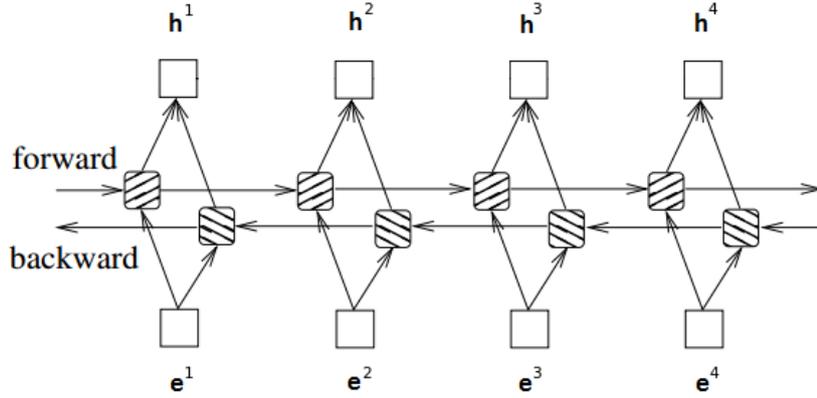


Figure 1.6 – *Bidirectional Recurrent Neural Network (BiRNN)*

futur (GRAVES, MOHAMED et G. E. HINTON 2013). De cette manière, on peut tirer parti des *features* passées (via la branche *forward*) et des *features* futures (via la branche *backward*) pour un instant donné de la séquence d'entrée. De plus, le réseau étant libre d'utiliser autant de ce contexte que nécessaire, la taille de la fenêtre temporelle à considérer n'est plus fixée *a priori*. Ce réseau prend en entrée une séquence  $\mathbf{x}^{1:T}$  dont il projette chaque élément dans un espace de faible dimension. Les activations de la couche de sortie  $\mathbf{y}^t$  représentent, dans le cas d'un problème de classification, la distribution de probabilités des différentes classes prédites au temps  $t$ .

$$\vec{\mathbf{h}}^t = \vec{\mathcal{F}}_{\text{RNN}}(\mathbf{x}^t, \vec{\mathbf{h}}^{t-1}) = \varphi(\vec{W}_h \cdot (\mathbf{x}^t \bullet \vec{\mathbf{h}}^{t-1})) \quad (1.28)$$

$$\overleftarrow{\mathbf{h}}^t = \overleftarrow{\mathcal{F}}_{\text{RNN}}(\mathbf{x}^t, \overleftarrow{\mathbf{h}}^{t+1}) = \varphi(\overleftarrow{W}_h \cdot (\mathbf{x}^t \bullet \overleftarrow{\mathbf{h}}^{t+1})) \quad (1.29)$$

$$\mathbf{h}^t = \mathcal{F}_{\text{BIRNN}}(\vec{\mathbf{h}}^t, \overleftarrow{\mathbf{h}}^t) = \varphi(W_h \cdot (\vec{\mathbf{h}}^t \bullet \overleftarrow{\mathbf{h}}^t)) \quad (1.30)$$

$$\hat{\mathbf{y}}^t = \mathcal{F}_{\text{OUTPUT}}(\mathbf{h}^t) = W_o \cdot \mathbf{h}^t \quad (1.31)$$

### 1.4.2.3 Modèle d'Attention

Le RNN classique ne peut considérer que l'état courant de l'historique  $\mathbf{h}^t$  pour réaliser une prédiction  $\mathbf{y}^t$ . On souhaiterait permettre au réseau d'accéder à la totalité de l'information séquentielle *i.e.* à l'ensemble des états pris par l'historique  $\mathbf{h}^{1:t}$ . On le surmonte d'un mécanisme d'attention (J. K. CHOROWSKI, BAHDANAU,

SERDYUK et al. 2015). Les entrées de la couche de sortie deviennent une combinaison linéaire des états cachés et non plus l'état caché courant. Les coefficients  $a_i^t$  sont relatifs à l'importance de chaque état caché  $i$  dans la prise de décision à un instant  $t$  donné. Par exemple, si  $a_2^3$  est grand, alors la couche de sortie porte davantage son attention sur  $\mathbf{h}^2$  lorsqu'elle considère la prédiction  $\mathbf{y}^3$ .

$$u_i^t = \mathbf{v}_A \cdot \sigma \left( W_{\mathcal{A}_1} \cdot \mathbf{h}^t + W_{\mathcal{A}_2} \cdot \mathbf{h}^i \right), \quad u \in \mathbb{R}^{T \times T} \quad (1.32)$$

$$a^t = \underset{i=1..T}{softmax} \left( u^t \right) \quad (1.33)$$

$$\mathbf{s}^t = \sum_{i=1}^T a_i^t \mathbf{h}^i \quad (1.34)$$

$$\hat{\mathbf{y}}^t = \mathcal{F}_{\text{OUTPUT}} \left( \mathbf{s}^t \right) = W_o \cdot \mathbf{s}^t \quad (1.35)$$

avec  $W_{\mathcal{A}_1}, W_{\mathcal{A}_2} \in \mathbb{R}^{A \times 2h}$  et  $\mathbf{v}_A \in \mathbb{R}^A$  alors comme désiré  $u_i^t$  est un scalaire.  $A$  est un hyper-paramètre caractérisant la dimension de l'état caché du modèle d'attention. Le mécanisme d'attention donne accès la succession d'états cachés  $\mathbf{h}^{t \pm \tau}$  au cours de la séquence. Il a pour but de déterminer la pertinence des informations contenues dans cette séquence d'*historiques*, compte-tenu de l'entrée et des *historiques droits* et *gauche* courants. Le réseau réalise une vue de la séquence d'états cachés, et estime des affinités entre les états afin de maximiser la justesse de ses prédictions. Lors du calcul de ces affinités (cf. équation 1.32), l'indice  $\mathcal{A}_1$  symbolise l'ensemble de paramètres dédiés à la prise en compte de l'état courant (temps  $t$ ) tandis que  $\mathcal{A}_2$  symbolise l'ensemble de paramètres dédiés au passage en revue de l'historique (temps  $0..i..T$ ). Il réalise la combinaison linéaire des états cachés, pondérée par leur affinités avec l'état courant. Cela signifie que l'accès aux états cachés est *soft*. Cela permet notamment de pouvoir entraîner ce type de réseau avec un algorithme d'optimisation par rétro-propagation du gradient qui repose sur la dérivabilité des fonctions sur lesquelles on l'applique.

### 1.4.3 Algorithme d'apprentissage

L'architecture récurrente décrite dans la section précédente est capable de prendre en entrée une séquence  $x$  de taille variable afin de prédire une séquence de sortie  $y$  de même taille. Ces couples de séquences forment nos exemples d'ap-

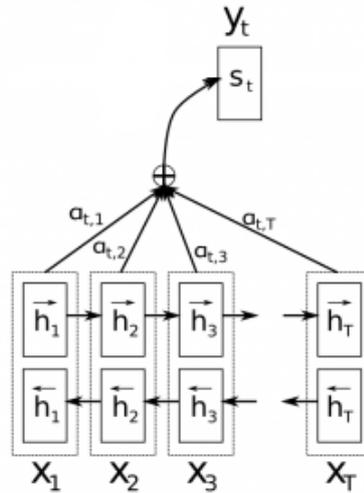


Figure 1.7 – *Modèle d'attention*

prentissage. Comme pour le Perceptron multi-couches, l'entraînement du réseau est réalisé par présentation successive d'exemples d'apprentissage. Cette procédure comporte deux phases. Premièrement, l'algorithme de *Back Propagation Through Time* sélectionne aléatoirement un exemple d'apprentissage  $(x, y)$  puis déploie la structure récurrente  $\theta$  sur le nombre de pas temporel  $\tau$  correspondant à la taille de la séquence d'entrée. La séquence d'entrée  $x$  est injectée dans le réseau. Les signaux d'entrée sont propagés dans les différentes couches *via* l'historique  $\mathbf{h}$  jusqu'à produire une séquence de sortie  $\hat{y} = \mathcal{F}_\theta(x)$ . C'est la phase de prédiction par propagation des entrées. Puis dans un second temps, il estime l'erreur  $\mathcal{L}(\hat{y}, y)$  qu'il commet vis-à-vis de la séquence de sortie attendue  $y$ . Le gradient de l'erreur en fonction de chaque paramètre interne  $\hat{\mathbf{g}} = \frac{\partial \mathcal{L}}{\partial \theta}$  est rétro-propagé. L'ensemble des paramètres formant la structure récurrente  $\theta$  est optimisé en vue de minimiser cette erreur. C'est la phase de correction par rétro-propagation du gradient de l'erreur. Ces étapes sont indépendantes du format d'un exemple d'apprentissage *i.e.* du découpage en sous-séquences des *observations*.

### 1.4.3.1 Back Propagation Through Time

L'entraînement de structures récurrentes nécessite de propager le gradient de l'erreur commise à un temps  $t$  dans les couches précédentes ( $t - 1, t - 2, \text{etc.}$ ) ayant agrégé l'information séquentielle  $\mathbf{h}^t$  utilisée lors de la prédiction  $\mathbf{y}^t$ . Cette propagation de l'erreur est réalisée à l'aide de l'algorithme de *BackPropagation Through Time* (GUO 2013). Le réseau est *déplié* sur un nombre de pas temporel fixé arbitrairement ( $\tau$ ) afin de s'abstraire temporairement de la structure ré-

currente. Les poids récurrents sont dupliqués et partagés pour construire une instance du réseau similaire à la Figure 3.4. Entraîner un réseau récurrent se ramène ainsi en partie à entraîner un réseau de neurones *feed-forward* possédant un très grand nombre de poids partagés.

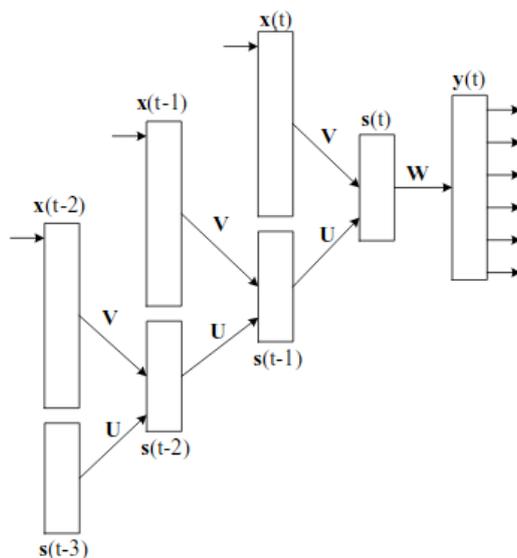


Figure 1.8 – *Unfolded Recurrent Neural Network (RNN)* - Guo 2013

On calcule récursivement le gradient de l'erreur  $\mathcal{L}(\mathcal{F}_\theta(\mathbf{x})^t, \mathbf{y}^t)$  vis à vis des paramètres de chaque couche récurrente ( $\theta^t, \theta^{t-1}, \dots, \theta^{t-\tau}$ ) comme s'il s'agissait de paramètres indépendants *i.e* sans considérer qu'il s'agit d'une même matrice de poids, dupliquée. On note  $\mathbf{g}_{t-i}^t$  le gradient de l'erreur commise au temps  $t$  vis à vis des paramètres  $\theta^{t-i}$ . On calcule donc indépendamment pour chaque instance de la couche récurrente la modification à apporter et on applique la somme de ces changements à  $\theta$  afin de conserver la consistance de la structure récurrente.

$$\forall i \in [0, \tau[ \quad \mathbf{g}_{t-i}^t = \frac{\partial \mathcal{L}(\mathcal{F}_\theta(\mathbf{x})^t, \mathbf{y}^t)}{\partial \theta^{t-i}} \quad (1.36)$$

$$\theta \leftarrow \theta + \varepsilon \frac{1}{\tau} \sum_{i=0}^{\tau-1} \mathbf{g}_{t-i}^t \quad (1.37)$$

En pratique, on fixe la taille  $T$  du réseau *déplié* et on s'intéresse à toute la séquence de prédictions, pas seulement  $\mathbf{y}^t$  comme précédemment. On considère ainsi l'erreur commise sur une séquence de prédiction  $\mathbf{y}^{1:T} = \mathbf{y}^1, \dots, \mathbf{y}^T$ . On crée

donc une instance du réseau déplié à  $T$  couches récurrentes  $(\theta^1, \dots, \theta^T)$  et on se munit d'une fonction de coût séquentielle  $\mathcal{L}(\mathcal{F}_\theta(\mathbf{x})^{1:T}, \mathbf{y}^{1:T})$  qui n'est autre que la moyenne des fonctions de coût indépendantes.

---

**Algorithm 5** BackPropagation Through Time

---

**Require:** Function  $\mathcal{F}_\theta(\mathbf{x})$  parameterized with shared parameters  $\theta^1, \dots, \theta^T$

**Require:** Training set of sequences input  $\mathbf{x}_1, \dots, \mathbf{x}_n$  and output  $\mathbf{y}_1, \dots, \mathbf{y}_n$

**Require:** Loss function  $\mathcal{L}$

```

while stopping criteria not met do
  Sample a training sequence  $\mathbf{x}_k, \mathbf{y}_k$ 
   $\hat{\mathbf{g}} \leftarrow 0$ 
   $\vec{\mathbf{h}}^0 \leftarrow 0$ 
  for  $t = 1$  to  $T$  do
    Compute the loss  $\mathcal{L}(\mathcal{F}_\theta(\mathbf{x}_k)^t, \mathbf{y}_k^t)$ 
    for  $i = 0$  to  $t$  do
       $\hat{\mathbf{g}} \leftarrow \hat{\mathbf{g}} + \frac{1}{T} \frac{1}{t} \frac{\partial \mathcal{L}(\mathcal{F}_\theta(\mathbf{x}_k)^t, \mathbf{y}_k^t)}{\partial \theta^{t-i}}$ 
    end for
  end for
   $\theta \leftarrow \theta + \varepsilon \hat{\mathbf{g}}$ 
end while
return  $\theta$ 

```

---

$$\mathcal{L}(\mathcal{F}_\theta(\mathbf{x})^{1:T}, \mathbf{y}^{1:T}) = \frac{1}{T} \sum_{t=1}^T \mathcal{L}(\mathcal{F}_\theta(\mathbf{x})^t, \mathbf{y}^t) \quad (1.38)$$

$$\forall i \in [0, t[ \quad \mathbf{g}_{t-i}^t = \frac{\partial \mathcal{L}(\mathcal{F}_\theta(\mathbf{x})^t, \mathbf{y}^t)}{\partial \theta^{t-i}} \quad (1.39)$$

$$\theta \leftarrow \theta + \varepsilon \frac{1}{T} \sum_{t=1}^T \frac{1}{t} \sum_{i=0}^{t-1} \mathbf{g}_{t-i}^t \quad (1.40)$$

Cependant, dans le présent algorithme l'erreur est calculée pour une seule séquence d'exemples. Pour les mêmes raisons que dans le cas d'un réseau *feed-forward*, on peut être amené à considérer la somme des erreurs sur un *minibatch* de  $m$  séquences.

---

**Algorithm 6** Minibatch BackPropagation Through Time

---

**Require:** Function  $\mathcal{F}_\theta(\mathbf{x})$  parameterized with shared parameters  $\theta^1, \dots, \theta^T$

**Require:** Training set of sequences input  $\mathbf{x}_1, \dots, \mathbf{x}_n$  and output  $\mathbf{y}_1, \dots, \mathbf{y}_n$

**Require:** Loss function  $\mathcal{L}$

**while** stopping criteria not met **do**

    Sample a minibatch of  $m$  examples  $\{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_m, \mathbf{y}_m)\}$

$\hat{\mathbf{g}} \leftarrow 0$

$\vec{\mathbf{h}}^0 \leftarrow 0$

**for**  $k = 1$  to  $m$  **do**

**for**  $t = 1$  to  $T$  **do**

            Compute the loss  $\mathcal{L}(\mathcal{F}_\theta(\mathbf{x}_k)^t, \mathbf{y}_k^t)$

**for**  $i = 0$  to  $t$  **do**

$\hat{\mathbf{g}} \leftarrow \hat{\mathbf{g}} + \frac{1}{m} \frac{1}{T} \frac{1}{t} \frac{\partial \mathcal{L}(\mathcal{F}_\theta(\mathbf{x}_k)^t, \mathbf{y}_k^t)}{\partial \theta^{t-i}}$

**end for**

**end for**

**end for**

$\theta \leftarrow \theta + \varepsilon \hat{\mathbf{g}}$

**end while**

**return**  $\theta$

---

### 1.4.4 Mise en forme des exemples

On considère, dans cette sous-section, l'application de réseaux récurrents à l'apprentissage de données textuelles. Une architecture récurrente est capable de prendre en entrée une séquence de mots  $x$  de taille variable afin de prédire une séquence de sortie  $y$  de même taille. Ces couples de séquences forment nos exemples d'apprentissage. L'entraînement du réseau est réalisé par présentation successive d'exemples d'apprentissage. Cette procédure comporte deux phases. La phase de prédiction (propagation) et la phase de correction (rétropropagation). Ces étapes sont indépendantes du format d'un exemple d'apprentissage *i.e.* du découpage en sous-séquences du corpus. Il est donc désormais nécessaire de déterminer la forme des données d'apprentissage lorsqu'elles seront présentées au réseau. Nous verrons qu'outre l'aspect technique, ces choix détermineront le comportement attendu du réseau, une fois l'entraînement mené à son terme.

### 1.4.4.1 Apprentissage phrase-par-phrase

L'approche standard consiste à considérer le corpus comme une collection de phrases  $C_{\text{task}} = \{x_1, \dots, x_N\}$  où  $x_n = (x^t)_{t=1..|x_n|}$ . À chaque itération, l'algorithme de *Back Propagation Through Time* choisit aléatoirement une phrase  $x_n$  de  $C_{\text{task}}$ . Un réseau de même taille  $|x_n|$  que la phrase compte de mots est instancié. L'algorithme du BiRNN utilise deux RNN qui initialisent leurs historiques respectifs à zéro. Autrement dit, les historiques initiaux gauche  $\vec{\mathbf{h}}^0$  et droit  $\overleftarrow{\mathbf{h}}^{|x_n|+1}$  sont fixés à valeur nulle. Les paramètres internes du réseau  $\theta^{1..|x_n|}$  sont mis à jour de sorte à minimiser l'erreur commise sur la séquence de classes prédites  $\hat{y}^{1..|x_n|}$  vis-à-vis de la séquence attendue  $y^{1..|x_n|}$  au terme des phases de prédiction et de correction.

La minimisation d'une erreur commise à un instant donné de la séquence prédite se répercute sur l'ensemble des paramètres internes à l'origine des représentations ayant conduit à cette erreur. La structure récurrente  $\theta$  apprend de cette manière à produire pour chaque mot d'entrée  $x^t$  une représentation  $\mathbf{h}^t$  minimisant l'erreur de classification sur ce mot. De plus,  $\mathbf{h}^t$  est également optimisé par le biais des branches *forward* et *backwards* pour servir de représentation du contexte immédiat des mots adjacents, et par extension *via* le modèle d'attention, comme représentation composant le contexte global de tout autre mot de la phrase. Au terme de son entraînement et indépendamment de ses performances, le réseau se comporte comme un analyseur avec un fort *a priori* sur l'indépendance des phrases pour la tâche considérée. Découper les exemples en phrases revient à poser comme hypothèse que seule l'information contenue dans la phrase est pertinente pour réaliser l'analyse désirée.

Cette hypothèse s'est révélée fondée dans la plupart des tâches d'analyse du TAL. C'est par exemple le cas de l'analyse morpho-syntaxique. Seul le contexte local est déterminant dans la détection de parties de discours. C'est d'ailleurs la raison pour laquelle l'ensemble des analyseurs morpho-syntaxiques considèrent des *N-gram* de mots comme seule représentation du contexte d'une occurrence à étiqueter. Cependant, dans le cadre d'une architecture unifiée capable d'être entraînée sur un ensemble des tâches d'analyse de séquences, on souhaiterait faire le moins d'hypothèses possibles sur la tâche considérée. On désire conserver l'aspect structurant des phrases d'un corpus sans pour autant les traiter indépendamment les unes des autres.

## Conclusion

Ce chapitre avait pour but de poser les bases du formalisme qui sera utilisé dans la suite de ce document. On y a défini un neurone formel ou *Perceptron* (*section 1.2*), un séparateur linéaire fondamental en apprentissage automatique. Cette définition a par la suite été étendue afin de construire un Perceptron multicouches (*section 1.3*), un réseau sans rétro-action composé de couches denses de neurones. Ces réseaux ont la propriété d'être des approximateurs universels utilisés pour résoudre des problèmes de classification. Le chapitre se clôture par la présentation des réseaux de neurones récurrents (*section 1.4*) et plus particulièrement sur les LSTM-BiRNN car ils offrent d'intéressantes propriétés vis à vis du traitement automatique de la langue.

Nous étudierons par la suite l'application de ces réseaux de neurones profonds au Traitement Automatique de la Langue. L'ensemble des algorithmes décrits précédemment seront ainsi utilisés afin de réaliser un modèle d'analyse syntaxique et sémantique de données textuelles appliqué à la compréhension automatique de la langue orale. Les systèmes traditionnels du TAL considèrent une analyse comme une hiérarchie de sous-problèmes. L'analyse est réalisée par des *pipelines* de modèles spécifiques à chaque sous-problème. Ces modèles sont généralement basés sur des heuristiques d'apprentissage automatique supervisé (CRF, SVM) s'appuyant sur une sélection de *features* réalisée par un expert. Nous désirons profiter de la flexibilité architecturale des réseaux de neurones afin de les adapter aux problématiques spécifiques au traitement automatique de la langue.

En parallèle de l'architecture du modèle, il convient d'apporter un soin particulier à la mise en forme de ses entrées. En effet, la représentation des mots, entités discrètes par essence, dans des modèles du *Machine Learning*, domaine des valeurs continues, est une étape délicate, et des algorithmes d'apprentissage de représentations vectorielles de mots (*word embeddings*) ont été proposés. Ces travaux se sont appuyés sur l'hypothèse distributionnelle qui stipule que des mots apparaissant dans des contextes similaires ont des sens ou des usages similaires. Afin de minimiser la dépendance du modèle à l'expertise linguistique, nous nous intéresserons aux techniques d'apprentissage *End-to-End* utilisant ces *word embeddings*. Cela signifie qu'on souhaite uniquement considérer ces représentations vectorielles des mots en entrée du modèle, et donc recourir au minimum de pré-traitements.

# 2 Application au Traitement Automatique de la Langue

Un des objectifs de cette thèse est de réaliser un analyseur sémantique reposant sur un système global d'analyse syntaxique. Ce chapitre sera l'occasion de présenter un sous-ensemble représentatif de ces tâches d'analyse fréquemment étudiées ainsi que les analyseurs traditionnellement employés. On s'intéressera successivement dans la section 2.2 aux tâches d'étiquetage morpho-syntaxique (*Tagging*), d'analyse syntaxique de surface (*Chunking*) et enfin d'analyse en dépendance syntaxique (*Parsing*). Nous montrerons dans la section 2.3 qu'au prix d'hypothèses simplificatrices, l'ensemble de ces tâches peuvent se décomposer en combinaisons d'opération d'étiquetage, de segmentation et de détection de relation. Nous formaliserons ces trois opérations sous forme de problèmes de classification afin d'y appliquer les méthodes exploitant la théorie de l'apprentissage automatique de réseaux de neurones profonds, présentées au cours du chapitre précédent. Enfin, nous nous étendrons sur la nécessité de produire des représentations vectorielles efficaces pour les mots. La section 2.4 est consacrée à la présentation des *word embeddings* ainsi que des méthodes usuelles de calcul de ces vecteurs. On souhaite mettre au point des réseaux de neurones capables de tirer parti de ces représentations génériques faiblement supervisées tout en les adaptant à la tâche et au corpus considérés.

## 2.1 Contexte et Motivations

Le Traitement Automatique de la Langue (TAL) concerne l'application de programmes et techniques informatiques à tous les aspects du langage humain. Ce domaine vise à produire par exemple des systèmes de reconnaissance et de synthèse de texte appliqués à la correction, à la traduction, au résumé automatique ou encore à l'analyse d'opinion. L'ensemble de ces systèmes repose en grande partie sur des modèles statistiques qui ne peuvent fonctionner de manière efficace qu'en s'appuyant sur des représentations des informations jugées pertinentes à la réalisation de leur tâche d'intérêt. Autrement dit, ces modèles reposent en partie sur des analyses préliminaires, produisant des traits ou *features* représentatifs des structures lexicales, syntaxiques et sémantiques présentes dans les énoncés considérés. Les recherches menées dans ce domaine ont identifié plusieurs niveaux linguistique rassemblant les tâches d'analyse permettant de produire automatiquement ces annotations.

Le domaine du TAL est soumis à un certain nombre de contraintes immanentes. Bien que des données soient disponibles en quantité importante, elles se composent d'une myriade de corpus de différents horizons, annotés de façon très hétérogène. Se côtoient donc un grand nombre de ressources, traitant de domaines divers avec leur vocabulaire propre, exprimées dans différents niveaux de langue. Le panorama du TAL est dominé par deux familles d'approche : dans la première, la connaissance linguistique s'exprime sous forme de règles (grammaticales pour le traitement syntaxique, d'inférence pour le traitement sémantique, etc.), et de symboles sur lesquels ces règles opèrent. La seconde repose sur l'hypothèse de l'existence d'un modèle probabiliste sous-jacent aux données, modèle dont les paramètres s'infèrent à partir de corpus de données linguistiques annotés.

### 2.1.1 Analyse sémantique

L'analyse sémantique consiste à représenter sous forme abstraite les concepts et les relations sémantiques présents dans une phrase. Dans le contexte du projet européen SENSEI dans lequel se déroule cette thèse, on s'intéresse en particulier au formalisme des cadres sémantiques tels que définis dans le projet Berkeley FrameNet. Chacun de ces cadres est déclenché par un mot de la phrase (*trigger*) et contient tous les constituants ayant un intérêt dans ce concept (*frame elements*). L'ensemble des modèles de détection de cadres sémantiques ne peuvent fonctionner de manière efficace qu'en s'appuyant sur un système de *features*, modélisant les informations jugées pertinentes pour prédire au mieux les structures sémantiques. En particulier, connaître la structure syntaxique d'un énoncé permet d'explicitier les relations de dépendance<sup>a</sup> entre les différents constituants, afin de construire une représentation du sens de cet énoncé. Et donc, même si la représentation sémantique en cadres s'abstrait de la structure syntaxique de la phrase, l'analyse sémantique prédisant cette structure requiert des informations syntaxiques issues d'une analyse préalable.

Les recherches menées sur ce sujet ont identifiés plusieurs tâches d'analyse syntaxique préliminaires permettant de produire les annotations requises à l'analyse sémantique. Cette analyse syntaxique fait elle-même habituellement suite à une analyse lexicale qui découpe le texte en un flux de mots (*tokens*). Elle se compose d'un ensemble de tâches incontournables de reconnaissance, généralement réalisées en cascade. Dans la suite, nous avons restreint notre étude à un sous-ensemble de ces tâches d'analyses primordiales. L'étiquetage morpho-syntaxique

---

a. Par exemple, entre sujet et objet.

détermine la classe grammaticale des mots d'une phrase considérée. L'analyse syntaxique de surface identifie les éléments constituant cette phrase tels que les groupes nominaux ou verbaux. L'analyse en dépendances syntaxiques détecte les relations syntaxiques existant entre les syntagmes identifiés précédemment.

### 2.1.2 État de l'art

Généralement, on a accès à un corpus de petite taille, spécifique à un domaine et annoté manuellement, qu'on utilise pour calibrer une succession de modèles (*pipelines*) afin de produire un système d'annotation syntaxique automatique (e.g. *MACAON* : NASR et BÉCHET 2011 ; *The Stanford Parser* : MANNING 2014). Ces *pipelines* sont composés de modèles hétérogènes. Les analyseurs syntaxiques à base de grammaire explicite (grammaires en constituants, en dépendances) expriment la connaissance linguistique sous forme de règles (grammaticales pour le traitement syntaxique, d'inférence pour le traitement sémantique, etc.), et de symboles sur lesquelles ces règles opèrent. Les modèles statistiques regroupent un ensemble très vaste de modèles (HMM, CRF, SVM, DNN) qui reposent sur l'hypothèse d'un modèle probabiliste sous-jacent aux données, modèle dont les paramètres s'infèrent automatiquement à partir de corpus de données linguistiques annotées. Ces modèles statistiques maximisent la vraisemblance des prédictions vis-à-vis des données d'apprentissage observées, afin de simuler au mieux le modèle naturel. Ces modèles optimisés indépendamment sur chaque tâche d'intérêt sont ainsi disposés en cascade. Les sorties des uns servant d'entrée aux autres, chaque modèle n'a accès qu'aux prédictions des modèles l'ayant précédé. Cela permet d'accumuler les annotations à mesure qu'on monte dans les niveaux linguistiques au prix toutefois d'un phénomène de cumul des erreurs.

De nombreux travaux ont étudié l'apprentissage de réseaux de neurones profonds (DNN) à partir de données textuelles (Modèle de langage : Yoshua BENGIO, DUCHARME, VINCENT et al. 2003) afin de l'appliquer, avec succès, à la résolution de diverses tâches du TAL au sein de ces *pipelines*. Ce type de modèle peut être utilisé sur différentes tâches du traitement des langues naturelles, en obtenant des performances similaires aux méthodes "état de l'art" (COLLOBERT, WESTON, BOTTOU et al. 2011). Les méthodes de pré-entraînement et d'apprentissage non-supervisé de représentation vectorielle de mot se sont raffinées, passant d'initialisations des couches d'entrée à l'aide d'*auto-encoder* à l'utilisation d'algorithmes basés sur l'analyse linguistique distributionnelle (Word2Vec : MIKOLOV, I. SUTSKEVER, K. CHEN et al. 2013). Ainsi, l'utilisation de modèles basés sur la notion de *word embeddings* s'est développée, combinés à des réseaux récurrents (Z. HUANG, XU et K. YU 2015) ou récursifs (SOCHER, PERELYGIN, J. WU et al. 2013), surmontés d'un modèle d'attention (BAHDANAU, J. CHOROWSKI, SER-

DYUK et al. 2015) ou à mémoire (WESTON, CHOPRA et BORDES 2014) pour une grande variété de tâches. Depuis que l'utilisation dans les réseaux récurrents de *gated units* telles que les LSTM s'est généralisée (SUNDERMEYER, NEY et SCHLUTER 2015), les RNN se sont montrés particulièrement efficaces dans la résolution de tâches complexes du TAL (Système de dialogue : SERBAN, SORDONI, Yoshua BENGIO et al. 2015 ; Traduction automatique : DONG, H. WU, HE et al. 2015).

## 2.2 Analyse syntaxique du texte

Cette section est dédiée à la présentation des tâches d'analyse syntaxique qui seront considérées dans toute cette étude. Seront décrites successivement les tâches d'étiquetage morpho-syntaxique (*Tagging*), d'analyse syntaxique de surface (*Chunking*) et d'analyse en dépendance syntaxique (*Parsing*). Chacune de ces tâches est associée à un corpus de référence permettant l'évaluation des différentes méthodes d'analyse présentées.

### 2.2.1 Étiquetage morpho-syntaxique

L'étiquetage morpho-syntaxique concerne l'analyse des classes grammaticales des mots composant une phrase. Aussi appelées parties de discours, les classes grammaticales regroupent les mots qui composent le discours selon les caractéristiques morphologiques et syntaxiques qu'ils ont en commun. On distingue traditionnellement les noms, les déterminants, les adjectifs, les pronoms, les verbes, les adverbes, les prépositions et autres conjonctions. Chaque classe est représentative de la fonction et du type d'information que les mots qui la composent peuvent avoir dans une phrase. Dans un énoncé considéré, chaque mot est associé à une et une seule classe grammaticale. Par exemple,

Le<sub>DET</sub> petit<sub>NC</sub> lit<sub>VRB</sub> du<sub>PREP</sub> Rousseau<sub>NP</sub> .

signifie que "Le" a été identifié comme un déterminant (DET), "petit" comme un nom (NC), "lit" comme un verbe (VRB), *etc.* L'étiquetage morpho-syntaxique est un problème relativement simple car, dans la plupart des cas, toutes les occurrences d'un mot appartiennent à la même classe. Cependant, certains mots, tel que *tout*, appartiennent à deux classes grammaticales, en l'occurrence il peut s'agir d'un déterminant comme d'un pronom. De plus, les homonymes, tel que *lit*,

renforcent les ambiguïtés auxquelles les analyseurs seront parfois confrontés. Les parties de discours forment les fondements de la syntaxe et toute tâche d'analyse de plus haut niveau requiert les annotations issues de l'analyse morpho-syntaxique. Il est donc primordial de parvenir à réaliser automatiquement une annotation de qualité, car toute erreur qui s'y glisserait se propagerait fatalement et serait responsable d'erreurs dans tout autre analyse suivante.

Un protocole d'évaluation des analyseurs morpho-syntaxiques est décrit dans TOUTANOVA, C. MANNING, KLEIN et al. 2003. Il se base sur un corpus de référence : le Penn TreeBank. Ce corpus journalistique est composé d'environ un million de mots annotés en partie de discours. Les sections 0 à 18 sont utilisées comme base d'apprentissage, tandis que les sections 19 à 21 sont dédiées à la l'ensemble de validation. Les sections 22 à 24 sont ainsi réservées à l'évaluation du modèle considéré. Cette évaluation,, systématique et comparable, de différents analyseurs a permis de déterminer que les meilleurs *classifiers* pour l'étiquetage morpho-syntaxique sont entraînés sur une fenêtre glissante de mots, qui est par la suite injectée dans un algorithme de décodage bidirectionnel, réalisant l'étape d'inférence. Les *features* communément utilisées sont composées d'hypothèses d'étiquettes grammaticales suivant et précédant le mot en cours d'analyse, du contexte de l'occurrence considérée qu'on représente généralement par un *N-gram* de mots, ainsi que des *features* empiriques<sup>b</sup> dédiées à la gestion des mots hors vocabulaire.

Model	PWA POS
C. MANNING 2011	97.32
COLLOBERT, WESTON, BOTTOU et al. 2011	97.37
SUN 2014	97.33
Z. HUANG, XU et K. YU 2015 (Bi-LSTM)	97.40
Z. HUANG, XU et K. YU 2015 (Bi-LSTM-CRF)	97.55

Table 2.1 – *État de l'art - étiquetage morpho-syntaxique.*

TOUTANOVA, C. MANNING, KLEIN et al. 2003 considèrent une classification par maximum d'entropie. Cette approche sera par la suite enrichie dans C. MANNING 2011. Une classification réalisée par un *Support Vector Machine* a été proposé par GIMÉNEZ et MÁRQUEZ 2004, l'inférence bidirectionnelle étant laissée à deux décodeurs basés sur l'algorithme de Viterbi. Plus récemment, COLLOBERT, WESTON, BOTTOU et al. 2011 et SUN 2014 ont proposé respectivement un *Multi-layer Perceptron* et un *Conditional Random Field* égalant tous deux les performances précédemment obtenues tandis que Z. HUANG, XU et K. YU 2015 repoussent les

b. Par exemple, des *features* morphologiques

limites de l'état de l'art à l'aide d'un modèle hybride *Bidirectional LSTM-CRF*, dans lequel les probabilités d'émission sont prédites par un réseau récurrent entraîné conjointement avec un *Conditionnal Random Field* chargé de l'optimisation globale de la séquence de prédiction. Nous nous intéressons en particulier à l'approche de COLLOBERT, WESTON, BOTTOU et al. 2011 qui appliquent avec succès une même architecture neuronale générique à différentes tâches du traitement de la langue.

## 2.2.2 Analyse syntaxique de surface

L'analyse syntaxique de surface segmente une phrase afin d'extraire des constituants syntaxiques. Les syntagmes généralement considérés sont les groupes nominaux et verbaux. Formellement, étant donné les formes et parties de discours des mots, on cherche à retrouver les étiquettes segmentales des *chunks* syntaxiques. Afin d'obtenir une annotation mot-à-mot, ces étiquettes peuvent être codées au format BIO : chaque mot est associé à un *label* précisant s'il marque le début (e.g., B-NP) ou l'intérieur (e.g., I-NP) d'un constituant considéré. Un mot extérieur à tout syntagme sera étiqueté O. Par exemple,

Le<sub>B-NP</sub> petit<sub>I-NP</sub> lit<sub>B-VP</sub> du<sub>I-VP</sub> Rousseau<sub>I-VP</sub> .

signifie que "Le petit" a été identifié comme un groupe nominal (NP) et "lit du Rousseau" comme un groupe verbal (VP). Toutefois, cette analyse ne produit qu'un étiquetage en syntagmes de la phrase et non un arbre syntaxique complet. Cela signifie qu'elle ne tire pas les liens existant entre les différents syntagmes d'une phrase.

Model	F1 Chunk
SHA et F. PEREIRA 2003	94.38
SHEN et SARKAR 2005	95.23
COLLOBERT, WESTON, BOTTOU et al. 2011	94.10
Z. HUANG, XU et K. YU 2015 (Bi-LSTM)	93.92
Z. HUANG, XU et K. YU 2015 (Bi-LSTM-CRF)	94.46

Table 2.2 – *État de l'art - analyse syntaxique de surface.*

Les méthodes d'analyse syntaxique de surface sont évaluées sur le corpus de la tâche partagée CoNLL 2000. Ce corpus est composé de 200 000 mots annotés en partie de discours (nom, verbe, *etc.*) et en constituants (groupes nominaux,

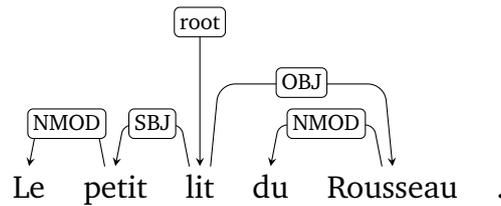
verbaux, etc.). Les sections 15 à 18 du Wall Street Journal constituent l'ensemble d'apprentissage tandis que la section 20 est réservée à l'évaluation. KUDOH et MATSUMOTO 2000 ont remporté le *CoNLL 2000 Challenge* avec un système basé sur des *Support Vector Machines* (SVMs). Chaque SVM réalisant une tâche de classification binaire à partir d'une fenêtre glissante centrée sur le mot d'intérêt, composée des mots associés à leur partie de discours respective, ainsi que les étiquettes de surface environnantes. Depuis, un certain nombre de systèmes basés sur des *Conditional Random Field* ont été proposés (SHA et F. PEREIRA 2003). Par la suite, SHEN et SARKAR 2005 considèrent une combinaison de *classifiers*, chacun entraîné sur un encodage différent de la segmentation (IOB, IOE, etc). Ils utilisent une annotation morpho-syntaxique provenant d'un étiqueteur externe et tirent parti de *features* issues de l'empirisme linguistique en concaténant notamment certains mots, bien choisis, à leur partie de discours. Ils construisent enfin des trigrammes de telles *features*, utilisés par un algorithme de Viterbi en inférence. Plus récemment, à l'aide de modèles basés sur l'apprentissage de réseaux de neurones, COLLOBERT, WESTON, BOTTOU et al. 2011 puis Z. HUANG, XU et K. YU 2015 ont tous deux obtenu des performances égalant celles précédemment présentées mais sans recourir à la sélection de *features*.

### 2.2.3 Analyse en dépendances syntaxiques

L'analyse en dépendances syntaxiques identifie les relations existantes entre les différents syntagmes d'une phrase afin de produire un graphe de relations syntaxiques. On distingue deux types d'approches traditionnelles NIVRE et MCDONALD 2008. D'un côté, les approches à base de graphes (*graph-based*) considèrent l'ensemble des arbres de dépendances possibles afin de déterminer, parmi eux, le plus vraisemblable (MCDONALD, NIVRE, QUIRMBACH-BRUNDAGE et al. 2013). Ce problème équivaut à chercher un arbre couvrant de poids maximal, dans un graphe complet dirigé où chaque sommet correspond à un mot de la phrase analysée. De l'autre, les approches à base de transition (*transition-based*) optent pour une redéfinition du problème, en considérant un processus itératif sur les transitions d'une machine abstraite réalisant l'analyse (NIVRE 2008). Cette machine réalise une représentation interne de la phrase<sup>c</sup> contenant les mots qui la compose. À chaque pas, la prochaine transition à effectuer, sachant la configuration courante de la phrase, est prédite automatiquement. Elle détermine ainsi une succession d'actions modifiant la représentation de la phrase analysée, tout en construisant la prédiction à mesure qu'émergent les dépendances syntaxiques.

---

c. On parle de configuration de la phrase



Il s’agit d’un problème fondamental en linguistique et en traitement automatique de la langue ayant une multitude d’applications pratiques, constituant souvent un pré-requis à l’analyse sémantique. Concrètement, l’analyse syntaxique a pour but de produire des éléments structurels de haut-niveau, et d’identifier les relations existant entre-eux, permettant de décrire syntaxiquement une phrase. Par exemple, un verbe (ou prédicat) est ainsi lié à ses dépendants (arguments comme modificateurs). L’analyse en dépendance représente la structure syntaxique de la phrase comme un ensemble de relations binaires entre ses constituants. Collectivement, l’ensemble de ces relations forment un arbre de dépendance faiblement connexe, acyclique et projectif.

Model	LAS
REN, JI, WAN et al. 2009	87.6
CHE, Z. LI, Y. LI et al. 2009	88.5
BOHNET 2010	90.3
SWAYAMDIPTA, BALLESTEROS, DYER et al. 2016	89.8

Table 2.3 – *État de l’art - analyse en dépendance syntaxique.*

Un *Benchmark* du *Parsing* est l’analyse du corpus CoNLL 2009 Shared Task<sup>d</sup>. Ce corpus, fondé sur le Penn Treebank, est composé de près d’un million de mots. Il est annoté en partie de discours, en dépendance syntaxique ainsi qu’en rôle sémantique. Ce domaine a fait l’objet de nombreuses recherches, ayant abouti à la réalisation d’analyseurs performants. L’état de l’art se compose d’analyseurs *graph-based* par recherche d’arbre couvrant maximal (CHE, Z. LI, Y. LI et al. 2009; BOHNET 2010) et *transition-based* utilisant un algorithme *Shift & Reduce* (REN, JI, WAN et al. 2009). Enfin, plutôt que de considérer une réinterprétation des relations, afin de simplifier et de faciliter les problèmes de rattachement, ou une sélection experte et empirique des features, les approches les plus récentes ont pris le parti d’utiliser des réseaux de neurones afin d’identifier les rattachements syntaxiques (VINYALS, KAISER, KOO et al. 2015; COAVOUX et CRABBÉ 2015; SWAYAMDIPTA, BALLESTEROS, DYER et al. 2016).

d. Corpus disponible <http://ufal.mff.cuni.cz/conll2009-st/>

## 2.3 Formalisation des tâches d'analyse

Dans le cadre du TAL, le langage peut être représenté par un flux de caractères destinés à former des mots rassemblés en syntagmes articulés à l'aide de relations de dépendances syntaxiques, ce afin de mettre en relation des concepts, en vue, par exemple, d'exprimer une opinion. Le fait qu'on choisisse de le représenter comme un flux de mots (Yoshua BENGIO, DUCHARME, VINCENT et al. 2003), de caractères (SANTOS et ZADROZNY 2014) ou même d'octets (GILLICK, VINYALS et SUBRAMANYA 2015) ne change rien à la généralité des propos tenus dans cette section. Dans ce qui suit, nous considérerons la *tokenisation* comme acquise. Le langage est alors perçu comme une séquence de mots représentés par leur indice dans un lexique déterminé. Les différentes tâches d'analyse peuvent se réduire à la combinaison d'opérations d'étiquetage, de segmentation et de détection de relation en vue d'être ramenées à des tâches de classification. Il est ainsi possible de formaliser l'ensemble de ces tâches d'analyse comme autant de problèmes d'apprentissage pour lesquels l'entraînement d'un réseau de neurones est envisageable.

### 2.3.1 Étiquetage

L'étiquetage vise à annoter des éléments (mots, constituants syntaxiques, etc.) à l'aide d'une étiquette choisie parmi une liste prédéterminée. Plus précisément, on se place dans le cadre de l'étiquetage de séquences. Cela signifie qu'étant donné une séquence d'observations, on cherche à prédire une séquence d'étiquette de même longueur. L'approche traditionnelle consiste à déterminer de manière indépendante pour chaque élément la probabilité associée à chaque étiquette. Il est possible de combiner ces probabilités d'émission d'étiquette aux probabilités de transition d'une étiquette à l'autre, afin d'optimiser globalement la séquence prédite. On retrouve d'ailleurs explicitement de tels paradigmes dans les *Hidden Markov Models* (RABINER 1989) ou les *Conditional Random Fields* (LAFFERTY, MCCALLUM et F. C. N. PEREIRA 2001). L'inférence des probabilités d'émissions peut également être réalisée à l'aide d'un réseau de neurones type *Multi-Layer Perceptron* (DO et ARTIERES 2010). Il est possible d'obtenir des performances état de l'art à l'aide de réseaux de neurones sans avoir recours à des modèles de langages ou des algorithmes d'optimisation globale de la séquence de prédiction (COLLOBERT, WESTON, BOTTOU et al. 2011). La différence majeure avec les modèles probabilistes précédents étant que les probabilités considérées ne sont plus calculées à partir des fréquences de *N-grammes* observés dans un corpus de référence, mais approchées par un modèle paramétrique dont les paramètres internes ont été optimisés automatiquement.

### 2.3.2 Segmentation

La segmentation vise à découper un flux en segments d'éléments a priori juxtaposés, afin de former des constituants complexes. On cherche à prédire, pour chaque élément, s'il correspond au début (B), à l'intérieur (I) ou à la fin (E) d'un segment, ou bien s'il n'appartient à aucun des constituants complexes qu'on cherche à détecter (O). On ramène ainsi ce problème à une tâche de classification à prises de décisions séquentielles. Il n'est dès lors plus possible de se contenter de prédictions locales mot à mot sans optimisation de la séquence prédite. Les réseaux de neurones récurrents (*Recurrent Neural Networks*) sont dédiés à l'étiquetage explicite de séquences et semblent donc être d'excellents candidats afin de résoudre pareille tâche. En effet, le maintien d'un historique, via un état caché tout au long du traitement d'une séquence donnée, est parfaitement adéquat à ce genre de tâche, où les prédictions moyennes sont très fortement corrélées. Plus précisément, on considère généralement des *Bidirectional Recurrent Neural Networks* afin de combiner les analyses issues des deux sens de lecture possibles (de droite-à-gauche et de gauche-à-droite). La couche récurrente de ces architectures est composée de cellules *Long Short Term Memory* qu'on peut interpréter comme étant une unité de mémoire, où un état interne permet de stocker de l'information.

### 2.3.3 Détection de relation

La détection de relation a pour but d'identifier les liens existants entre différents éléments ou constituants d'une séquence d'entrée. L'ensemble de ces relations forment un graphe pouvant parfois être réduit à un arbre à l'aide de certaines hypothèses propres à la tâche considérée. Dans ce graphe, où chaque élément est associé à un nœud, nous cherchons donc à prédire les arcs. Il est possible de formaliser la détection de relation comme la prédiction, pour chaque mot, de l'indice du mot auquel il est lié. On peut, par exemple, envisager de réaliser une classification binaire au sujet de l'existence d'une relation entre deux éléments donnés. Les relations étudiées dépendant du contexte d'occurrence des éléments, il est donc nécessaire de fournir au modèle une représentation efficace du contexte de cette paire d'éléments. De plus, ces relations s'appliquent parfois sur de très longues distances, par exemple lorsqu'on considère la résolution de certaines anaphores. En plus des heuristiques d'apprentissage propres aux BiRNNs, il est possible d'ajouter des modules d'attention qui au lieu de se focaliser sur l'historique courant vont considérer la succession des états internes

par lequel est passé la couche de récurrence sur l'ensemble de la séquence.

## 2.4 Représentation vectorielle de mot

Afin d'appliquer des modèles numériques au TAL, il est nécessaire de produire des représentations des mots et de leur usage sur lesquelles appliquer des méthodes génériques. Dans les réseaux de neurones, la couche d'entrée assure (avec les pré-traitements) une conversion du type symbolique/numérique entre un observable (un mot, une image, un spectre, etc) et une représentation vectorielle. Dans le cadre de l'étude du traitement automatique de la langue, ces observables sont généralement des mots. On s'intéresse à des systèmes *End-to-End*, ce qui signifie qu'on souhaite avoir recourt au minimum de connaissances du domaine<sup>e</sup>. Donc, la façon dont on va représenter les mots au sein de ces systèmes va directement impacter leurs performances ainsi que leur complexité.

### 2.4.1 Motivations

Qu'est-ce qu'un mot? Un mot est une suite de caractères graphiques possédant un contenu sémantique ou pragmatique. Les mots sont reliés entre eux. D'une part, les formes graphiques dérivent les unes des autres par construction, notamment *via* les dérivations flexionnelles, marqueurs notamment de genre et de nombre, ou *via* les procédés de suffixation et de préfixation permettant de nuancer un contenu sémantique. D'autre part, il est possible de regrouper les mots partageant un contenu sémantique et les relations qui existent entre leurs sens (synonymie, antonymie, hyperonymie, etc.), essentiellement basé sur leurs différents usages.

La forme graphique d'un mot est le seul observable naturel auquel est associé le contenu syntaxique et sémantique propre à ce mot. Il est donc courant de rassembler ces formes graphiques au sein d'un vocabulaire  $V$ . La représentation naturelle des données symboliques est le vecteur *one-hot*. Ce formalisme permet de représenter une variable discrète catégorielle *i.e.* pouvant prendre un nombre fini de valeurs rassemblées au sein d'un lexique. Chaque mot devient ainsi une dimension dans un espace de dimension  $|V|$ . Le produit scalaire entre deux représentations *one-hot* de mots est toujours nul.

---

e. moins dépendre de l'expertise linguistique nécessaire à la sélection des *features*

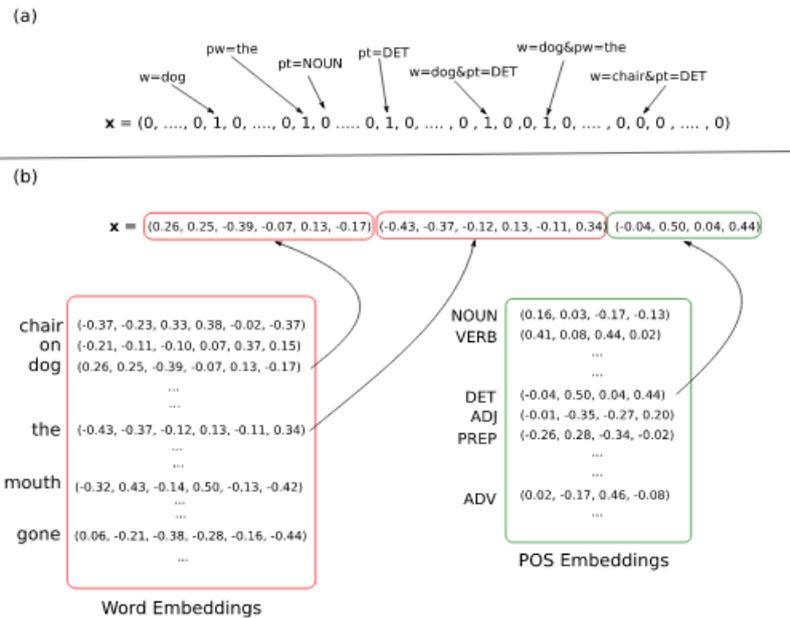


Figure 2.1 – Représentation de features dans le cas du Traitement de la Langue (a) sous forme de one-hot vectors, (b) sous forme de word embeddings. - Goldberg 2015

Ainsi, dans cette représentation *one-hot* les mots sont "orthogonaux" alors qu'il existe des synonymes, ou des mots partageant la même fonction syntaxique. On aimerait au contraire que nos systèmes soient sensibles à ce genre d'invariants afin de mieux généraliser, notamment dans le cas de mots inconnus. De plus, lorsqu'on a  $10^6$  mots différents, les réseaux de neurones ne passent pas à l'échelle, surtout si l'on veut capturer des *N-grammes* (soit  $10^{6N}$  neurones d'entrée). Une solution est de filtrer les mots par rapport à leur fréquence d'apparition. Par exemple, on pourrait fournir une représentation individuelle aux  $10^4$  mots les plus fréquents et une représentation générique pour tout les autres. Mais cela revient à ajouter encore plus de mots inconnus. Nous nous intéressons ici à une seconde solution qui exploite des techniques de projection de mots dans des espaces continus. Il est possible de configurer ces techniques afin que deux mots ayant des natures ou des sens *proches* soient associés à des vecteurs *proches* dans l'espace de représentation.

## 2.4.2 One-hot Vectors

Afin de satisfaire au formalisme des réseaux de neurones décrits dans le chapitre précédent, il est nécessaire d'encoder numériquement les mots en établissant une table d'association liant un mot et sa représentation numérique dédiée. Les *1-hot vectors* sont des vecteurs binaires permettant de représenter trivialement des symboles qu'on considère a priori indépendants. Soit  $|V|$  la taille de notre vocabulaire, on considère comme représentation du mot  $w$  d'indice lexical  $n_w$  le vecteur de dimension  $|V|$  :

$$\mathbf{w} = \begin{cases} \mathbf{w}_{n_w} & = 1 \\ \mathbf{w}_i & = 0 \quad \forall i \neq n_w \end{cases} \quad (2.1)$$

On obtient ainsi une représentation vectorielle des mots, chacun d'eux étant associé à un vecteur indépendant. L'espace de représentation de grande dimension  $|V|$  ainsi créé est peu dense puisque chaque mot y occupe sa propre dimension. Ils en deviennent indissociables puisqu'on perd toute notion de similarité. Avec l'augmentation des ressources disponibles, on a accès à des corpus de plus en plus volumineux. Or,  $|V|$  augmente linéairement avec la taille du corpus, ce qui induit l'utilisation d'espaces de représentation de plus grande dimension. De plus, ce mode de représentation ne peut pas tenir compte des liens existant entre les mots, tels que des affinités étymologiques, flexionnelles, syntaxiques et sémantiques existant entre les mots et leurs usages. On repose donc sur les couches suivantes afin de déterminer les affinités entre deux mots, dans le contexte de la tâche ciblée. On souhaiterait pouvoir considérer des représentations plus riches afin de permettre au modèle de mieux appréhender la nature des éléments qu'il manipule. De plus, se pose le problème du choix de la représentation d'un mot inconnu, absent du lexique initial et apparaissant inopinément dans l'ensemble d'évaluation. Généralement, ces mots sont remplacés par un symbole commun et seront tous représentés par un même indice lexical *i.e.* un même vecteur d'entrée.

## 2.4.3 Word Embeddings

La représentation des mots, entités discrètes par essence, dans des algorithmes d'apprentissage automatique, domaine des valeurs continues, a toujours posé problème. Comme nous l'avons vu, représenter les mots sous forme de *one-hot vectors* n'apporte que peu d'information sur leur nature lexicale, syntaxique et sémantique. Ces dernières années, plusieurs méthodes permettant de projeter un vocabulaire vers un espace continu de vecteurs denses ont été proposées. Elles se sont appuyées sur l'hypothèse distributionnelle (HARRIS 1981) qui sti-

pule que des mots apparaissant dans des contextes similaires ont des sens similaires. Par exemple, il est classique de considérer que des mots qui ont des environnements syntaxiques comparables partagent des propriétés sémantiques. Ainsi, en représentant un mot par son ou ses contextes d'utilisation, des mots employés dans des contextes similaires se verraient représentés de la même façon. D'après (TURIAN, RATINOV et Yoshua BENGIO 2010), l'utilisation de ces représentations distribuées (*word embeddings*), apprises de manière non-supervisée, est une méthode simple et générale afin d'améliorer les performances de systèmes d'apprentissage supervisé, pour le Traitement Automatique de la Langue. En pratique, on peut observer des gains de performances accompagnant l'utilisation de ces *word embeddings* dans des modèles à maximum d'entropie (LECHELLE et LANGLAIS 2014). Cependant, il arrive également que leur impact soit relativement marginal lorsque utilisé au sein de modèles DNN (CERISARA, KRAL et LENC 2017).

L'avantage de ce type de représentation est premièrement de fournir un espace d'entrée de plus faible dimension, ce qui se répercute sur la taille des entrées à fournir au modèle. Dans le cas des réseaux de neurones, cela influence directement les dimensions (nombre de neurones et de connexions) de la couche d'entrée. Cela se caractérise par une augmentation du nombre de paramètres de nos modèles, se qui se répercute sur les temps d'apprentissage nécessaires à l'entraînement du réseau. Ensuite, l'accès à un espace continu de représentations permet de mettre en place des approches plus fines concernant la gestion des mots inconnus, absents du corpus ayant servi à entraîner un modèle, mais apparaissant en évaluation. L'accès à un espace de représentation continu permet d'envisager qu'une "bonne" représentation pour un mot inconnu (si nous la connaissions) devrait être "proche" (dans l'espace de représentation, selon une métrique à déterminer) de celle d'un mot connu "proche" (linguistiquement, selon une similarité à déterminer).

#### **2.4.3.1 Heuristiques d'apprentissage de représentations**

Ces dernières années, plusieurs méthodes permettant de projeter un vocabulaire vers un espace de représentations continues ont été proposées. Elles varient selon les corpus, les algorithmes et la nature des contextes utilisés afin d'apprendre ces représentations. Ces méthodes d'apprentissage de représentations, capables d'être entraînées sur d'importants volumes de données, avec des temps de calcul acceptables, ont été rendues possible d'une part grâce au progrès technologiques permettant une meilleure parallélisation et distribution des calculs, et d'autre part grâce l'avènement de techniques novatrices permettant de minimiser le coût des calculs nécessaires à l'algorithme d'apprentissage. L'objectif de

ces recherches est de proposer un espace de représentation offrant couverture, proximité contextuelle et non-supervision.

Dans les années 2000, la technologie des réseaux de neurones, ainsi que les grandes quantités de données disponibles, ont permis d'envisager des modèles linguistiques dédiés à l'apprentissage de représentations. Yoshua BENGIO, DUCHARME, VINCENT et al. [2003](#) proposent un modèle de langage prédisant un mot compte tenu des  $n$  précédents à l'aide d'un *multi-layer perceptron*. Afin d'optimiser les temps de calcul, les  $n$  mots de la fenêtre sont projetés, quelque soit leur position, vers un espace de plus petite dimension. Cette projection est réalisée à l'aide d'une matrice de paramètres apprise durant l'apprentissage de la tâche de modélisation de langage (*langage modeling*). Il s'avère que ce modèle apprend des *embeddings* pour chaque mot du vocabulaire conjointement à la tâche de prédiction, de façon à maximiser la proximité des *embeddings* des mots utilisés de façon similaire dans le corpus d'entraînement. Dans cette situation, le contexte correspond à une fenêtre de mots précédant le mot à prédire.

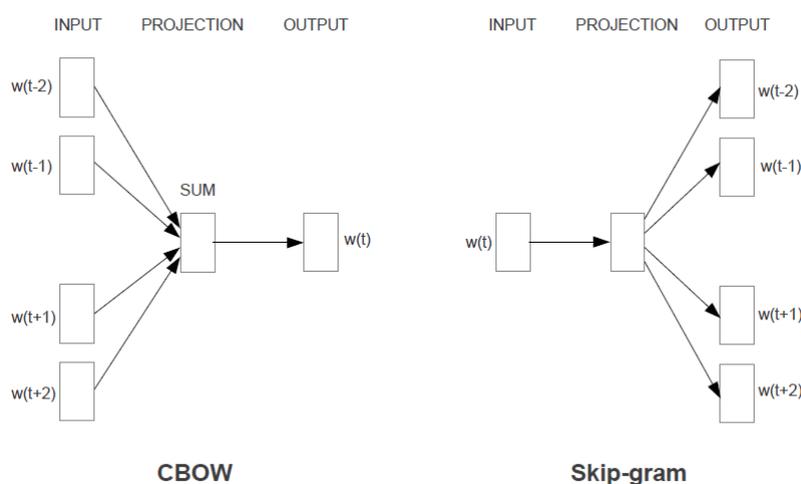


Figure 2.2 – *Word2Vec* - Mikolov, I. Sutskever, K. Chen et al. [2013](#)

MIKOLOV, I. SUTSKEVER, K. CHEN et al. [2013](#) proposent d'entraîner un réseau de neurones (*Word2Vec*) à prédire un mot, sachant les mots environnants, et réciproquement. Ils introduisent la notion de *negative sampling*. Plutôt que d'optimiser le score du mot attendu par rapport à toutes les autres prédictions possibles, on cherche uniquement à maximiser la différence de score entre une *bonne* et une *mauvaise* réponse (notées respectivement  $x$  et  $\tilde{x}$ ) au moyen d'un *pair-wise loss* :

$$\mathcal{L} = \sum_{(x, \tilde{x}) \in \mathcal{X} \times \tilde{\mathcal{X}}} \max(0, \delta + f(x) - f(\tilde{x})) \quad (2.2)$$

L'architecture proposée est composée de deux couches linéaires (sans fonction d'activation non-linéaire). Tout comme dans Yoshua BENGIO, DUCHARME, VINCENT et al. 2003, on peut interpréter la matrice des poids de la couche d'entrée comme une projection linéaire, permettant de passer de l'espace des mots sous forme de *one-hot vectors*, à un espace continu de plus petite dimension. Deux heuristiques d'apprentissage ont été proposées afin d'entraîner cette matrice.

L'approche *Continuous Bag of Words* (CBOW) consiste à entraîner le réseau à prédire un mot à partir de son contexte. Il s'agit dans ce cas d'une fenêtre de mot centré sur le mot d'intérêt. À l'inverse, l'approche *Skip-gram* se propose de prédire le contexte d'un mot donné en entrée. En d'autres termes, à partir d'un mot, on cherche à prédire l'ensemble des mots composant l'ensemble des fenêtres centrées sur ce dernier. Une fois l'un de ses deux entraînements arrivé à son terme, les *embeddings* sont extraits des activations de la couche linéaire d'entrée du réseau. Le nombre de dimensions  $n$  de ces *embeddings* est ainsi librement paramétrable car totalement déterminée par le nombre de neurones composant cette couche de projection.

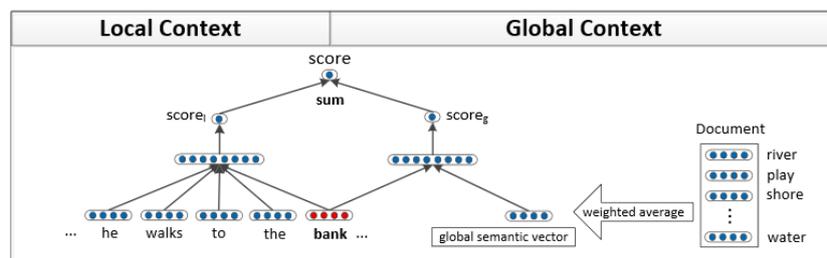


Figure 2.3 – *GloVe* - E. H. Huang, Socher, C. D. Manning et al. 2012

E. H. HUANG, SOCHER, C. D. MANNING et al. 2012 proposent un système prenant en compte aussi bien le contexte local que le contexte global des mots. Les auteurs ont utilisé un réseau de neurones (*GloVe*) considérant deux types de *features* dans le calcul des *embeddings*. D'une part, ils considèrent les mots environnants le mot d'intérêt (le contexte local). D'autre part, ils tiennent compte du vocabulaire du document dans lequel apparaît le mot d'intérêt (le contexte global). Ces deux représentations du contexte sont indépendamment combinées à celle du mot d'intérêt afin d'obtenir deux scores, qui seront ensuite sommés. Les auteurs abordent également un problème récurrent de l'apprentissage de *word*

*embedding* : la notion de polysémie et d'homonymie.

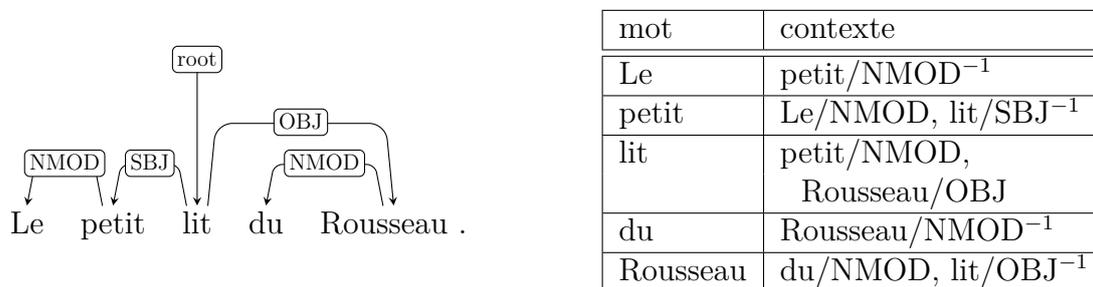


Figure 2.4 – *W2Vf* - Levy et Goldberg 2014

LEVY et GOLDBERG 2014 ont remis en cause le fait que les approches précédentes utilisaient uniquement des fenêtres de mots. Ils proposent de généraliser le modèle *Skip-Gram* afin de prendre en compte des contextes plus arbitraires. En particulier, ils s'intéressent aux contextes qu'on peut dériver des relations syntaxiques dans lesquelles un mot joue un rôle. Cette méthode permet de prendre en compte des mots potentiellement éloignés du mot cible, mais liés à lui syntaxiquement. Cette hypothèse rejoint notre analyse qui soutient que la sémantique et la syntaxe sont étroitement liées et donc que des mots qui ont des environnements syntaxiques comparables partagent des propriétés sémantiques communes.

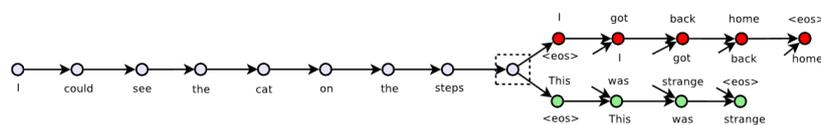


Figure 2.5 – *Skip-Thought Vectors* - Kiros, Zhu, Salakhutdinov et al. 2015

KIROS, ZHU, SALAKHUTDINOV et al. 2015 ont par ailleurs montré qu'il était possible de généraliser l'approche *Skip-Gram* en l'étendant à l'apprentissage de représentation de phrase. Ils ont entraîné un système *Sequence-to-Sequence* composé de deux RNNs. Un encodeur, qui prend une phrase  $s_i^{1..T}$  en entrée et produit une représentation  $h^T$ , et un décodeur qui tente de reconstruire la phrase précédente  $s_{i-1}$  et la phrase suivante  $s_{i+1}$ . Dans l'exemple 2.5, les phrases de référence sont "I got back home. I could see the cat on the steps. This was strange."

### 2.4.3.2 Extension aux représentations de *features*

Il est possible d'étendre l'apprentissage non-supervisé de représentations distributionnelles aux *features* communément utilisées en Traitement Automatique de la Langue, telles que les lemmes et flexions, les classes morpho-syntaxiques, les étiquettes de segmentation et les annotations en dépendance. Pour ce faire, on a cependant besoin de données annotées ou d'analyseurs automatiques performants, à partir desquels on extrait des séquences de *features* qui serviront de base d'apprentissage aux heuristiques précédemment exposées.

### 2.4.4 Application aux réseaux de neurones

Les méthodes citées précédemment permettent la mise au point de modes de représentation vectorielle non-supervisée de mots, à partir d'importants volumes de données textuelles, en s'appuyant sur l'hypothèse distributionnelle. On se propose désormais d'étudier l'utilisation de ces représentations dans des réseaux de neurones entraînés sur une tâche spécifique. Apprendre à réaliser une tâche en même temps que les représentations des entrées, est le propre des réseaux de neurones. On souhaite tirer profit de cette propriété afin de mettre au point des systèmes capables de tirer parti de représentations génériques faiblement supervisées, tout en les adaptant à la tâche et au corpus considéré.

#### 2.4.4.1 Injection statique des *embeddings*

On souhaite utiliser les *word embeddings* au sein d'un réseau de neurones. Une solution simple consiste à utiliser ces *embeddings* comme vecteurs d'entrée afin de représenter directement les mots, en lieu et place des vecteurs *one-hot* binaires. En plus de fournir au réseau des représentations plus fines des mots, ceci permet de réduire significativement la dimension de la couche d'entrée et donc le coût de la rétro-propagation du gradient dans celle-ci. Ce faisant, l'espace d'*embedding* initial devient un ensemble de constantes du modèle. Ce n'est pas réellement problématique si les corpus utilisés, d'un part pour l'apprentissage de représentation et d'autre part pour entraîner le réseau, sont relativement proches, en terme de lexique comme de registre de langue.

Cependant, en pratique, le problème se pose lorsque l'on souhaite traiter du langage non-canonique, tel que des transcriptions automatiques de conversations orales. Les données de ce type étant rares et chères, on n'a généralement

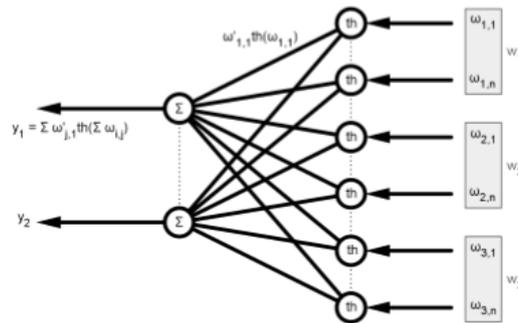


Figure 2.6 – Architecture MLP prenant des embeddings en entrée

accès qu'à un nombre limité d'exemples d'apprentissage. Si cette quantité de données annotées est suffisante pour entraîner un modèle à réaliser la tâche d'intérêt, l'apprentissage de représentations distributionnelles de bonne qualité nécessite une quantité de texte d'un tout autre ordre de grandeur. Les *embeddings* sont donc appris sur de larges corpus de texte non annotés, généralement encyclopédiques et journalistiques. Les représentations ainsi extraites ne correspondent pas parfaitement aux données sur lesquelles on souhaite les appliquer. Ne pas mettre à jour les représentations vectorielles des mots lors de l'apprentissage n'est donc pas optimal.

#### 2.4.4.2 Raffinage des embeddings

On souhaite permettre au réseau de transformer les représentations des mots au fur et à mesure de l'apprentissage. Ceci permettrait d'adapter les *embeddings* initiaux, calculés sur un grand nombre de données  $C_{emb}$ , afin qu'ils satisfassent aux spécificités propres au petit jeu d'exemples utilisés lors de l'entraînement. Ce dernier ensemble  $C_{task}$  peut-être particulièrement distant de  $C_{emb}$  si on considère différents types de modalités de langage (écrit ou oral) comme de registres de langue (journalistique, technique, scientifique d'un côté; débat, interview, conversation téléphonique de l'autre) caractéristiques du contexte entourant ces données.

Le rôle de la couche d'entrée d'un réseau de neurones est de projeter (via. une fonction  $\Phi$ ) des vecteurs d'entrée dans un espace de représentation continu. Chaque mot est représenté par un vecteur de *features* composé de  $N$  éléments formant un tuple  $x = (x_1, \dots, x_N) \in \mathcal{X}_1 \times \dots \times \mathcal{X}_N$ . Dans cette description,  $\mathcal{X}_n$  fait référence à l'ensemble des valeurs pouvant être prise par la  $n^{eme}$  *feature*.

On parlera de lexiques associés aux *features*<sup>f</sup>. On décrit premièrement le fonctionnement de notre couche d'entrée dans le cas  $N = 1$ , c'est-à-dire lorsqu'une occurrence est uniquement représentée par un mot, sans autre information que son indice dans un lexique  $\mathcal{X}$ . Puis on généralise dans le cas où s'ajoutent à ce mot d'autres informations, provenant le plus souvent de traitements préliminaires et formant un ensemble de *features* additionnelles dont on souhaite faire dépendre l'analyse.

**Cas  $N = 1$**  Soit un corpus  $C_{\text{task}}$  composé de mots formant un vocabulaire  $\mathcal{X}$ . On note  $|\mathcal{X}|$  le nombre de mots différents composant ce lexique  $\mathcal{X}$ . Les exemples d'apprentissage deviennent donc des entiers représentant des mots à l'aide d'un lexique indicé. Comme présenté à la section [section 2.4.2](#), il est possible de plonger ces indices dans un espace vectoriel de dimension  $|\mathcal{X}|$  à l'aide de *one-hot vectors* notés  $\mathbf{x}$  où chaque dimension est associée à un et un seul mot.

$$\mathbf{x} = \begin{cases} \mathbf{x}_x = 1 \\ \mathbf{x}_i = 0 \end{cases} \quad \forall i \neq x \in \{0, 1\}^{|\mathcal{X}|} \quad (2.3)$$

Les données d'entrée sont donc représentées à ce stade comme des vecteurs binaires  $\mathbf{x}$  de dimensions  $|\mathcal{X}|$ . On obtient ainsi une représentation vectorielle de très grande dimension  $\mathbf{x} \in \{0, 1\}^{|\mathcal{X}|} \subset \mathbb{R}^{|\mathcal{X}|}$ . On souhaite obtenir des représentations vectorielles plus denses *i.e* de dimension plus faible. Il est possible de projeter les représentations binaires  $\mathbf{x}$  dans un espace d'*embedding* de dimension  $d$  à l'aide d'une application linéaire  $\Phi$ .

$$\begin{aligned} \mathbf{e} = \Phi(\mathbf{x}) &= W_{\text{INPUT}} \cdot \mathbf{x} \\ &= (\mathbf{w}_1 \dots \mathbf{w}_x \dots \mathbf{w}_{|\mathcal{X}|}) \cdot (0 \dots 1 \dots 0) \\ &= \mathbf{w}_x \in \mathbb{R}^d \end{aligned} \quad (2.4)$$

Cela revient à appliquer une matrice  $W_{\text{INPUT}} \in \mathbb{R}^{d \times |\mathcal{X}|}$  à nos vecteurs  $\mathbf{x}$ . La matrice  $W_{\text{INPUT}}$  est une matrice de paramètres dont les valeurs seront déterminées lors de l'apprentissage.  $\mathbf{w}_i \in \mathbb{R}^d$  est la  $i^{\text{eme}}$  ligne de cette matrice et correspond à la représentation vectorielle du  $i^{\text{eme}}$  mot dans  $\mathcal{X}$ . Cette représentation peut aussi bien être initialisée aléatoirement ou à l'aide d'*embeddings* obtenus par apprentissage non-supervisé. On obtient ainsi une représentation vectorielle de faible dimension  $\mathbf{e} \in \mathbb{R}^d$ . La dimension  $d$  de ce nouvel espace de représentation est un hyper-paramètre, fixé *a priori*.

---

f. De façon analogue et afin d'homogénéiser les notations, les analyses attendues sont ramenées à des annotations, syntaxiques ou sémantiques en fonction de la tâche d'intérêt, appartenant à un lexique  $\mathcal{Y}$ .

Cette première couche a donc pour but de transformer un mot d'entrée  $x$  en un vecteurs  $\mathbf{e}$  de dimension  $d$  via l'application de  $W_{\text{INPUT}}$ . En pratique, l'implémentation standard consiste à utiliser des tables d'association (*Look-up Tables*) afin de réaliser une *embedding layer* sans avoir à réaliser le produit de la représentation *one-hot*  $\mathbf{x} \in \mathbb{R}^{|\mathcal{X}|}$  par la matrice d'*embedding*  $W_{\text{INPUT}} \in \mathbb{R}^{d \times |\mathcal{X}|}$ . Il est important de noter que cette matrice de paramètres sera modifiée automatiquement tout au long de l'apprentissage. Ce qui signifie que la représentation associée à chaque mot sera raffinée à chaque fois que l'algorithme considérera une de ses occurrences.

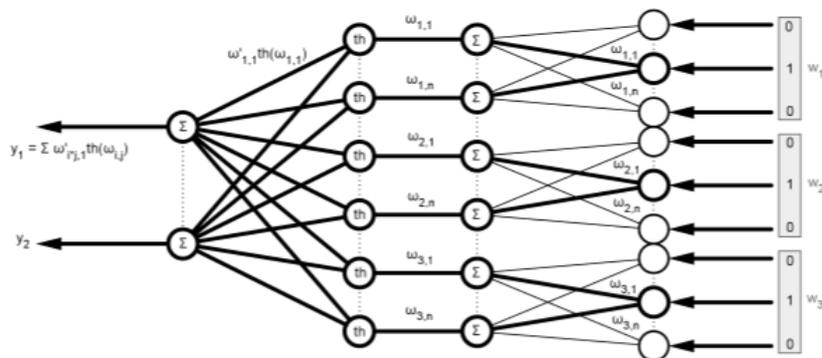


Figure 2.7 – Architecture MLP permettant un raffinement des embeddings

**Cas général** Afin de prendre en compte les spécificités linguistiques d'une tâche, on peut tirer profit d'un vecteur de *features* pour caractériser les mots (lemme, morphologie, annotations syntaxiques ou sémantiques résultant d'analyses préliminaires, etc.). Chaque mot est associé à un certain nombre de *features* descriptives qu'on souhaiterait prendre en compte dans l'analyse. On généralise le processus d'encodage de la couche d'entrée en considérant qu'un mot  $x$  peut se décomposer en  $N$  éléments formant un tuple  $x = (x_1, \dots, x_N) \in \mathcal{X}_1 \times \dots \times \mathcal{X}_N$  où  $\mathcal{X}_n$  correspond au lexique associé à la  $n^{\text{eme}}$  *feature*.

Soit  $x = (x_1, \dots, x_N) \in \mathcal{X}_1 \times \dots \times \mathcal{X}_N$ . Par exemple, la *feature* principale  $x_1$  d'un mot est son indice dans le vocabulaire. On utilise  $N$  *Look-Up Tables* afin d'encoder chaque *feature* mais tout se passe comme si on appliquait individuellement le procédé d'encodage décrit précédemment (cf. équation 2.3). Chaque *Look-Up Table* peut être représentée comme une application linéaire  $W_{\text{INPUT}_n} \in \mathbb{R}^{d_n \times |\mathcal{X}_n|}$  permettant de projeter individuellement chaque *feature*  $\mathbf{x}_n$  dans un espace de dimension respective  $d_n$ . Ces nouvelles représentations denses des *features* sont ensuite concaténées afin de former la représentation  $\mathbf{e}$  du vecteur d'entrée  $\mathbf{x}$ .

$$\mathbf{e} = \Phi(\mathbf{x}) = W_{\text{INPUT}_1} \cdot \mathbf{x}_1 \bullet \cdots \bullet W_{\text{INPUT}_N} \cdot \mathbf{x}_N \in \mathbb{R}^d \quad (2.5)$$

Un mot  $x$  est ainsi représenté comme la concaténation des représentations vectorielles de ses *features* associées. Les dimensions respectives  $d_n$  de ces représentations sont autant d'hyper-paramètres. En posant  $\sum d_n = d$ , on retombe sur un vecteur  $\mathbf{e} \in \mathbb{R}^d$ . Cette couche d'entrée généralisée transforme donc un mot annoté en un vecteur de  $\mathbb{R}^d$ .

Les matrices contenant les *embeddings*  $W_{\text{INPUT}_{1..N}}$  sont peu à peu modifiées durant l'apprentissage supervisé lorsque l'algorithme de rétro-propagation du gradient optimise les paramètres du réseau de neurones. On note  $\Phi^r$  la transformation (raffinée) après apprentissage et  $W_{\text{INPUT}}^r$  les *embeddings* associés. Afin de ne pas trop dénaturer les représentations initiales  $W_{\text{INPUT}}^0$  lors de l'adaptation de l'espace d'*embedding*, on ajoute un terme de régularisation spécifique à la transformation  $\Phi$  via la matrice  $W_{\text{INPUT}}$  au critère d'apprentissage. Les représentations raffinées sont ainsi contraintes de ne pas trop s'éloigner de leur valeur initiale.

$$\mathcal{L} = \sum_{x \in \mathcal{X}} \log p(y_k | x, \theta) - \lambda \|W_{\text{INPUT}} - W_{\text{INPUT}}^0\|^2 \quad (2.6)$$

## Conclusion

Ce chapitre avait pour but premier de présenter les tâches d'analyse linguistiques les plus fréquemment étudiées (*section 2.2*) : étiquetage morpho-syntaxique (*Tagging*), analyse syntaxique de surface (*Chunking*) et analyse en dépendances syntaxiques (*Parsing*). Il est possible de formaliser ces analyses comme des problèmes de classification afin de tenter de les résoudre à l'aide de réseaux de neurones. Dans un second temps, nous avons détaillé la notion de *word embeddings*, des représentations vectorielles de mots provenant d'algorithmes d'apprentissage basés sur l'hypothèse distributionnelle. Ces vecteurs permettent de représenter les mots dans un espace continu de faible dimension. Ils peuvent être utilisés en entrée d'un réseau, ou bien afin d'initialiser les poids de sa couche d'entrée avant son apprentissage. Cette méthode permet de raffiner les représentations des mots, c'est-à-dire qu'elles seront modifiées par l'algorithme d'apprentissage. On espère ainsi tirer parti de représentations génériques faiblement supervisées, tout en les adaptant à la tâche et au corpus considérés.

Appliquer des réseaux de neurones, et plus généralement le *Deep Learning* au Traitement Automatique de la Langue, est devenu omniprésent dans la littérature. Les modèles qui en sont issus se sont montrés tout aussi efficaces que les systèmes traditionnels sur les tâches d'analyse syntaxique et sémantique. Les réseaux de neurones peuvent tirer profit des *word embeddings* afin d'augmenter leur capacité de généralisation et de se passer, en partie, de connaissances *a priori* des structures syntaxiques et sémantiques du langage. Il est devenu possible d'entraîner avec moins de supervision des réseaux neuronaux sur d'importants volumes de données qui généralement égalent, voire surpassent, les méthodes probabilistes classiques.

Au cours du chapitre suivant, nous montrerons que la flexibilité structurelle des réseaux de neurones permet de proposer un apprentissage efficace de modèles joints, réalisant conjointement les tâches d'analyse évoquées dans ce chapitre. Nous proposons d'utiliser un modèle multitâche d'analyse syntaxique et sémantique en tant qu'alternative au *pipeline* traditionnel, notamment afin de réduire les phénomènes de cumul des erreurs. Ce DNN sera par la suite appliqué à la compréhension automatique de la langue orale au travers de transcriptions de conversations téléphoniques. Dans ce contexte, nous montrerons que des modèles neuronaux basés sur les *word embeddings* permettent une meilleure généralisation, au travers de la gestion des mots hors-vocabulaire.

# 3 Algorithme d'apprentissage multitâche pour l'analyse syntaxique

Le Traitement Automatique de la Langue repose sur un découpage hiérarchique des différents niveaux d'analyse linguistique. L'approche traditionnelle consiste à développer des modèles spécifiques à chacune de ces analyses au sein de chaînes de traitement. La résolution séquentielle entraîne un phénomène de cumul des erreurs et les modèles composant ces *pipelines* sont sensibles aux changements de domaine. On souhaite s'abstraire de ces limitations en développant un système d'analyse globale au moyen de modèles génériques et de stratégies d'adaptation.

Ce chapitre a pour objectif d'étudier un modèle multitâche basé sur l'apprentissage de représentations communes à un ensemble de tâches. Cet algorithme consiste à considérer autant de réseaux LSTM-BiRNN que de tâches d'intérêt. Les poids de la couche d'entrée et de la couche cachée récurrente sont partagés par l'ensemble des réseaux qui sont entraînés conjointement à l'aide d'un critère d'erreur globale. Cet algorithme peut être utilisé afin de transformer des *features* d'entrée en sortie additionnelle. En effet, plutôt que de fournir une *feature* en entrée du réseau, il est possible de considérer une tâche supplémentaire réalisant sa prédiction. C'est intéressant lorsqu'on considère des *features* prédites<sup>a</sup>. Durant la phase d'évaluation, ceci permet de rendre les prédictions du modèle indépendantes de la qualité des *features* *i.e* de s'affranchir des erreurs commises lorsqu'on prédit ces *features*, puisqu'elles ne sont alors plus fournies en entrée du réseau. Nous validerons notre approche en l'appliquant à l'apprentissage de tâches de référence de l'analyse linguistique.

## Contexte et Motivations

Les modèles du *Deep Learning* appliqués à la résolution de diverses tâches du TAL au sein de *pipelines* d'analyse linguistique ont déjà prouvé leur efficacité. En effet, les réseaux de neurones profonds sont des modèles de classification performants, qui peuvent être le plus souvent utilisés en lieu et place d'un ou plu-

---

a. par opposition aux *features* qu'on extrait des données prises en entrée

sieurs modules d'un *pipeline* traditionnel (COLLOBERT, WESTON, BOTTOU et al. 2011). Cependant, ces chaînes de traitement sont développées afin de résoudre une tâche d'analyse de haut niveau linguistique (sémantique, analyse d'opinion, etc.). L'ensemble des modules intermédiaires sont chargés de réaliser les analyses pré-requises à l'aide des résultats des analyses précédentes. Cela signifie que l'agencement des modules dans le *pipeline* impacte leur comportement individuel. Chacun de ces modèles entraînés est spécifique non-seulement à sa tâche d'intérêt mais aussi à l'architecture du *pipeline* dans lequel il s'inscrit. C'est dans ce contexte que nous souhaitons développer un DNN dédié au TAL et donc spécifique à la nature de la Langue, mais tout en étant aussi indépendant que possible de sa tâche d'application.

On souhaite utiliser une même architecture neuronale, par le biais de différentes instances, afin d'apprendre indépendamment différentes tâches. Afin de réaliser une tâche d'analyse de haut niveau linguistique nécessitant des *features* spécifiques, l'approche *pipeline* consisterait à disposer des réseaux en cascade, afin de réaliser en amont un ensemble d'analyses produisant automatiquement ces *features*. En alternative à ces approches séquentielles de type pipeline, nous proposons d'utiliser le *Multi Task Learning* afin de superviser l'apprentissage de représentation intermédiaire à l'aide d'une tâche sur les *features* pré-requises.

Nous souhaitons ainsi nous abstraire du traitement séquentiel des tâches, induit par la structure des *pipelines*. Au contraire, on privilégie une analyse plus globale, considérant simultanément l'ensemble des tâches d'intérêt. On espère que réaliser un apprentissage conjoint de l'ensemble des tâches permettra de dégager des bénéfices mutuels, ainsi que de pallier les phénomènes de propagation des erreurs d'analyse habituellement observées dans les *pipelines*. Ce type d'apprentissage multitâche bien connu permet de traiter plusieurs tâches simultanément, et de factoriser ce qui est commun à l'ensemble des tâches et ce qui est spécifique à chacune. On espère généralement que la supervision d'une tâche permette d'apprendre des représentations plus robustes pour les autres tâches<sup>b</sup>. Cela passe par la mise en commun de représentations *via* le partage de paramètres internes entre les différents réseaux.

### 3.1 État de l'Art

L'apprentissage multitâche (*Multi Task Learning* / *MTL*) a été originellement introduit et décrit par CARUANA 1997 afin de résoudre conjointement différents

---

b. ce qui peut être interprété comme une forme de régularisation.

problèmes de classification à l'aide d'un unique réseau de neurones. Plus généralement, il est destiné à la réalisation d'un ensemble de tâches à priori reliées entre elles à l'aide d'un unique *classifieur*. Ceci équivaut à considérer autant de réseaux que de tâches, mais dont une partie des paramètres internes est partagée. Généralement, les paramètres partagés entre tous les réseaux sont localisés dans les premières couches. En effet, c'est dans cette région que sont calculées les représentations communes qui seront ensuite classifiées par les couches cachées spécifiques à chaque tâche d'intérêt. Ces réseaux, et plus particulièrement ces représentations partagées, sont entraînés en parallèle sur chacune des tâches. Ce qui permet d'apprendre une tâche est mis en commun afin de faciliter l'apprentissage d'autres tâches.

Les avancées dans le domaine de l'apprentissage automatique, et plus précisément du *Deep Learning*, ainsi que l'augmentation croissante des quantités de données d'apprentissage et de la puissance de calcul disponibles ont donné lieu, depuis une dizaine d'années, à de nombreux travaux portant sur l'application de réseaux de neurones au Traitement Automatique de la langue, et notamment dans le cadre d'apprentissage multitâche. Cette section est consacrée à la présentation des travaux de (COLLOBERT et WESTON 2008), (DONG, H. WU, HE et al. 2015) et (LUONG, LE, Ilya SUTSKEVER et al. 2015) qui proposent trois modèles d'apprentissage multitâche appliqués à des données textuelles à l'aide d'architectures neuronales respectivement *feed-forward* et récurrentes.

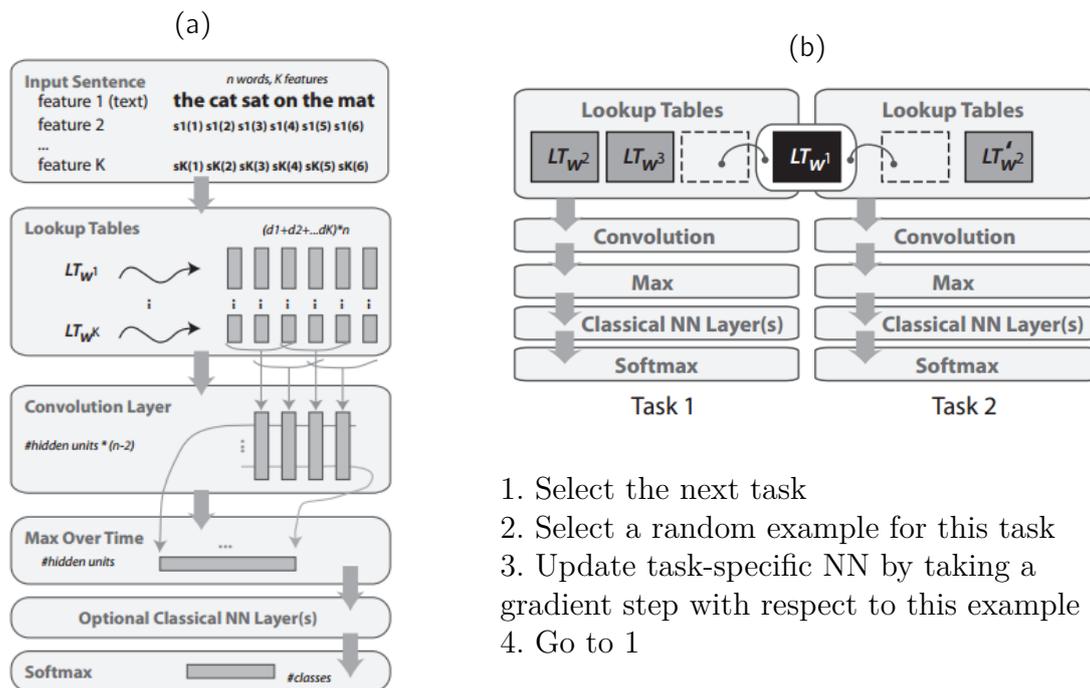
Avant d'entrer davantage dans les détails, il est important de noter que même si les architectures diffèrent d'un auteur à l'autre, et que les tâches d'application sont radicalement différentes, elles se résument toutes à réaliser une collection de  $K$  tâches à l'aide de réseaux partageant certains de leurs paramètres internes. On formalise chaque tâche  $Y_k$  comme un problème de classification qu'on résout par optimisation d'une architecture neuronale.

$$C_{\text{task}} = (X, Y_1, \dots, Y_K) = (\mathbf{x}^t, \mathbf{y}_1^t, \dots, \mathbf{y}_K^t)_{t=1..T} \quad (3.1)$$

$$\text{avec } \forall t \mathbf{x}^t \in \mathcal{X} \subset \mathbb{N} \text{ et } \forall k \forall t \mathbf{y}_k^t \in \mathcal{Y}_k \subset \mathbb{N}$$

### 3.1.1 A Unified Architecture for NLP : DNN with MTL

COLLOBERT et WESTON 2008 ont proposé un réseau de neurones *feed-forward* entraîné sur six tâches d'analyse différentes : l'étiquetage morpho-syntaxique, l'analyse syntaxique de surface, la reconnaissance d'entités nommées, la détection de rôle sémantique, la détection de synonymie et un modèle de langage.



1. Select the next task
2. Select a random example for this task
3. Update task-specific NN by taking a gradient step with respect to this example
4. Go to 1

Figure 3.1 – Approche MTL envisagée par Collobert et Weston 2008 dans le cadre de l'analyse syntaxique - (a) Une architecture générique dédiée au TAL qui prédit la classe d'un mot à partir d'une fenêtre injectée dans un Time-Delay Neural Network. Les représentations vectorielles des mots et de leur features sont accessibles via des Lookup tables. - (b) Un exemple de système multitâche utilisant cette architecture pour apprendre conjointement deux tâches. Une Lookup table est partagée entre les TDNN, tous les autres paramètres étant spécifiques. Ces réseaux sont entraînés à l'aide d'un algorithme qui alterne entre les tâches.

Ils ont considéré une architecture relativement classique (cf. figure 3.1) composée d'une couche d'entrée contenant les *Lookup-Tables* associé à chaque *features* d'entrée. Les sorties de cette couche sont injectées dans un *Time-Delay Neural Network* (TDNN). Cette couche prend en entrée une séquence de taille variable et réalise une convolution sur l'ensemble des fenêtres de mots. S'ensuit un certain nombre de couches denses aboutissant sur un *softmax* analogue à un *Multi-layer Perceptron* (MLP) réalisant la classification des représentations des occurrences produites par le TDNN.

À cette époque, on sait entraîner efficacement des réseaux convolutionnels, notamment grâce aux travaux de Yann LeCun (Y. LECUN, L. BOTTOU, Y. BENGIO et al. 2001). Respectivement, bien qu'obtenant de bon résultats, les RNN n'en

sont qu'à leurs prémices, tant en terme d'implémentation que d'entraînement efficace (en temps d'apprentissage). Ces derniers sont encore principalement exploités dans le cadre de la modélisation du langage (Yoshua BENGIO, SIMARD et FRASCONI 1994). Le contexte d'une occurrence est jusqu'alors classiquement représenté par une fenêtre de mots, centrée sur le mot d'intérêt. COLLOBERT et WESTON 2008 ont choisi de représenter le contexte *via* une convolution sur la séquence des fenêtres de mots, suivie d'un *pooling* à l'aide d'un *Max over Time*. Cela permet d'agréger les *features* les plus pertinentes sur l'ensemble d'une séquence d'entrée de taille variable et revient à modéliser un sac de *N-grammes*.

Concernant l'algorithme d'apprentissage multitâche (*cf. figure 3.1*), chaque tâche est associée à son corpus d'apprentissage et à son propre réseau de neurones. Les différentes tâches sont ainsi susceptibles d'avoir des *features* d'entrée différentes et l'architecture unifiée alloue autant de *Lookup-Table* que nécessaire. Les *features* en commun entre les tâches (typiquement les mots) partagent la même *Lookup-Table* d'un réseau à l'autre. Ces réseaux sont entraînés en alternance : l'algorithme d'apprentissage choisi aléatoirement une tâche (donc un réseau) et un exemple associé à cette tâche. Les paramètres du réseau sélectionné, partagés ou non, sont ensuite optimisés vis-à-vis de l'erreur commise sur cet exemple. Cette méthode permet d'apprendre des espaces de représentations de *features* optimisés pour l'ensemble des tâches. Ces représentations sont ensuite contextualisées (*via* TDNN) puis classifiées (*via* MLP) par des couches spécifiques à chaque tâche *i.e.* entraînées indépendamment d'une tâche à l'autre.

### 3.1.2 Multi-Task Learning for Multiple Language Translation

DONG, H. WU, HE et al. 2015 ont proposé un réseau de neurones récurrents dédié à la traduction automatique multilingue. Il s'agit d'une architecture de type *Sequence-to-Sequence*. Elle est composée de deux réseaux de neurones récurrents dédiés à la traduction d'un langage source (en l'occurrence l'Anglais) en un certain nombre de langues cibles (Français, Espagnol, Allemand et Portugais). Ce réseau est entraîné en *Encoder-Decoder* à l'aide de corpus parallèles. L'*Encoder* est commun à l'ensemble des paires de langues, tandis que les *Decoders* sont spécifiques à chaque langue cible.

Le RNN *Encoder* est un réseau récurrent bidirectionnel composé de cellules GRU (*Gated Recurrent Unit*). Ces cellules comportant une *reset gate* possèdent des propriétés analogues aux LSTM. Elles pallient les problèmes de disparition du gradient ou *Vanishing Gradient* (*cf. section 1.4.2.1*) et permettent de détecter des régularités longue distance grâce à une gestion explicite de leur état interne.

Ce Bi-GRU est entraîné à produire des représentations communes à l'ensemble des quatre tâches de traduction *i.e* de l'Anglais vers le Français, l'Espagnol l'Allemand et le Portugais. Les représentations produites par l'Encoder forment une séquence de même taille que la séquence d'entrée. Chaque élément de cette séquence est la concaténation des historiques droit et gauche du Bi-GRU. Ces représentations sont injectées dans autant de modèles d'attention que le problème compte de langues cibles. Ils sont similaires à celui présenté dans le chapitre précédent et sont chargés de produire une représentation de taille fixe de la séquence d'historiques issue de l'Encoder. Ce sont ces représentations, spécifiques à chaque langue cible, qui seront injectées dans leur Decoder respectif.

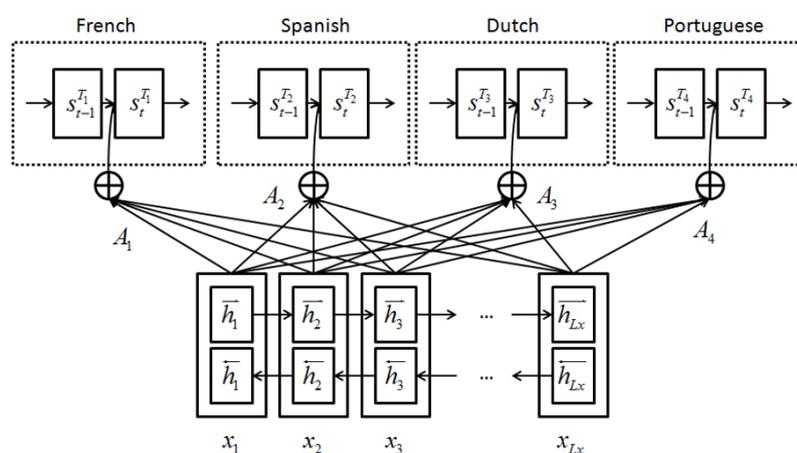


Figure 3.2 – *Approche MTL envisagée par Dong, H. Wu, He et al. 2015 dans le cadre de la traduction automatique. Ils réalisent la traduction simultanée de l'anglais vers le Français, l'Espagnol l'Allemand et le Portugais. Leur modèle se compose d'un Encoder BiRNN et de quatre Decoders RNN. La communication entre Encoder et Decoder passe par un modèle d'attention spécifique au Decoder. Les quatre Sequence-to-Sequence ainsi définis sont entraînés à l'aide d'un algorithme considérant une alternance de mini-batches monolingues.*

Concernant l'algorithme d'apprentissage multitâche, chaque paire de langues est associée à son propre corpus parallèle d'apprentissage. Le réseau est entraîné par descente de gradient stochastique avec mini-batch, où chaque mini-batch est composé d'exemples d'une paire de langues en particulier. Cela signifie que les Decoders sont entraînés en alternance : l'algorithme choisit aléatoirement un mini-batch (et donc une paire de langues) en veillant à alterner les langues cibles. Cette méthode permet d'entraîner l'Encoder à produire une représentation de la séquence d'entrée Anglaise, optimisée pour l'ensemble des tâches de

traduction. Les représentations composant la séquence analysée sont par la suite combinées par les modèles d'attention indépendants, en vue de produire des représentations spécifiques à chaque langue cible. C'est cette représentation de la séquence d'entrée, propre à chaque tâche de traduction, qui sera injectée en entrée des *Decoders*.

### 3.1.3 Multi-task Sequence to Sequence Learning

LUONG, LE, Ilya SUTSKEVER et al. 2015 ont proposé une approche *Sequence-to-Sequence* basée sur de multiples *encoders* et *decoders* partagés. Ils s'intéressent à une grande variété de tâches : la traduction de l'anglais vers l'allemand et réciproquement, le *parsing* du *Penn TreeBank*, la génération de légendes à partir d'images (*image captioning*) et un *auto-encoder* non-supervisé. Leur approche peut être considérée comme une généralisation de celle proposée par DONG, H. WU, HE et al. 2015.

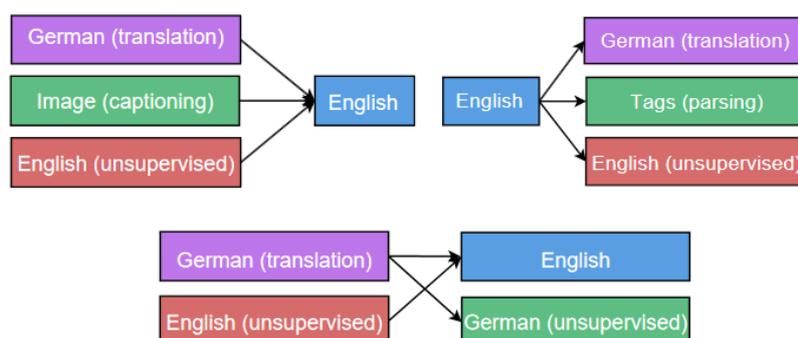


Figure 3.3 – Approche MTL envisagée par Luong, Le, Ilya Sutskever et al. 2015 dans le cadre de la traduction automatique, du parsing et de l'image captioning. Ils proposent trois types de structure *Sequence-to-Sequence* où les *encoders* et les *decoders* sont partagés : "one-to-many", "many-to-one" et finalement, la plus générale, "many-to-many". Les couples (*encoder/decoder*) ainsi définis sont entraînés deux à deux à l'aide d'un algorithme considérant une alternance de *mini-batches* propres à une tâche. Chaque tâche est associée à un paramètre  $\alpha$  qui caractérise la proportion d'exemples qu'on souhaite considérer lors de l'apprentissage de cette tâche.

Ils envisagent trois types de structure *Sequence-to-Sequence* multitâche. "One-

"to-many" où un *encoder* est partagé avec plusieurs *decoders* (par exemple, entre une tâche de traduction et une tâche de *parsing*). Réciproquement, "many-to-one" considère le cas d'un même *decoder* partagé entre de plusieurs *encoders* (par exemple, entre un *auto-encoder* et un modèle d'*image captioning*). Enfin, "many-to-many" est le cas général de plusieurs *encoders* et plusieurs *decoders* partagés.

Concernant l'algorithme d'apprentissage multitâche, quelle que soit la structure, les auteurs considèrent l'apprentissage d'une tâche principale, soutenu par l'apprentissage de tâches secondaires. Afin de favoriser cette tâche principale, mais également parce qu'ils considèrent des tâches très diverses pour lesquelles les quantités de données disponibles peuvent drastiquement varier, ils attribuent à chaque tâche un hyper-paramètre  $\alpha$ . Ces hyper-paramètres gèrent explicitement l'alternance des apprentissages entre les tâches. Celui de la tâche principale  $\alpha_1$  est fixé à 1 tandis que les  $\alpha$  associés aux tâches secondaires varient entre 1 et 0.01. Durant l'apprentissage, l'algorithme considère une tâche à la fois. Si on a fixé à  $N$  le nombre d'itérations désirées pour la tâche principale, alors une tâche secondaire  $i$  sera entraînée durant  $\alpha_i N$  itérations. Lorsqu'on bascule entre les tâches, une nouvelle tâche  $i$  est sélectionnée aléatoirement avec une probabilité  $\frac{\alpha_i}{\sum \alpha}$ .

### 3.1.4 Discussion

Il ressort de l'étude de ces travaux qu'un réseau de neurones multitâche entraîné sur  $K$  tâches différentes peut se ramener à  $K$  réseaux de neurones monotâche partageant une partie de leur paramètres. Le fait de partager des paramètres entre différentes tâches permet de mutualiser leur apprentissage. Ces paramètres partagés sont optimisés en vue de produire des représentations qui minimisent l'erreur commise sur l'ensemble des tâches.

Le rôle de la couche d'entrée composée de *Lookup-Tables* est d'encoder sous forme de vecteurs denses les éléments composant les entrées. Partager ces représentations entre différentes tâches revient à considérer qu'on peut apprendre, dans le cadre d'une tâche, quelque chose au sujet d'un élément, qui pourrait nous être utile dans le cadre d'une autre tâche. Ce qu'on mutualise, c'est la capacité du réseau à représenter les éléments constituant ses entrées *via* des espaces de représentation communs à l'ensemble des tâches.

Le rôle de la couche cachée (CNN ou RNN) est de produire une représentation d'une séquence d'éléments de taille variable. Partager ces représentations entre différents tâches, c'est considérer qu'on peut apprendre, dans le cadre d'une tâche, quelque chose au sujet de la manière de combiner les signaux d'entrée

constituant le contexte, qui pourrait être utile dans le cadre d'une autre tâche. Par exemple, s'il y a une *feature* particulièrement pertinente à mémoriser dans l'historique d'une tâche, elle sera aussi disponible pour les autres. Cela va dans le sens d'un modèle linguistique sous-jacent cohérent i.e les représentations, pertinentes pour une sous-tâche, le sont aussi pour les autres.

Le rôle de la couche de sortie (MLP ou RNN) est de produire la sortie attendue, qu'il s'agisse de la distribution de probabilité des classes pour l'entrée considérée, ou bien une séquence générée par un *Decoder*. On considère donc autant de couches de sorties différentes que le problème compte de tâches. Ce qui signifie que les ensembles de paramètres associés seront optimisés indépendamment pour chaque tâche. *A priori*, rien n'interdit de mélanges les architectures. C'est notamment ce que font COLLOBERT et WESTON 2008 en considérant à la fois des tâches de classification à l'aide d'un critère de *cross-entropy* et d'un *softmax*, et des tâches de *ranking* à l'aide d'un *triplet loss* et d'un générateur de sorties négatives artificielles.

## 3.2 Modèle Joint

Lorsqu'on entraîne un modèle d'apprentissage sur une tâche de haut niveau telle que l'analyse syntaxique ou sémantique, on a accès à un corpus d'entraînement annoté selon les niveaux précédents dans la chaîne d'analyse (POS et chunks par exemple). Ces corpus ont généralement été annotés manuellement, ou automatiquement, par un *Parser* par exemple, mais corrigés manuellement. Cependant durant la phase d'évaluation, le modèle repose sur des *features* prédites automatiquement par un analyseur indépendant ce qui peut induire des erreurs. Une solution est de faire du *End-to-End* et donc de se passer de *features* tel que les POS en laissant le modèle apprendre lui-même cette représentation intermédiaire. Nous proposons de suivre cette idée en utilisant le *MTL* afin de superviser la partie "apprentissage de représentation intermédiaire" à l'aide d'une tâche sur les *features* pré-requises.

On pose donc comme hypothèse qu'une représentation apprise sur les tâches intermédiaires contiendra des informations utiles pour la tâche finale. Par exemple, pour faire du *Chunking*, en ajoutant une tâche de *POS-Tagging* on force le réseau à créer des représentations similaires pour les déterminants, ce qui permet de mieux généraliser dans le cas de la détection des frontières de groupes nominaux. De plus, les tâches annexes ne sont pas nécessairement des tâches intermédiaires du TAL, mais pourraient être n'importe quelles annotations manuelles liées aux entrées qui peut aider à la généralisation.

On souhaite combiner des LSTM-BiRNNs monotâche afin de produire une analyse complète suivant plusieurs tâches syntaxiques et sémantiques. En considérant que chaque tâche est associée à un réseau qui lui est propre, nous décrivons une approche d'apprentissage conjoint de ces réseaux à l'aide de paramètres partagés et d'une combinaison de critères d'apprentissage.

### 3.2.1 LSTM-BiRNN

Dans cette sous-section, nous proposons d'utiliser une architecture paramétrique dédiée au traitement des tâches d'analyse linguistique du TAL. Le formalisme utilisé est celui introduit dans le chapitre 1. Le corpus d'apprentissage, noté  $C_{\text{task}}$ , est une collection de phrases annotées. Une phrase annotée est une séquence de couples  $(\text{mot}, \text{analyse})^{t=1..T}$  au sein de laquelle chaque occurrence d'un mot est associée à son analyse dans le contexte courant. Un mot est représenté par un vecteur de *features*  $x^t$  qui est une description de l'ensemble de ses caractéristiques prises en compte (forme morphologique, lemme, *etc.*). L'analyse associée à une occurrence est représentée comme la classe attendue  $y^t$  (classe grammaticale, gouverneur syntaxique, *etc.*). On souhaite entraîner un modèle qui associe une séquence de classes de sortie  $Y = y^{t=1..T}$  à une séquence de vecteurs de *features* d'entrée  $X = x^{t=1..T}$ .

$$C_{\text{task}} = (X, Y) = (x^t, y^t)_{t=1..T} \quad (3.2)$$

avec  $\forall t x^t \in \mathcal{X}$  et  $\forall t y^t \in \mathcal{Y} \subset \mathbb{N}$

On se base sur un réseau récurrent qui peut être décrit comme une fonction paramétrique  $\mathcal{F}_\theta$ . Cette fonction prend en entrée une séquence de taille arbitraire et produit une séquence de même taille. On souhaite apprendre au réseau à prédire la séquence de classes  $Y$  associée à une séquence de mots  $X$ . Autrement dit, on souhaite déterminer les valeurs des paramètres  $\theta$  de cette fonction afin que l'image par  $\mathcal{F}_\theta$  d'une occurrence observée corresponde à l'analyse attendue *i.e.*  $\forall (X, Y) \in C_{\text{task}} \mathcal{F}_\theta(X) = Y$ . Les paramètres  $\theta$  forment un réseau de neurones récurrents. Il se compose d'une couche d'encodage chargée de la projection des *features*  $x$  dans un espace de représentation, d'une couche cachée récurrente dont le rôle est d'enrichir la représentation d'une occurrence à l'aide de son contexte et d'une couche de décodage responsable de la prédiction des sorties attendues  $y$  par classification.

### 3.2.1.1 Couche d'entrée : Encodage

On considère qu'un exemple d'apprentissage est une phrase annotée *i.e* une séquence d'occurrences  $x^t$ , chacune associée à son analyse attendue  $y^t$ . Chaque occurrence est représentée par un vecteur de *features* composé de  $N$  éléments formant un tuple  $x^t = (x_1^t, \dots, x_N^t) \in \mathcal{X}_1 \times \dots \times \mathcal{X}_N$ . Dans cette description,  $\mathcal{X}_n$  fait référence à l'ensemble des valeurs pouvant être prises par la  $n^{\text{eme}}$  *feature*. On parlera de lexiques associés aux *features* <sup>c</sup>.

La couche d'entrée d'un réseau de neurones récurrent a pour but de transformer mot-à-mot une phrase d'entrée  $(x^1, \dots, x^T)$  composée de  $T$  mots, en une séquence  $\mathbf{e}$  de vecteurs de dimension  $d$  via l'application mot à mot de  $W_{\text{INPUT}}$ . Nous utilisons la couche d'entrée linéaire définie dans la section 2.4.4.2. Cette implémentation consiste à utiliser des tables d'association (*Look-up Tables*) afin de réaliser une *embedding layer* sans avoir à réaliser le produit de la représentation *one-hot*  $\mathbf{x}^t \in \mathbb{R}^{|\mathcal{X}|}$  par la matrice d'*embedding*  $W_{\text{INPUT}} \in \mathbb{R}^{d \times |\mathcal{X}|}$ .

$$\begin{aligned} \mathbf{e}^t &= \mathcal{F}_{\text{INPUT}}(\mathbf{x}^t) = W_{\text{INPUT}} \cdot \mathbf{x}^t \\ &= (\mathbf{w}_1 \dots \mathbf{w}_{x^t} \dots \mathbf{w}_{|\mathcal{X}|}) \cdot (0 \dots 1 \dots 0) \\ &= \mathbf{w}_{x^t} \in \mathbb{R}^d \end{aligned} \quad (3.3)$$

Afin de prendre en compte les spécificités linguistiques d'une tâche, on peut tirer parti d'un vecteur de *features* pour caractériser les mots (lemme, morphologie, annotations syntaxiques ou sémantiques résultant d'analyses préliminaires, *etc.*). Chaque mot est associé à des *features* descriptives qu'on souhaiterait prendre en compte dans l'analyse. On généralise le processus d'encodage de la couche d'entrée en considérant qu'un mot  $x^t$  peut se décomposer en  $N$  éléments formant un tuple  $x^t = (x_1^t, \dots, x_N^t) \in \mathcal{X}_1 \times \dots \times \mathcal{X}_N$  où  $\mathcal{X}_n$  correspond au lexique associé à la  $n^{\text{eme}}$  *feature*. Soit  $X = (x_1^t, \dots, x_N^t)_{t=1..T}$  avec  $x^t \in \mathcal{X}_1 \times \dots \times \mathcal{X}_N$ . On utilise  $N$  *Look-Up Tables* afin d'encoder chaque *feature* individuellement. Chaque *Look-Up Table* peut être représentée comme une application linéaire  $W_{\text{INPUT}_n} \in \mathbb{R}^{d_n \times |\mathcal{X}_n|}$  permettant de projeter individuellement chaque *feature*  $\mathbf{x}_n^t$  dans un espace de dimension respective  $d_n$ . Ces nouvelles représentations denses ainsi obtenues sont ensuite concaténées afin de former la représentation  $\mathbf{e}^t$  du vecteur d'entrée  $\mathbf{x}^t$ .

$$\mathbf{e}^t = \mathcal{F}_{\text{INPUT}}(\mathbf{x}^t) = W_{\text{INPUT}_1} \cdot \mathbf{x}_1^t \bullet \dots \bullet W_{\text{INPUT}_N} \cdot \mathbf{x}_N^t \in \mathbb{R}^d \quad (3.4)$$

---

c. De façon analogue, les analyses attendues sont ramenées à des annotations, syntaxiques ou sémantiques en fonction de la tâche d'intérêt, appartenant à un lexique  $\mathcal{Y}$ .

On note  $\theta_{\text{INPUT}} = \{\forall n, W_{\text{INPUT}_n} \in \mathbb{R}^{d_n \times |\mathcal{X}_n|}\}$  l'ensemble des paramètres internes de notre couche d'entrée associant une matrice d'encodage spécifique à chacun des  $N$  éléments du vecteur d'entrée. Un mot  $x^t$  est ainsi représenté comme la concaténation des représentations vectorielles de ses *features* associées. Les dimensions respectives  $d_n$  de ces représentations sont autant d'hyper-paramètres. En posant  $\sum d_n = d$ , on retombe sur une séquence encodée  $\mathbf{e} \in \mathbb{R}^{T \times d}$ . Cette couche d'entrée généralisée transforme donc une séquence de mots annotés en une séquence de vecteurs de  $\mathbb{R}^d$ .

### 3.2.1.2 Couche cachée : Mise en contexte

On cherche à réaliser une classification de séquences de mots, c'est-à-dire qu'on souhaite prédire la séquence de classes  $(y^t)_{t=1..T}$  associée à une séquence d'entrée  $(x^t)_{t=1..T}$ . Lorsqu'une entrée  $x^t$  est présentée au réseau, on souhaite que son analyse dépende des vecteurs de *features* précédents ( $x^{t-1}$ ,  $x^{t-2}$ , etc.) et suivants ( $x^{t+1}$ ,  $x^{t+2}$ , etc.) l'entrée courante. On cherche ainsi à prendre en compte les contextes passés et futurs d'une occurrence car dans le cas du traitement automatique de la langue, il est courant que l'analyse d'un mot soit déterminée par le mot suivant. Par exemple l'analyse morpho-syntaxique du mot "lit" dans la phrase "Le petit lit du Rousseau" est ambiguë si on a pas accès au contexte "du Rousseau". Pour ce faire, on a choisi d'utiliser un réseau récurrent bidirectionnel (cf. section 1.4.2.2). Cette couche cachée prend en entrée la séquence de vecteurs  $\mathbf{e} \in \mathbb{R}^{T \times d}$  produite par la couche d'entrée.

$$\vec{\mathbf{h}}^t = \vec{\mathcal{F}}_{\text{HIDDEN}}(\mathbf{e}^t, \vec{\mathbf{h}}^{t-1}) \in \mathbb{R}^{\vec{h}} \quad (3.5)$$

$$\overleftarrow{\mathbf{h}}^t = \overleftarrow{\mathcal{F}}_{\text{HIDDEN}}(\mathbf{e}^t, \overleftarrow{\mathbf{h}}^{t+1}) \in \mathbb{R}^{\overleftarrow{h}} \quad (3.6)$$

$$\mathbf{h}^t = \mathcal{F}_{\text{HIDDEN}}(\vec{\mathbf{h}}^t, \overleftarrow{\mathbf{h}}^t) \in \mathbb{R}^h \quad (3.7)$$

Le principe fondamental des réseaux récurrents bidirectionnels est de présenter chaque séquence de *gauche à droite* et de *droite à gauche* à deux RNN distincts. Les *historiques* des deux RNN à un instant  $t$  sont concaténés et combinés par une couche linéaire. Cela signifie qu'en tout point d'une séquence donnée, le BiRNN a accès à une agrégation de l'information séquentielle en aval et en amont. On note respectivement  $h$ ,  $\vec{h}$  et  $\overleftarrow{h}$  les hyper-paramètres de la couche cachée caractérisant les dimensions des espaces vectoriels alloués à la représentation de l'historique.

Afin de permettre au réseau de gérer efficacement des séquences d'entrée de longueur importante, il faut passer outre les problèmes de *vanishing gradient*. On a pour ce faire recourt aux cellules LSTM (cf. section 1.4.2.1). Elles tirent profit d'un système de portes (les *gates*) permettant au modèle de gérer explicitement la mise à jour et la remise à zéro d'un état interne (cf. équation 1.25). Ces cellules sont ainsi capables de capturer des dépendances "longue distance" *i.e* entre deux mots particulièrement distants dans la séquence d'entrée, grâce à la conservation de leur état interne  $\mathbf{c}$ . Soit  $h$  la dimension de l'espace alloué à l'état interne  $\mathbf{c}^t \in \mathbb{R}^h$ , alors les *gates* sont des vecteurs de  $[0, 1]^h$  (cf. fonction sigmoïde). Les *gates* peuvent être interprétées comme des facteurs d'atténuation des entrées, de l'état interne et des sorties. Dans le cas de leur application dans un réseau bidirectionnel, il est nécessaire de modifier les équations 1.25 de la branche *backward* en considérant  $\mathbf{h}^{t+1}$  en lieu et place de  $\mathbf{h}^{t-1}$ . Une alternative consiste à considérer des séquences d'entrée renversées  $\mathbf{e}_r^t = \mathbf{e}^{T-t}$  en prenant soin de renverser également les sorties obtenues  $\mathbf{h}^t = \mathbf{h}_r^{T-t}$ .

$$\mathbf{h}^t = \mathcal{F}_{\text{HIDDEN}} \left( \vec{\mathbf{h}}^t, \overleftarrow{\mathbf{h}}^t \right) = f \left( W_{\text{HIDDEN}} \cdot (\vec{\mathbf{h}}^t \bullet \overleftarrow{\mathbf{h}}^t) + b_{\text{HIDDEN}} \right) \quad (3.8)$$

L'équation 3.8, relie l'activation  $\mathbf{h}^t$  de la couche cachée aux activations des RNN *forward* et *backward*. Dans cette équation,  $f(\cdot)$  symbolise une fonction d'activation ReLU (Unité Rectifié linéaire) à valeur dans  $\mathbb{R}^+$  suivie d'un *DropOut* de probabilité d'extinction  $P_{DO}$ . Cette couche récurrente produit ainsi pour chaque entrée  $\mathbf{e}^t$  une représentation  $\mathbf{h}^t$  issue des historiques *forward*  $\vec{\mathbf{h}}^t$  et *backward*  $\overleftarrow{\mathbf{h}}^t$ . On souhaite permettre au réseau d'accéder, à tout instant, à la totalité de l'information séquentielle  $\mathbf{h}^{t \pm \tau}$ , plutôt que de le forcer à encoder cette information dans un *historique* de dimension fixe. Pour ce faire, on surmonte le réseau d'un mécanisme d'attention (cf. section 1.4.2.3). La sortie  $\mathbf{s}^t$  de la couche cachée est alors une combinaison linéaire des états cachés (cf. équation 3.9) et non plus exclusivement l'état caché courant  $\mathbf{h}^t$ .

$$\mathbf{s}^t = \sum_{i=1}^T a_i^t \mathbf{h}^i \quad (3.9)$$

On rappelle que  $h$ ,  $\vec{h}$  et  $\overleftarrow{h}$  sont des hyper-paramètres de la couche cachée caractérisant les dimensions des espaces vectoriels alloués à la représentation de l'historique. En découlent les dimensions des matrices de paramètres et des biais formant l'ensemble  $\theta_{\text{RNN}} = \{ \vec{\theta}_{\text{LSTM}}, \overleftarrow{\theta}_{\text{LSTM}}, \theta_{\text{HIDDEN}}, \theta_{\mathcal{A}} \}$  des paramètres de la couche cachée récurrente :

$$\begin{array}{ll}
\vec{\theta}_{\text{LSTM}} : & \overleftarrow{\theta}_{\text{LSTM}} : \\
\text{— } \vec{W}_i, \vec{W}_f, \vec{W}_o \in \mathbb{R}^{\vec{h} \times (d+2\vec{h})}; & \text{— } \overleftarrow{W}_i, \overleftarrow{W}_f, \overleftarrow{W}_o \in \mathbb{R}^{\overleftarrow{h} \times (d+2\overleftarrow{h})}; \\
\text{— } \vec{W}_c \in \mathbb{R}^{\vec{h} \times (d+\vec{h})}; & \text{— } \overleftarrow{W}_c \in \mathbb{R}^{\overleftarrow{h} \times (d+\overleftarrow{h})}; \\
\text{— } \vec{b}_i, \vec{b}_f, \vec{b}_o, \vec{b}_c \in \mathbb{R}^{\vec{h}}. & \text{— } \overleftarrow{b}_i, \overleftarrow{b}_f, \overleftarrow{b}_o, \overleftarrow{b}_c \in \mathbb{R}^{\overleftarrow{h}}. \\
\theta_{\text{HIDDEN}} : & \theta_{\mathcal{A}} : \\
\text{— } W_{\text{HIDDEN}} \in \mathbb{R}^{h \times (\vec{h} + \overleftarrow{h})}; & \text{— } W_{\mathcal{A}_1}, W_{\mathcal{A}_2} \in \mathbb{R}^{A \times h}; \\
\text{— } b_{\text{HIDDEN}} \in \mathbb{R}^h; & \text{— } b_{\mathcal{A}} \in \mathbb{R}^A; \\
\text{— } P_{DO} \in [0, 1]. & \text{— } \mathbf{v}_{\mathcal{A}} \in \mathbb{R}^A.
\end{array}$$

### 3.2.1.3 Couche de sortie : Prédiction

Dans la section 2.3, on a montré comment décomposer les analyses incontournables du TAL en combinaisons d'opération d'étiquetage, de segmentation et de détection de relations. Afin de rendre synchrones les prédictions sur l'ensemble des tâches, nous avons choisi de définir des cibles à chaque instant et pour chaque tâche (ce qui n'est pas la formalisation habituelle de toutes les tâches de TAL considérées). Cette formalisation passe essentiellement par la définition des cibles attendues. Pour les tâches d'étiquetage, on associe une classe à chacune des étiquettes possibles parmi une liste prédéterminée  $\mathcal{Y}$ . Pour les tâches de segmentation, on se restreint à prédire si un mot correspond au début (B), à l'intérieur (I) ou à la fin (E) d'un segment, ou bien si au contraire il n'appartient à aucun des constituants complexes qu'on cherche à détecter (O). On ramène ainsi la segmentation à un problème de classification où  $|\mathcal{Y}| = 4$  classes possibles. Si les segments sont annotés, on considérera alors l'annotation issue du produit cartésien des classes de segmentation et des classes d'étiquetage. La détection de relation peut également être traitée comme un problème de classification où on cherche à prédire, pour chaque mot, l'indice  $i$  du mot auquel il se rapporte (ou alternativement la distance qui l'en sépare). On va donc chercher à prédire  $p(i | x^t)$  la distribution de probabilité d'existence d'une relation de  $x^t$  vers  $x^i$ .

Considérons un ensemble de  $K$  réseaux paramétrés indépendamment. On souhaite combiner l'ensemble de ces réseaux afin de réaliser une analyse conjointe et de mutualiser l'apprentissage sur l'ensemble des tâches d'intérêt. DONG, H. WU, HE et al. 2015 considèrent le partage d'un BiRNN entre les tâches afin de produire des représentations cachées communes. Ils profitent ainsi du fait que l'ensemble des tâches de traduction ont la même langue source, ce qui uniformise la forme des entrées entre les différentes instances. Nous avons choisi de nous situer dans la lignée de ces travaux et de ne différencier d'une tâche à l'autre

que les paramètres de la couche de sortie, dont les équations se réécrivent :

$$\mathbf{z}_k^t = \mathcal{F}_{\text{OUTPUT}_k}(\mathbf{h}^t, \mathbf{s}^t) = W_{\text{OUTPUT}_k} \cdot (\mathbf{h}^t \bullet \mathbf{s}^t) + b_{\text{OUTPUT}_k} \in \mathbb{R}^{|\mathcal{Y}_k|} \quad (3.10)$$

$$\mathbf{p}_k^t = \underset{i=1..|\mathcal{Y}_k|}{\text{softmax}}(\mathbf{z}_k^t) \in [0, 1]^{|\mathcal{Y}_k|} \quad (3.11)$$

$$p(i \in \mathcal{Y}_k \mid \mathbf{x}^t, X, \theta, \theta_k) = \mathbf{p}_{k,i}^t \quad (3.12)$$

$$\mathcal{L}_k = \sum_{(\mathbf{x}^t, \mathbf{y}_k^t) \in C_{\text{task}}} \log p(\mathbf{y}_k^t \mid \mathbf{x}^t, X, \theta, \theta_k) \quad (3.13)$$

Pour chaque élément  $\mathbf{x}^t$  de la séquence d'entrée, on calcule  $K$  vecteurs  $\mathbf{z}_{k=1..K}^t$  à l'aide de couches linéaires respectivement associées aux  $K$  tâches. L'activation  $\mathbf{z}_{k,n}^t$  permet de prédire la probabilité que  $\mathbf{x}^t$  appartienne à la classe d'indice  $n \in |\mathcal{Y}_k|$ . Ces couches de sortie spécifiques prennent toutes en entrée les mêmes représentations  $\mathbf{h}^t \bullet \mathbf{s}^t$  issues de la couche cachée et du modèle d'attention partagés. Ainsi, les autres paramètres de la couche d'entrée et de la couche cachée  $\theta = \{\theta_{\text{INPUT}}, \theta_{\text{RNN}}, \theta_{\mathcal{A}}\}$  sont communs à l'ensemble des réseaux. Tandis que les paramètres associés aux couches de sortie  $\theta_k = \{W_{\text{OUTPUT}_k}, b_{\text{OUTPUT}_k}\}$  sont propres à chaque tâche  $k$ . Chaque couche de sortie possède son propre critère d'apprentissage en log-vraisemblance  $\mathcal{L}$  (cf. section 1.3.2.1).

### 3.2.2 Critère d'apprentissage global

Maintenant que nous avons défini une architecture commune, il faut déterminer la manière dont elle sera entraînée conjointement sur l'ensemble des tâches. Dans un premier temps, on peut simplement considérer qu'il s'agit d'un ensemble de réseaux partageant une partie de leurs paramètres internes. Une approche proposée consiste ainsi à entraîner individuellement chacun de ces réseaux. L'optimisation par propagation de l'erreur commise pour une tâche, modifie donc les paramètres de sa couche de décodage spécifique ainsi que ceux des couches communes à l'ensemble des réseaux. On produit ainsi un mécanisme d'encodage des données commun à l'ensemble des tâches.

L'heuristique considérée dans COLLOBERT et WESTON 2008, DONG, H. WU, HE et al. 2015 et LUONG, LE, Ilya SUTSKEVER et al. 2015 afin d'entraîner ce type d'architecture à poids partagés, est constituée d'une alternance des tâches par tirage au sort. Cet algorithme considère que chaque tâche est associée à un corpus différent. On sélectionne une tâche, puis un *batch* d'exemples propres à cette tâche et on entraîne le réseau associé sur ce *batch*. Chaque réseau est ainsi entraîné

individuellement mais tout en étant indirectement modifié par l'apprentissage des autres tâches. Ce faisant, on espère que l'ensemble de ces optimisations spécifiques finisse par converger afin de produire un paramétrage permettant la réalisation simultanée de toutes les tâches.

On souhaite profiter du fait qu'on a accès à un corpus annoté sur l'ensemble des tâches. C'est la raison pour laquelle on peut s'abstraire de l'apprentissage par alternance proposée par les méthodes précédentes, afin de réaliser simultanément les prédictions et ainsi mutualiser les calculs et les représentations sur la partie commune des réseaux. On peut pour cela combiner les critères afin de définir explicitement un critère d'erreur global qui sera minimisé par l'algorithme de rétro-propagation. Ainsi, cet apprentissage peut se résumer comme l'optimisation d'une fonction de coût globale, combinaison des fonctions de coût spécifiques. On définit un critère d'apprentissage  $\mathcal{L}_{global}$ , combinaison linéaire des critères individuels  $\mathcal{L}_k$  :

$$\mathcal{L}_{global} = \sum_{k=1}^K \beta_k \mathcal{L}_k = \sum_{k=1}^K \beta_k \sum_{(x^t, y_k^t) \in C_{task}} \log p(y_k^t | x^t, X, \theta, \theta_k) \quad (3.14)$$

avec  $\sum \beta_k = 1$ , et où  $\theta$  contient l'ensemble des paramètres partagés par l'ensemble des tâches et  $\theta_k$  l'ensemble des paramètres spécifiques à la  $k^{ieme}$  tâche d'intérêt. Les pondérations  $\beta$  sont des hyper-paramètres déterminant l'importance relative que le modèle apportera à l'apprentissage de chaque tâche.  $\beta_k$  détermine la part de l'erreur relative  $\mathcal{L}_k$  dans l'erreur globale  $\mathcal{L}_{global}$ , et donc la prise en compte de  $\mathcal{L}_k$  dans l'optimisation des paramètres joints  $\theta$  du modèle. Il est nécessaire de pondérer l'intérêt qu'on porte à chaque tâche durant l'apprentissage car d'une part, certaines analyses sont plus faciles (moins ambiguës) et d'autre part, la distribution des classes peut-être très variable d'une tâche à l'autre.

### 3.2.3 Apprentissage par tronçon

Le modèle décrit précédemment n'est pas dédié spécifiquement à une tâche d'analyse en particulier. Il est au contraire voué à être applicable dans un grand nombre de tâches du TAL et doit être capable de lever des ambiguïtés ou de détecter des relations *longue distance*, pouvant potentiellement sortir du cadre de la phrase. Afin de nous abstraire de l'hypothèse d'indépendance des phrases (cf. section 1.4.4), on souhaiterait entraîner un réseau récurrent qui considérerait le corpus d'apprentissage  $C_{task}$  comme une seule et même séquence d'entrée. Les limitations matérielles (mémoire, puissance de calcul) bornent la taille des réseaux qu'on peut effectivement allouer et entraîner en parallèle en un temps

raisonnable. Dans ce qui suit, on note cette taille  $\tau$  qu'on estimera comme petite devant la taille  $T$  du corpus mais grande vis-à-vis de la taille des phrases. On cherche à prédire la même séquence que le réseau *total* prenant l'intégralité du corpus en entrée, uniquement à l'aide de réseaux prenant en entrée des sous-séquences de taille  $\tau$ .

Afin d'obtenir rigoureusement les mêmes sorties sur une sous-séquence, il suffit que chacun des nœuds  $t$  traitant cette sous-séquence reçoive les mêmes entrées  $(x^t, h^{t-1})$  que dans le cas du réseau *total*. Si c'est trivial dans le cas des  $x^t$ , il est nécessaire d'apporter une attention particulière à la gestion des historiques. En effet, on peut montrer que dans le cas d'une séquence de mot  $x^{i..j}$ , la prédiction réalisée par un tronçon est la même que celle prédite par le réseau *total* si et seulement si l'historique  $\vec{h}^{i-1}$  est initialisé à l'aide de celui obtenu par une autre instance du réseau dans lequel on a injecté les mots précédents  $x^{0..i-1}$ . Une solution consiste donc à ordonner les sous-séquences et à initialiser l'historique initial  $\vec{h}^{i-1}$  de chaque tronçon en copiant l'historique calculé  $\vec{h}^j$  sur la sous-séquence précédente. Cependant, la conservation des historiques d'un tronçon à l'autre par copie est impossible avec un BiRNN. À cause de problèmes de causalité, il n'est pas possible d'ordonner l'analyse des sous-séquences d'exemples d'apprentissage de telle sorte qu'on soit assuré de posséder des représentations de  $\vec{h}^i$  et  $\overleftarrow{h}^{j+1}$  pour toute sous-séquence.

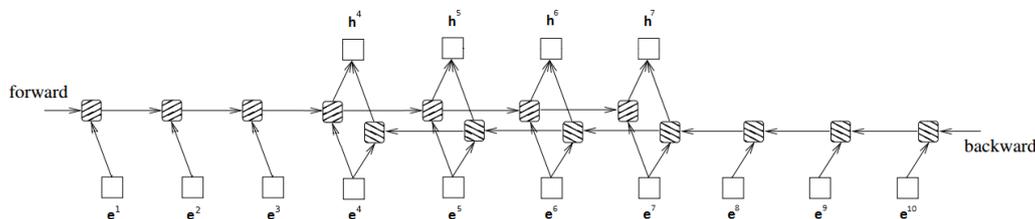


Figure 3.4 – *Scoped Back Propagation Through Time* avec  $\tau = 10$  et  $s = 3$ .

On réalise une approximation à la volée des historiques initiaux. Pour ce faire, on considère des entrées  $\mathbf{x}^{1..s}$ . On concentre l'apprentissage sur une portion centrale de la sous-séquence et on munit le réseau d'une forme de *vision périphérique*. Les structures périphériques<sup>d</sup> servent uniquement à approximer  $\vec{h}^s$  et  $\overleftarrow{h}^{\tau-s}$ . On minimise uniquement l'erreur commise par les nœuds  $s+1$  à  $\tau-s-1$ . On se refuse de corriger les erreurs de prédictions commises dans la région périphérique car on estime que les représentations des historiques n'y sont pas assez bonnes pour prêter attention aux décisions qu'on prendrait uniquement à cause d'elles. En revanche, les paramètres de ces nœuds périphériques sont mis à jour

d. les nœuds 1 à  $s$  d'une part, et  $\tau-s$  à  $\tau$  d'autre part

afin de fournir des représentations des historiques permettant de minimiser les erreurs de classification dans la portion centrale.

---

**Algorithm 7** BackPropagation Through Time with Scope

---

**Require:** Function  $\mathcal{F}_\theta(\mathbf{x})$  parameterized with shared parameters  $\theta^1, \dots, \theta^\tau$

**Require:** Training sequence  $C_{\text{task}}$  input  $\mathbf{x}^{1..T}$  and output  $\mathbf{y}^{1..T}$

**Require:** Loss function  $\mathcal{L}$

```

while stopping criteria not met do
  Sample a training sequence  $\mathbf{x}^{k+1..k+\tau}, \mathbf{y}^{k+1..k+\tau}$ 
   $\hat{\mathbf{g}} \leftarrow 0$ 
   $\overrightarrow{\mathbf{h}}^k \leftarrow 0$ 
   $\overleftarrow{\mathbf{h}}^{k+\tau+1} \leftarrow 0$ 
  for  $t = 1 + s$  to  $\tau - s - 1$  do
    Compute the loss  $\mathcal{L}(\mathcal{F}_\theta(\mathbf{x}^{k+t}), \mathbf{y}^{k+t})$ 
    for  $i = 1$  to  $\tau$  do
       $\hat{\mathbf{g}} \leftarrow \hat{\mathbf{g}} + \frac{1}{\tau-2s} \frac{1}{\tau} \frac{\partial \mathcal{L}(\mathcal{F}_\theta(\mathbf{x}^{k+t}), \mathbf{y}^{k+t})}{\partial \theta^i}$ 
    end for
  end for
   $\theta \leftarrow \theta + \varepsilon \hat{\mathbf{g}}$ 
end while
return  $\theta$ 

```

---

Si  $s$  est suffisamment grand, les historiques initiaux  $\overrightarrow{\mathbf{h}}^s$  et  $\overleftarrow{\mathbf{h}}^{\tau-s}$  ont des valeurs très proches de celles qu'ils auraient eu dans le cas du réseau déplié sur l'intégralité du corpus. En revanche, lors de la phase de correction, le gradient ne peut pas se propager plus loin qu'un tronçon. Tout se passe donc comme si on entraînait le réseau *total* mais en tronquant la propagation du gradient. Intuitivement, une différence notable avec l'apprentissage "phrase-par-phrase" est que le réseau entraîné par tronçon semble capable d'optimiser ses paramètres internes afin de détecter des dépendances de longue distance (mais bornée par  $\tau$ ). Une analyse d'erreur sur la longueur des dépendances, appliquée par exemple à la résolution d'anaphores pronominales où le pronom est particulièrement éloigné du nom auquel il se rapporte, permettrait d'étudier en détail ce phénomène.

### 3.3 Applications

Cette section est dédiée à l'évaluation du modèle décrit dans la section précédente. Trois tâches de l'analyse syntaxique sont considérées : l'étiquetage morpho-syntaxique, l'analyse syntaxique de surface et l'analyse syntaxique en dépen-

dance. L'ensemble de ces tâches sera évalué sur des corpus de référence composés de textes journalistiques anglais afin de comparer les performances obtenues par nos modèles vis-à-vis des approches standards publiées.

Cependant, l'ensemble de ces approches requiert des données annotées. Ces annotations, lorsqu'elles sont réalisées manuellement, sont coûteuses et réclament la participation d'un expert. Afin d'estimer les capacités de généralisation de notre modèle lorsque peu de données sont disponibles, une étude de leurs performances, lorsqu'on fait varier la taille du corpus d'entraînement, est systématiquement réalisée. On espère montrer que l'utilisation de *word embeddings* et de *features* morphologique permet de pallier en partie le manque de données annotées en fournissant de meilleures représentations des mots.

## Cadre Expérimental

Dans nos expérimentations, nous utilisons des *word embeddings* de dimension 300 calculés à partir du *GigaWord* (GRAFF, KONG, CHEN et al. 2003) annoté en dépendances syntaxiques et grâce à *Word2Vec*<sup>e</sup> (LEVY et GOLDBERG 2014). Nous avons considéré en tant que *features* morphologiques un sac des *bi-grammes* de caractères non-consécutifs présents dans le mot d'intérêt. Ainsi, "petit" est représenté par { *pe*, *pt*, *pi*, *et*, *ei*, *ti*, *tt*, *it* }.

Nos BiRNN sont composés chacun de  $2 \times 512$  cellules LSTM avec une probabilité de *DropOut* fixée à 0,5. Ils ont été entraînés sur des *mini-batches* de 128 sous-séquences composées de 256 mots. La rétro-propagation est restreinte au 128 mots centraux. La couche d'entrée contenant les *word embeddings* est régularisée sur ses valeurs initiales avec un facteur  $\lambda = 0,001$ . Nous avons envisagé plusieurs distributions  $\beta$  du critère d'apprentissage global. Une distribution uniforme ( $\forall k \beta_k = \frac{1}{K}$ ) et des distributions favorisant une tâche en particulier ( $\beta_n \gg \beta_k \forall k \neq n$ ).

### 3.3.1 Étiquetage morpho-syntaxique

Une tâche de référence du Traitement Automatique de la Langue est l'étiquetage morpho-syntaxique (ou *POS tagging*) du Penn TreeBank. Ce corpus journalistique est composé d'environ un million de mots annotés en partie de discours. Un grand nombre de modèles ont été évalués à l'aide de ce corpus comme

---

e. <https://bitbucket.org/yoavgo/word2vecf>

l'illustre le tableau 3.1. Seuls sont cités les approches les plus récentes et efficaces : le *classifier* par maximum d'entropie de C. MANNING 2011, l'approche basée sur un *Conditional Random Field* de SUN 2014, le réseau de neurones de COLLOBERT, WESTON, BOTTOU et al. 2011 et les architectures profondes bidirectionnelles de Z. HUANG, XU et K. YU 2015.

Model	PWA POS
C. MANNING 2011	97.32
COLLOBERT, WESTON, BOTTOU et al. 2011	97.37
SUN 2014	97.33
Z. HUANG, XU et K. YU 2015	97.55
<b>E2E :BiRNN</b>	97.62

Table 3.1 – Évaluation sur l'étiquetage morpho-syntaxique sur le Penn TreeBank.

Dans ce cas d'utilisation, notre modèle **E2E :BiRNN** correspond à un seul BiRNN prédisant les *POS-tags* à partir des mots. Il semble tout aussi efficient sur ce corpus que les méthodes état de l'art malgré l'absence de modèle d'optimisation globale des séquences de prédictions. Afin de jauger les capacités de généralisation de notre modèle, nous avons étudié l'impact de la quantité de données d'apprentissage et des *features* spécifiques sur ses performances. Les résultats obtenus sont présentés dans le tableau 3.2. On remarque ainsi que le BiRNN prédisant les POS tags directement à partir des mots, offre des performances acceptables au vu de sa très faible spécificité. En revanche, les *features* morphologiques sont essentielles à l'obtention de performances état de l'art. Nous avons uniquement considéré un encodage des *bi-grammes* de caractères présents dans le mot d'intérêt, afin de minimiser l'impact de l'expertise linguistique.

Data (%)	10%	20%	40%	60%	80%	100%
<b>E2E :BiRNN</b>	92.9	94.6	95.7	96.5	96.8	97.18
+ Embedding	94.1	95.7	95.9	96.6	96.7	97.21
+ Morph	95.2	96.3	96.8	97.3	97.3	97.62

Table 3.2 – Évaluation sur l'étiquetage morpho-syntaxique en fonction du volume de l'ensemble d'apprentissage sur le Penn TreeBank.

L'initialisation de la couche d'encodage  $W_{\text{INPUT}}$  à l'aide de *word embeddings* offre un gain substantiel dans le cas où peu de données d'apprentissage sont disponibles (jusqu'à environ 20% soit 200 000 mots). Il est important de noter que ces représentations doivent alors être fortement régularisées<sup>f</sup> sur leur

f. avec un facteur de régularisation L2  $\lambda = 0,001$

position d'origine afin d'éviter que des effets de sur-apprentissage n'écrasent l'initialisation de l'espace d'*embedding* obtenu par apprentissage non-supervisé (cf. 1.3.2.4). Si cette régularisation n'est pas appliquée, le gain de performances observé s'estompe à chaque itération jusqu'à disparaître.

Cependant, on ne sait pas si ce gain de performances, grâce aux *word embeddings*, observé lorsqu'on diminue la quantité de données dédiée à l'apprentissage, provient d'une meilleure représentation initiale de l'ensemble des mots, inatteignable par optimisation sur aussi peu de données, ou bien si ils sont dus à l'augmentation des proportions de mots absents du corpus d'apprentissage, dans le corpus d'évaluation. En effet, les mots qui profitent le plus de l'initialisation de la couche d'encodage à l'aide de représentations distributionnelles sont les mots inconnus. Cette forme de *pré-entraînement* permet de fournir une *meilleure*<sup>g</sup> représentation d'un mot absent du corpus d'apprentissage. Il serait donc bon d'étudier, d'une part la distribution des mots inconnus en fonction du volume du corpus alloué à l'apprentissage, et d'autre part la distribution du gain dû aux *word embeddings* sur ces différentes distributions de mots, connus comme inconnus.

### 3.3.2 Analyse syntaxique de surface

Une autre tâche de référence du Traitement Automatique de la Langue est l'analyse syntaxique surfacique (ou *Chunking*) du CoNLL 2000<sup>h</sup>. Ce corpus est composé de 200 000 mots annotés en partie de discours (nom, verbe, etc.) et en constituants (groupes nominaux, verbaux, etc.). Le tableau 3.3 référence les méthodes les plus efficaces à ce jour pour résoudre ce problème : l'approche *Conditional Random Field* proposée par SHA et F. PEREIRA 2003, la combinaison de *classifiers* entraînés sur différentes représentations des données à l'aide d'un grand nombre de *features* empiriques de SHEN et SARKAR 2005, le réseau de neurones de COLLOBERT, WESTON, BOTTOU et al. 2011 et les architectures profondes et bidirectionnelles de Z. HUANG, XU et K. YU 2015.

**E2E:BiRNN** correspond au même modèle que pour la tâche précédente, mais cette fois entraîné à prédire les étiquettes de *Chunking* à partir des mots. Il s'agit de notre modèle *baseline* de référence. Le pipeline de réseaux **PL:BiRNN** et notre réseau multitâche **MT:BiRNN** obtiennent des résultats comparables à ceux proposés dans l'état de l'art. Ces performances sont supérieures à celles des autres modèles également entraînés avec le minimum d'*a priori* linguistiques. **PL:MT:BiRNN** correspond à un pipeline de réseau multitâche. Lorsqu'on fournit

---

g. *meilleure* vis-à-vis d'une représentation initialisée aléatoirement.

h. Corpus disponible <http://www.cnts.ua.ac.be/conll2000/chunking/>

Model	F1 Chunk
SHA et F. PEREIRA 2003	94.38
SHEN et SARKAR 2005	95.23
COLLOBERT, WESTON, BOTTOU et al. 2011	94.10
Z. HUANG, XU et K. YU 2015	94.46
<b>E2E :BiRNN</b>	94.55
<b>PL :BiRNN</b>	94.79
<b>MT :BiRNN</b>	94.95
<b>PL :MT :BiRNN</b>	95.18

Table 3.3 – Évaluation sur l’analyse syntaxique de surface sur le CoNLL 2000.

en entrée du **MT :BiRNN** les *features* morpho-syntaxiques<sup>i</sup>, tout en entraînant le réseau à prédire les *POS-tags* de références en parallèle du *Chunking*, on obtient des performances comparables à l’approche de SHEN et SARKAR 2005. L’ensemble de ces modèles disposent de *features* morphologiques et d’une couche d’entrée initialisée à l’aide de *word embeddings* dont l’impact est étudié dans le tableau 3.4.

Data (%)	20%	40%	60%	80%	100%
<b>E2E :BiRNN</b>	84.1	90.6	92.5	93.2	94.17
+ Embedding	85.5	90.9	92.6	93.2	94.20
+ Morph	86.5	91.5	93.1	93.8	94.55
<b>MT :BiRNN</b>	84.9	90.6	93.3	94.0	94.66
+ Embedding	85.9	91.6	93.2	93.4	94.61
+ Morph	86.8	92.1	93.7	94.5	94.95

Table 3.4 – Étude des *features* lexicales sur l’analyse syntaxique de surface en fonction du volume de l’ensemble d’apprentissage sur le CoNLL 2000.

Concernant l’impact de la quantité de données et les *features* lexicales disponibles, le comportement du **MT :BiRNN** est analogue à celui du système mono-tâche **E2E :BiRNN**. L’initialisation des représentations initiales des mots dans la couche d’entrée a, cette fois encore, une incidence significative dans le cas où on a accès à un volume limité de données d’apprentissage (jusqu’à 40% soit environ 80 000 mots). Les *features* morpho-syntaxiques sont essentielles à l’obtention de bonnes performances, particulièrement dans le cas où on a accès à peu de données annotées. Enfin, le **MT :BiRNN** prédisant simultanément les étiquettes

i. prédites par un **E2E :BiRNN** *POS tagger*

grammaticales et les constituants syntaxiques surpasse (à partir de 20%) l’approche *End-to-End* mono-tâche dans l’ensemble des configurations.

On veut tester si le réseau multitâche **MT:BiRNN** généralise mieux que le pipeline **PL:BiRNN** en fonction de la quantité de données annotées disponibles (cf. *Tableau 3.5*). Le **MT:BiRNN** a été entraîné à prédire simultanément les étiquettes de *Chunking* et les *POS-tags*. Le **PL:BiRNN** est composé d’un **E2E:BiRNN Chunker** utilisant des *POS-tags* provenant d’un **E2E:BiRNN POS tagger** entraîné préalablement sur le même corpus. Afin d’estimer les gains potentiels, on compare également l’utilisation en entrée de *features* morpho-syntaxiques suivant s’il s’agit d’annotations de référence ou de *POS-tags* prédits.

Data (%)	20%	40%	60%	80%	100%
<b>E2E:BiRNN</b>	86.5	91.5	93.1	93.8	94.55
+ <i>POS</i> référence	91.8	93.8	94.6	95.0	95.45
<b>PL:BiRNN</b>	88.5	92.7	93.4	94.2	94.79
<b>MT:BiRNN</b>	86.8	92.1	93.7	94.5	94.95
<b>PL:MT:BiRNN</b>	91.6	93.8	94.4	94.9	95.18

Table 3.5 – *Évaluation sur l’analyse syntaxique de surface en fonction du volume de l’ensemble d’apprentissage sur le CoNLL 2000.*

On se rend ainsi compte que la connaissance des *POS-tags* via un pipeline apporte un gain net vis-à-vis de l’approche *End-to-End* (94.8%/94.6%), quelque soit la quantité de données. En effet, le réseau multitâche ne surpasse l’approche *End-to-End* que lorsqu’on a accès à suffisamment de données d’apprentissage (entre 40 et 60% dans le cas qui nous intéresse). Cependant, lorsqu’on a accès à suffisamment de données (ici à partir de 80%) le réseau multitâche a des performances comparables à celles du pipeline (95%/94.8%). Cela signifie que l’apprentissage multitâche permet, dans une certaine mesure, de se passer d’analyse morpho-syntaxique préliminaire et donc de la connaissance des classes grammaticales des mots d’entrée, au cours de l’évaluation d’un système *Chunking*.

Dans le cas d’un apprentissage sur la totalité du corpus disponible, un réseau **E2E:BiRNN POS tagger** prédit des étiquettes morpho-syntaxiques fiables à 97.5%. Dans le modèle **PL:MT:BiRNN**, le fait d’injecter ces *features* en entrée du réseau, tout en l’entraînant à prédire les annotations grammaticales de référence améliore encore les performances de notre réseau jusqu’à 95.18%. En d’autres termes, l’addition de *features* morpho-syntaxiques prédites permet de gagner environ +0.2 points de F-mesure tandis que l’apprentissage en parallèle de la tâche de *POS tagging* rapporte +0.4 points. Ainsi, les 2.5% d’erreur en *POS tagging* ne

nous coûtent plus que 0.3%<sup>j</sup> d'erreur sur les 0.9%<sup>k</sup> initiaux. Cette approche est donc plus robuste que le *pipeline* aux erreurs d'annotations morpho-syntaxiques.

Ces expérimentations ont permis de déterminer que la connaissance des *POS-tags* des mots d'entrée, *a minima* durant l'apprentissage, est nécessaire à l'obtention de bonnes performances. Lorsqu'on a suffisamment de données, notre approche multitâche permet de se passer des *POS-tags* en évaluation. Dans le cas où on posséderait un *POS tagger* externe, produisant des annotations fiables mais présentant des erreurs, notre approche multitâche est plus robuste que le *pipeline*, tout particulièrement lorsqu'on a peu de données d'apprentissage. On souhaiterait savoir si cette propriété permet de diminuer le cumul des erreurs produit par l'application successive de modèles indépendants. Afin de vérifier cette hypothèse, il faudrait pouvoir réaliser des expérimentations sur un grand nombre de tâches imbriquées. Il serait ainsi possible de comparer les performances de chaque module d'un *pipeline* vis-à-vis de celles obtenues par un réseau multitâche. Si on tient à étudier la robustesse de nos réseaux, il est également envisageable de faire varier artificiellement le taux d'erreurs dans les *features* d'entrée.

### 3.3.3 Analyse en dépendance syntaxique

On veut tester la capacité du réseau multitâche à apprendre deux tâches étroitement imbriquées. On espère que le système **MT:BiRNN** corrèle mieux ses prédictions qu'un *pipeline* de BiRNNs. Afin de tester cette hypothèse, nous évaluons ces modèles sur l'annotation en dépendances syntaxiques (ou *Parsing*). Cette tâche nécessite, pour chaque mot d'une phrase, les prédictions conjointes d'une part de son gouverneur (le mot, ou la tête du constituant syntaxique, auquel il se rapporte) et d'autre part de l'étiquette syntaxique de la relation qui les lie. Collectivement, l'ensemble de ces relations forme un arbre de dépendance. Un *Benchmark* du *Parsing* est l'analyse du corpus *CoNLL 2009 Shared Task*<sup>l</sup>. Ce corpus, fondé sur le *Penn Treebank*, est composé de près d'un million de mots. Il est annoté en parties de discours, en dépendances syntaxiques ainsi qu'en rôles sémantiques<sup>m</sup>.

On distingue deux types d'approches traditionnelles. D'un côté, les approches *graph-based* considèrent l'ensemble des arbres de dépendances possibles afin de déterminer le plus probable (MCDONALD, NIVRE, QUIRMBACH-BRUNDAGE et al.

---

j. 95.5 – 95.2

k. 95.5 – 94.6

l. Corpus disponible <http://ufal.mff.cuni.cz/conll2009-st/>

m. Cette dernière annotation sera ignorée dans cette étude.

2013). De l'autre, les approches *transition-based* optent pour une redéfinition du problème en considérant un processus itératif sur les transitions d'une machine abstraite réalisant l'analyse (NIVRE 2008). Ces approches sont incompatibles avec notre modèle multitâche qui suppose que toutes les tâches soient synchrones *i.e.* basées sur les mêmes entrées et produisant des prédictions de même forme (des séquences de classes de même taille que la séquence d'entrée).



La formalisation de la tâche d'identification du gouverneur d'un mot, dans le cadre de la classification, est problématique. On peut, par exemple, envisager de réaliser une classification binaire au sujet de l'existence d'une dépendance entre deux éléments donnés. Cependant, l'exploration de l'espace de recherche ainsi décrit est asynchrone avec les autres tâches. On redéfinit donc la tâche de détection de gouverneur syntaxique, comme la prédiction de la distance relative séparant un mot donné de son gouverneur. Si cette approche a l'avantage de conserver la synchronie des prédictions sur l'ensemble des tâches, le prix à payer est l'absence de cohérence structurelle de la prédiction sur l'ensemble de la phrase. Concrètement, il n'y a aucune contrainte assurant qu'on prédise bien un arbre.

Data (%)	10%	20%	40%	60%	80%	100%
<b>PL:BiRNN</b>						
-Arc	66.9	77.7	81.8	84.2	85.5	87.5
-Label	84.2	90.6	92.2	92.9	93.2	93.9
-LAS	61.3	75.4	79.2	81.9	83.4	85.5
<b>MT:BiRNN</b>						
-Arc	59.3	78.8	82.8	85.0	86.6	88.5
-Label	68.7	90.2	91.8	92.5	93.0	93.4
-LAS	47.3	75.9	80.1	82.6	84.3	86.3

Table 3.6 – Évaluation sur l'analyse en dépendance syntaxique en fonction du volume de l'ensemble d'apprentissage sur le CoNLL 2009.

Nous comparons les performances d'un **PL:BiRNN** réalisant successivement l'étiquetage morpho-syntaxique, la détection de gouverneur (Arc) et l'étiquetage

de la dépendance (*Label*), et celles de notre **MT:BiRNN** entraîné sur l'ensemble de ces tâches. L'impact de la quantité de données sur l'apprentissage du *pipeline* et de notre modèle multitâche est étudié dans le tableau 3.6. On y dissocie les performances du modèle en considérant d'une part sa capacité à tirer des liens (*Arc*) et à les étiqueter (*Label*). On se rend ainsi compte que l'utilisation de notre algorithme multitâche s'accompagne d'un gain global en LAS dès 20% de données disponibles. Ce gain de LAS semble indiquer que le réseau multitâche affiche une meilleure adéquation des prédictions de détection et d'étiquetage que le *pipeline* de BiRNNs.

Cependant, les performances obtenues sont globalement inférieures aux résultats état de l'art (*cf. tableau 2.3*). En effet, la plupart des modèles état de l'art atteignent une LAS entre 87% et 90% là où notre approche atteint les 86% de taux de reconnaissance. Cette expérience montre toutefois qu'il est possible d'entraîner directement des réseaux de neurones à la détection de relation, mais que des améliorations sont nécessaires si on désire égaler les performances d'analyseur par transition du type *Shift & Reduce* (REN, JI, WAN et al. 2009 ; SWAYAMDIPTA, BALLESTEROS, DYER et al. 2016) ou par arbre couvrant maximal (CHE, Z. LI, Y. LI et al. 2009 ; BOHNET 2010).

Model	LAS
REN, JI, WAN et al. 2009	87.6
CHE, Z. LI, Y. LI et al. 2009	88.5
BOHNET 2010	90.3
SWAYAMDIPTA, BALLESTEROS, DYER et al. 2016	89.8
<b>PL:BiRNN</b>	85.5
<b>MT:BiRNN</b>	86.3

Table 3.7 – Évaluation sur l'analyse en dépendance syntaxique sur le CoNLL 2009.

L'écart de performances observées dans le cadre de la détection de gouverneur a quatre causes principales. D'une part, aucune contrainte structurelle n'est imposée à la prédiction. On conserve la propriété qu'un dépendant possède un et un seul gouverneur, mais on ne prédit pas nécessairement un arbre de dépendance faiblement connexe, acyclique et projectif. L'algorithme prédit au contraire les poids des arcs d'un graphe complet où chaque mot est un sommet. D'autre part, du fait de notre algorithme d'apprentissage par tronçon (*cf. chapitre 1*) le réseau a la possibilité de tirer des liens au-delà des frontières de la phrase. De plus, l'apprentissage de la prédiction d'une distance relative à l'aide d'un critère *LogLikeHood* (*cf. chapitre 1*) considère chaque valeur de distance comme une classe indépendante. Autrement dit, il n'y a, à aucun moment, prise en compte

d'une quelconque relation d'ordre. Enfin, le décodage glouton se contente, pour chaque mot de la phrase, de sélectionner comme gouverneur associé la cible de la transition sortante de poids maximal (ce qui assure tout de même qu'un mot ait au plus une tête syntaxique).

Notre approche multitâche **MT:BiRNN** est plus efficace sur l'analyse en dépendances syntaxiques que le *pipeline PL:BiRNN* lorsqu'on possède suffisamment de données annotées. Il semble donc que le réseau multitâche affiche une meilleure adéquation des prédictions de détection et d'étiquetage vis-à-vis du *pipeline* de BiRNNs (indiqué par un gain de LAS). Cependant, les performances de nos modèles sont inférieures à celles des modèles état-de-l'art. Une heuristique d'inférence attachant davantage d'importance à l'optimisation globale de la séquence de relations prédite, pourrait permettre de récupérer en partie les contraintes structurelles des prédictions. Par exemple, on pourrait envisager d'injecter les distributions de probabilités d'arc et/ou de label comme *features* d'un *maximum spanning tree* ou d'un analyseur par transition. Alternativement, on peut également faire du *rescoring* sur les  $n$  meilleures hypothèses de prédictions de modèles état-de-l'art sus-cités. On sélectionne l'arbre de dépendance le plus probable, au sens des probabilités prédites par le réseau. Cela permettrait notamment de récupérer les propriétés de connectivité, acyclicité et projectivité qui nous font défaut à l'heure actuelle. Une dernière alternative serait de s'inspirer des travaux décrits dans VINYALS, FORTUNATO et JAITLY 2015. Les auteurs y définissent un *Pointer Network*. Il s'agit d'un réseau de neurones récurrent dont le modèle d'attention a été détourné de son rôle initial. Plutôt que d'envisager le calcul de coefficient de mixture, ils choisissent d'interpréter la distribution  $a^t$  (cf. équation 1.33) produite, afin de prédire explicitement un nœud du réseau. Ce principe est adaptable dans notre cas, afin de prédire la position du gouverneur dans la séquence d'entrée, sans passer par l'abstraction d'indépendance des classes.

## Conclusion

Dans ce chapitre, nous proposons d'utiliser un modèle multitâche basé sur l'apprentissage de représentations communes à la résolution d'un ensemble d'analyses syntaxiques du TAL. Notre approche se base sur un ensemble de *LSTM Bidirectional Recurrent Neural Networks* partageant la même architecture et capable de réaliser conjointement des tâches d'analyse syntaxique du texte. Le cœur de ce modèle consiste à considérer autant de réseaux LSTM-BiRNN que de tâches d'intérêt. Les poids de la couche d'encodage et de la couche cachée récurrente sont partagés par l'ensemble des modèles. Ces réseaux, aux paramètres étroitement intriqués, sont ensuite entraînés conjointement à l'aide d'un critère d'erreur glo-

bale. Afin de permettre au système de s'abstraire de l'hypothèse d'indépendance des phrases, l'algorithme d'apprentissage considère des sous-séquences du corpus.

Ce modèle **E2E :BiRNN** appliqué au *POS Tagging* obtient des performances état-de-l'art en se basant sur une quantité minimale de connaissances *a priori* du domaine. Ce modèle nous servira de *baseline* tout au long de notre étude. Il a permis de déterminer que la prise en compte de *features* morphologiques et l'utilisation d'un espace de *Word Embedding* non-supervisé avait un impact favorable sur les performances du modèle, particulièrement lorsqu'on a accès à peu de données. Dans le cas du *Chunking*, l'utilisation de *features* d'entrée morpho-syntaxiques (tel que les *POS-tags*) sont nécessaires à l'obtention de performances état de l'art. Cependant, l'utilisation de *features* générées automatiquement par un analyseur indépendant entraîne un phénomène de cumul des erreurs. En effet, les *features* utilisées en évaluation sont de moins bonne qualité que celles considérées en apprentissage, ce qui a tendance à induire en erreur le réseau. C'est pourquoi nous avons désiré nous passer de cette dépendance vis-à-vis de la qualité des *features* obtenues automatiquement.

Le **MT:BiRNN** appliqué au *Chunking* atteint également des performances l'état de l'art et surpasse l'approche *End-to-End* monotâche même lorsque peu de données sont disponibles. De plus, si on possède suffisamment de données, il est possible d'utiliser notre stratégie d'apprentissage afin de s'abstraire de la connaissance des classes grammaticales des mots d'entrée, et donc d'analyse morpho-syntaxique préliminaire au cours de l'évaluation. De façon inattendue, il s'avère que l'utilisation des *POS-tags* à la fois en tant qu'entrée et en tant que sortie additionnelle, améliore globalement les performances du système. Notre approche **PL:MT:BiRNN** est donc plus robuste à l'utilisation en entrée de *features* morpho-syntaxiques erronées, que le **PL:BiRNN**. Ces propriétés pourraient être particulièrement utiles dans le cas de données bruitées ou annotées automatiquement. De plus, cela pourrait influencer sur les phénomènes de cumul d'erreurs produits par l'application successive de modèles indépendants.

Utiliser un **MT:BiRNN** afin de réaliser une tâche de *Parsing* nécessite une ré-interprétation afin de la formaliser comme un problème de classification terme-à-terme de séquences. De cette façon, la détection de relation, l'étiquetage de la relation détectée et l'ensemble des autres tâches peuvent être appris simultanément. Dans le cadre d'une analyse en dépendance syntaxique, chaque mot possède un et un seul gouverneur syntaxique. Il est donc possible d'entraîner des réseaux de neurones à la détection de relation par le biais de la prédiction mot à mot de chaque gouverneur. Il en va de même pour l'étiquette de la relation, prédite sachant un dépendant. Ainsi, notre réseau multitâche affiche une meilleure adéquation des prédictions de détection et d'étiquetage vis-à-vis du

*pipeline* de BiRNNs. Cependant, des améliorations sont nécessaires afin d'égaliser les approches état de l'art, notamment une heuristique d'inférence attachant davantage d'importance à la structure arborescente des relations prédites.

Enfin, il est important de noter que l'avantage principal de notre approche réside dans le fait de produire une annotation enrichie, à différents niveaux d'analyse linguistique et sans coût additionnel, à la différence de l'approche *pipeline* traditionnelle. Cependant, une hypothèse fondamentale du *Machine Learning*, vérifiée dans les corpus sur lesquels nous nous sommes évalués, est que la distribution d'évaluation correspond à celle d'entraînement. Mais dans les applications pratiques, cette contrainte n'est plus assurée (erreur de transcription, changement de domaine, mots hors-vocabulaires, *etc.*). Ce sont les conditions expérimentales que nous exploreront dans le chapitre suivant. Il sera également l'occasion d'aborder d'autres pistes, notamment concernant la distribution de poids  $\beta$  du critère d'apprentissage à considérer. On désire étudier le comportement de notre modèle dans le cas de l'apprentissage d'un grand nombre de tâches afin de vérifier le passage à l'échelle de notre approche. De plus, afin d'éprouver la généralisation ainsi que la robustesse de notre approche, on souhaiterait s'évaluer sur des données bruitées ou annotées automatiquement.

# 4 Analyse et Compréhension de la langue orale

Nous avons décrit dans les chapitres précédents un réseau de neurones dédié à l'étiquetage de séquences et un algorithme d'apprentissage multitâche capable de réaliser conjointement un ensemble de tâches d'analyse linguistique. Ce modèle tire profit de corpus annotés pour différentes tâches. On a notamment illustré son efficacité sur des tâches d'analyse syntaxique, sur des corpus de référence composés de textes journalistiques annotés manuellement. On souhaite désormais éprouver ce modèle en l'évaluant dans le cadre de la compréhension du langage oral. On s'intéresse dans ce chapitre à l'analyse sémantique de transcriptions de conversations téléphoniques telles que définies dans le projet européen SENSEI. On considérera distinctement le cas des transcriptions manuelles et automatiques afin d'éprouver la robustesse de notre modèle aux erreurs de reconnaissance.

## 4.1 État de l'Art

La compréhension automatique de la langue orale a suscité beaucoup d'intérêt ces dernières années. Cela passe notamment par la création de ressources sémantiques telles que les projets ATIS ou MEDIA. Ces modèles sémantiques, spécifiques à un domaine, sont basés sur des transcriptions de conversations scriptées simulant une interaction naturelle. Ces ressources, bien qu'artificielles, ont permis de développer des modèles génériques désormais applicables à des cas d'utilisation plus réalistes. TUR et DE MORI [2011](#) dressent un panorama des modèles et méthodes traditionnellement employés. Indépendamment des paradigmes envisagés (analyse, classification, étiquetage de séquence), les concepts et relations sémantiques sont toujours prédites à partir des transcriptions automatiques, produites par des systèmes ASR (*Automatic Speech Recognition*), parfois annotées au moyen de modèles génériques d'analyse syntaxique et sémantique.

Ces analyseurs ont cependant été conçus afin de traiter du texte écrit, majoritairement composé de texte journalistique. Leur application dans le cadre de l'oral spontané reste encore limitée. Parmi les modèles développés dans le cadre de la compréhension de l'oral, on peut citer les modèles à prédicat/argument tels que la reconnaissance de rôle sémantique ou SRL (*Semantic Role Labeling* : GIL-

DEA et JURAFSKY 2002 ; PRADHAN, WARD, HACIOGLU et al. 2005), la détection de cadres sémantiques (FrameNet : BAKER, Charles J FILLMORE et LOWE 1998) et plus récemment des méthodes d'analyse syntaxique et sémantique telle que le modèle *Abstract Meaning Representation* (BANARESCU, BONIAL, CAI et al. 2012). Le point commun de l'ensemble de ces travaux est qu'il reposent tous sur des modèles sémantiques génériques, initialement dédiés à l'analyse de l'écrit. Ces méthodes sont ainsi appliquées à des transcriptions automatiques comportant possiblement des erreurs, sans adaptation particulière qui auraient permis d'appréhender les spécificités de l'oral spontané. Cette approche est raisonnable tant que les énoncés à traiter sont relativement courts et faciles à analyser (HAKKANITUR, JU, ZWEIG et al. 2015).

Cependant, lorsqu'on considère des dialogues spontanés comme par exemple des conversations téléphoniques, BECHET, Alexis NASR et FAVRE 2014 ont mis en évidence qu'une adaptation des modèles génériques était nécessaire pour deux principales raisons :

1. D'une part, les transcriptions de conversations spontanées sont toujours difficiles à analyser convenablement à partir de modèles développés pour l'écrit, à cause des nombreux phénomènes syntaxiques propres à l'oral spontané (agrammaticalité, disfluences telles que les faux-départs, les corrections et autres répétitions intempestives) ;
2. D'autre part, les transcriptions obtenues *via* les systèmes de reconnaissance automatique contiennent bien souvent des erreurs. Ce taux d'erreurs croît fortement avec le degré de spontanéité du discours, ce qui perturbe le bon fonctionnement des systèmes d'analyse standard.

Quand les transcriptions automatiques contiennent trop d'erreurs, même les systèmes d'analyse les plus robustes peuvent échouer. C'est particulièrement observable dans le cas des approches *pipeline*, courantes en traitement automatique de la langue. Ces systèmes étant composés de plusieurs modèles disposés en cascade, une erreur d'analyse commise en amont du *pipeline* se répercute inévitablement en aval. Une solution courante pour traiter le problème des erreurs en cascade est le passage d'hypothèses multiples entre les modules. Au prix d'une augmentation de la complexité des modules, la connaissance des  $n$  meilleures prédictions du module précédent, permet de limiter la propagation des erreurs. Nous proposons une approche utilisant le *pipeline* MACAON (NASR et BÉCHET 2011) composé de modèles génériques d'analyse syntaxique et sémantique, adaptés à l'oral spontané (BECHET, Alexis NASR et FAVRE 2014) afin d'annoter automatiquement des transcriptions manuelles. Ces transcriptions seront utilisées, dans un second temps, afin d'entraîner en *End-to-End* notre réseau de neurones multitâche à reproduire l'ensemble de ces annotations uniquement

à partir des mots. On se passe ainsi de la connaissance des annotations syntaxiques en inférence.

Notre objectif est de montrer que l'apprentissage multitâche évalué au cours du chapitre précédent est robuste dans le cadre difficile de transcriptions automatiques, car reposant sur des représentations riches, supervisées conjointement par un grand nombre de tâches linguistiques. De plus, ce modèle n'est dépendant des annotations intermédiaires produites par le *pipeline*, que pour la phase d'apprentissage. Lors de la phase de prédiction, seuls les mots sont considérés, ce qui permet de s'abstraire des erreurs potentielles qu'auraient commises l'analyse préliminaire.

Un autre aspect qui sera traité dans ce chapitre est le décalage entre, d'une part le corpus qui sert à apprendre les représentations continues des mots (*embeddings*), et d'autre part celui servant à entraîner les modèles d'analyse sémantique. Le premier est constitué de très grandes quantités de textes génériques, principalement journalistiques, alors que le second est limité aux transcriptions annotées spécifiques à la tâche visée, disponibles en petites quantités. Ceci nous a poussé à étudier les phénomènes de mots hors-vocabulaire et à utiliser des stratégies d'adaptation, afin de produire de meilleures représentations pour ces mots inconnus.

## 4.2 Corpus de transcriptions

Dans le cadre de notre étude, nous avons considéré le corpus RATP-DECODA<sup>a</sup> composé de près de 2000 conversations téléphoniques enregistrées sur une période de deux jours auprès du centre d'appel des transports publics de la ville de Paris (BECHET, MAZA, BIGOUROUX et al. 2012). Ce cadre applicatif est particulièrement intéressant car il permet de collecter énormément de données, met en scène un grand nombre d'interlocuteurs, et contient peu d'informations personnelles qu'il faudrait anonymiser afin de librement distribuer la ressource ainsi produite. Le corpus totalise près de 74 heures de français oral spontané. Bien que les conversations durent en moyenne 3 minutes, un tiers d'entre-elles ne dépassent pas la minute, 12% font près de 5 minutes tandis que les plus longues dépassent les 10 minutes. Les appels mettent généralement en scène seulement deux interlocuteurs, bien qu'on en compte parfois davantage lorsque l'agent prend contact avec un autre service, tandis que le client est prié de patienter. L'ensemble de ces conversations a été anonymisé, segmenté en tour de parole

---

a. Le corpus RATP-DECODA est disponible sur le site Ortolang SLDR data repository : <http://sldr.org/sldr000847/fr>

et transcrit. Le centre d'appel est responsable du service client et les deux jours d'enregistrement, couvrent un large champ de situations, tel que des demandes d'horaires, de direction, d'informations tarifaires, des requêtes d'objets perdus ou des renseignements administratifs.

## 4.2.1 Annotations syntaxiques

Les conversations téléphoniques du corpus RATP-DECODA contiennent de la parole très spontanée, contenant de nombreuses disfluences telles que des répétitions (e.g. *le le ...*), des marqueurs de discours (e.g. *eah, bien*) et des faux-départs (e.g. *bonj- bonjour*). À cause de cet important degré de spontanéité dans les énoncés à traiter, les modèles d'étiquetage morpho-syntaxique ou d'analyse en dépendances syntaxiques, développés pour l'écrit canonique doivent être adaptés au traitement de l'oral. Cette adaptation a été réalisée à l'occasion du projet ANR ORFEO<sup>b</sup> pour la chaîne d'analyse linguistique que nous utilisons dans cette étude (MACAON A. NASR, F. BÉCHET, REY et al. 2011). Cette adaptation est décrite précisément dans BAZILLON, DEPLANO, BECHET et al. 2012. Elle consiste à annoter automatiquement un corpus de transcriptions de conversations téléphoniques à l'aide d'analyseurs génériques, puis à corriger manuellement ces annotations. Ce processus itératif est réalisé jusqu'à ce qu'un système entraîné à partir des annotations corrigées, atteigne un taux d'erreur acceptable. Une évaluation de cette méthode d'adaptation est présentée dans (BECHET, Alexis NASR et FAVRE 2014). À ce stade, le corpus RATP-DECODA est annoté en parties de discours, en disfluences ainsi qu'en dépendances syntaxiques. À ces trois niveaux d'analyse syntaxique s'ajoutent deux niveaux d'annotations sémantiques que nous allons décrire ci-après.

## 4.2.2 Annotations sémantiques

Le modèle sémantique considéré dans cette étude est constitué de deux niveaux d'annotation : les entités nommées d'une part, et les cadres sémantiques d'autre part. Les entités nommées représentent aussi bien des localisations (rues, adresses, stations de bus ou de métro), des organisations (différents pôles de la RATP), des services (lignes de bus ou de métro) ou encore des informations temporelles. L'analyse en cadres sémantiques a été réalisée dans le cadre du projet SENSEI. Elle est décrite dans TRIONE, BECHET, FAVRE et al. 2015 et est basée

---

b. <http://www.agence-nationale-recherche.fr/?Projet=ANR-12-CORP-0005>

sur le formalisme *FrameNet* adapté à la langue française au travers du projet ANR ASFALDA<sup>c</sup>. Ce type d'analyse sémantique en cadres, provient de Charles J. FILLMORE 1976 et consiste à représenter, sous forme abstraite, les concepts sémantiques présents dans une phrase. Chacun de ces cadres est déclenché par un mot de la phrase (*trigger*) et contient tous les constituants jouant un rôle dans la réalisation de ce concept (les *frame elements*). Les annotations sémantiques *FrameNet* permettent de détecter dans une phrase l'expression des cadres sémantiques. Comme ces cadres s'abstraient, en partie, des variations syntaxiques et lexicales, ils permettent d'accéder au sens, à la signification d'un énoncé (*qui a fait quoi, où, quand et comment ?*).

Ce schéma d'annotation en cadres sémantiques a été appliqué au corpus annoté syntaxiquement comme décrit dans (TRIONE, BECHET, FAVRE et al. 2015). Le corpus n'a cependant pas été annoté dans son intégralité. Les auteurs ont sélectionné un sous-ensemble d'unités lexicales pertinentes pour le domaine d'application, et ont été annotés les cadres et leur éléments respectifs, déclenchés par ces unités lexicales dans le corpus. Dans la théorie *FrameNet*, les cadres sémantiques sont les briques essentielles porteuses de sens. On pourrait définir les cadres comme composés d'un noyau autour duquel graviteraient un certain nombre d'éléments. Le noyau correspond au concept que le cadre décrit, alors que les éléments apportent de l'information supplémentaire concernant ce concept. Par exemple, dans le cadre *Activity\_finish* (déclenché lorsqu'un agent finit une activité), les éléments sont généralement l'*agent*, l'*activité* ainsi que diverses *circonstances*. Bien que les cadres soient définis de façon abstraite, leurs réalisations sont associées aux mots de la phrase que le cadre illustre. Lorsqu'ils sont instanciés, le noyau du cadre est associé à un mot déclencheur (*trigger*) tandis que les éléments sont associés aux mots remplisseurs de rôles (*frame element*). Les cadres sont considérés indépendants, ce qui signifie que rien n'interdit à un même mot de remplir un rôle dans deux cadres différents.

Afin d'appliquer le réseau de neurones décrit précédemment à la résolution de cette tâche, il est nécessaire de formaliser la prédiction de ces annotations comme un problème de classification, et plus précisément comme un problème d'étiquetage de séquence. Nos expérimentations ont donc nécessité qu'on réalise l'approximation suivante : l'annotation en cadres sémantiques a été projetée mot-à-mot *i.e* chaque mot est, soit étiqueté null s'il ne participe pas à la réalisation d'une *frame*, soit associé au nom du cadre qu'il déclenche ainsi qu'à son rôle dans la réalisation de ce cadre. Dans notre corpus d'apprentissage, 28% des mots ont ainsi une étiquette sémantique parmi un total de 335 étiquettes différentes à prédire, représentant 71 cadres sémantiques différents. De nombreuses ambiguïtés proviennent notamment des disfluences apparaissant dans ce corpus de conversations orales hautement spontanées.

---

c. <https://sites.google.com/site/anrasfalda>

<i>id</i>	<i>mot</i>	<i>POS</i>	<i>disf</i>	<i>NE</i>	<i>label</i>	<i>arc</i>	<i>frame</i>
1	I	prp	rep	—	disf	2	—
2	I	prp	—	—	sbj	3	B_losing_agent
3	lost	vbp	—	—	root	0	B_losing_LU
4	my	prp\$	—	—	nmod	5	B_losing_obj
5	phone	nn	—	—	obj	3	I_losing_obj
6	in	in	—	—	loc	5	B_losing_obj
7	bus	nn	—	B_tran	pmod	6	I_losing_obj
8	38	cd	—	I_tran	mod	7	I_losing_obj

Table 4.1 – Exemple d’annotation pour la phrase : *I I lost my phone in bus 38*. Les étiquettes de cadre sémantique suivent une syntaxe simple : **position** (soit *B* pour *begin*, soit *I* pour *inside*); **frame name**; **role** (*agent*, *object* ou *LU* pour *lexical unit*).

Les annotations automatiques en cadres sémantiques sont généralement basées sur une analyse syntaxique préalable. Cela a d’ailleurs été le cas, pour l’annotation sémantique du corpus RATP-DECODA présentée dans TRIONE, BECHET, FAVRE et al. 2015. L’avantage principal de cette méthode est qu’elle permet de minimiser la supervision nécessaire à la réalisation de l’annotation sémantique si l’on dispose d’une analyse syntaxique fiable des énoncés. Cependant, cela signifie également qu’en phase de prédiction, l’analyse sémantique devient tributaire de la bonne qualité de l’analyse syntaxique préliminaire, analyse en général peu fiable dans le cas de transcriptions automatiques contenant des erreurs. On se propose de répondre à ce problème en considérant que seules les transcriptions de références utilisées en apprentissage sont annotées syntaxiquement. Ce corpus d’apprentissage est utilisé afin d’entraîner le **MT:BiRNN** à prédire l’ensemble des annotations syntaxiques et sémantiques conjointement. Ainsi, en phase de prédiction, seules les transcriptions non-annotées sont nécessaires.

## 4.3 Expérimentations

Dans cette section, on désire montrer que le modèle exposé dans le chapitre 3 supporte l’apprentissage d’un grand nombre de tâches sans dénaturer les prédictions par rapport à un apprentissage individuel. On l’évalue donc sur le corpus RATP-DECODA où sont annotées les classes morpho-syntaxiques, les disfluences, les entités nommées, les dépendances syntaxiques et les cadres ou *frames* sémantiques. La tâche principale sur laquelle nous nous sommes concentrés est

la détection de *frames* sémantiques, tout en réalisant conjointement l'ensemble des autres tâches d'analyses. Le corpus RATP-DECODA étant composé de transcriptions de conversations téléphoniques, nous souhaitons mettre au point des conditions expérimentales permettant de vérifier la robustesse de notre modèle aux erreurs de reconnaissance introduites par les systèmes de transcription automatique.

### 4.3.1 Conditions expérimentales

Les expérimentations abordées dans cette section portent sur le corpus RATP-DECODA annoté automatiquement présenté dans la section précédente. Ce corpus est divisé en trois partitions : un ensemble d'apprentissage *train*, de validation *dev* et d'évaluation *test* décrits dans le tableau 4.2. Seules les annotations automatiques de l'ensemble d'évaluation *test* ont été vérifiées et, le cas échéant, corrigées manuellement. Les ensembles d'apprentissage *train* et de validation *dev* ont, quant à eux, été annotés de manière semi-supervisée (BAZILLON, DEPLANO, BECHET et al. 2012 ; BECHET, Alexis NASR et FAVRE 2014) et contiennent donc un certain nombre d'erreurs.

La tâche principale sur laquelle nous nous sommes évalués est la détection de *frames* sémantiques. Par exemple, dans la phrase annotée citée en 4.1, il y a quatre segments à détecter : **losing\_agent** (*I*), **losing\_LU** (*lost*), **losing\_obj** (*my phone*), **losing\_obj** (*in bus 38*).

part.	#dialogues	#tours	#mots	%frames
apprentissage	1243	76158	495451	28.8%
dev	144	9074	60968	28.7%
test	100	3347	23258	29.2%

Table 4.2 – Description des corpus d'apprentissage, de développement et de test du corpus RATP-DECODA

On considère d'une part les transcriptions manuelles de référence, et d'autre part les transcriptions automatiques. Ces transcriptions automatiques proviennent de LAILLER, LANDEAU, Frédéric BÉCHET et al. 2016. Elles sont obtenues grâce au système LIUM, lui-même basé sur Kaldi (POVEY, GHOSHAL, BOULIANNE et al. 2011), incorporant des modèles neuronaux acoustiques ainsi que des outils de *re-scoring* (DELÉGLISE, ESTÈVE, Sylvain MEIGNIER et al. 2005). Le taux d'erreur mot

moyen (WER) est de 34.4%. Ce taux élevé est majoritairement dû à la difficulté de gestion des disfluences et des bruits environnant. Tous les hyper-paramètres des modèles présentés ont été déterminés à partir de leurs performances respectives sur l'ensemble de validation.

### 4.3.2 Évaluation sur les transcriptions manuelles

En premier lieu, on évalue les différentes variations de notre modèle BiRNN sur les transcriptions manuelles du corpus RATP-DECODA. Les résultats obtenus par un apprentissage individuel de chaque tâche d'intérêt (**E2E :BiRNN**) sont comparés aux performances d'un *pipeline* de BiRNNs (**PL :BiRNN**) et du modèle multitâche (**MT :BiRNN**) dans le tableau 4.3. Les modèles **E2E :BiRNN** et **MT :BiRNN** réalisent des prédictions *End-to-End*, c'est-à-dire à partir des mots uniquement. Le *pipeline* a, quant à lui, accès aux prédictions des tâches précédentes. Par exemple, le système **PL :BiRNN**(NER) dédié à la détection d'entités nommées, prend en entrée un mot ainsi que les prédictions des modèles **E2E :BiRNN**(PoS) et **PL :BiRNN**(Disf). Afin de vérifier l'impact des erreurs de reconnaissance lors de la prédiction de ces *features* syntaxiques sur la détection de *frames* sémantiques, un modèle **PL :BiRNN**(Frame) ayant accès aux annotations *gold* initialement présentes dans le corpus mais inaccessibles en pratique, atteint une F-mesure de 81.6%, à comparer aux 79.2% d'un **PL :BiRNN**(Frame) en conditions réelles.

Modèle	PoS	Disf	NER	Syntax	Frame
<b>E2E :BiRNN</b>	95.8	87.6	78.6	87.5	78.9
<b>PL :BiRNN</b>	95.8	87.4	78.6	85.6	79.2
<b>MT :BiRNN</b>	95.8	89.8	79.0	88.4	79.6

Table 4.3 – Évaluation sur les transcriptions manuelles du corpus RATP-Decoda : comparaison entre l'approche BiRNN End-to-End, Pipeline et Multi-tâche.

Ces résultats montrent que les modèles multitâche égalent voire surpassent les performances des apprentissages monotâche et du *pipeline*, sur l'ensemble des tâches, ce qui valide notre approche. Toutefois, il est étonnant de constater le faible apport de nos *features pipeline* par rapport au gain espéré dans le **PL :BiRNN**. Ceci est probablement dû à un phénomène de cumul des erreurs de reconnaissance, lors de l'évaluation. En effet, les *features* prédites sur l'ensemble d'apprentissage sont fiables, tandis que celles utilisées en test sont de bien moins bonne qualité (seulement 78,6% dans le cas des entités nommées).

Data (%)	10%	20%	40%	60%	80%	100%
<b>E2E :BiRNNN</b>						
-POS	92.3	94.8	95.2	95.6	95.7	95.8
-Disf	73.2	80.8	83.4	85.3	86.9	87.6
-NER	65.9	75.8	77.6	78.0	78.3	78.6
-Syntax	47.5	81.1	85.8	86.5	86.9	87.5
-Frame	46.2	71.2	75.9	77.6	78.5	78.9
<b>MT :BiRNN</b>						
-POS	87.7	94.4	95.2	95.5	95.6	95.8
-Disf	74.3	85.3	87.4	88.1	88.8	89.8
-NER	61.6	76.1	77.8	78.3	78.4	79.0
-Syntax	40.2	80.2	85.4	86.5	87.6	88.4
-Frame	42.6	69.6	75.8	77.7	78.3	79.6

Table 4.4 – Évaluation sur l'étiquetage morpho-syntaxique, l'analyse en dépendance syntaxique, la détection de disfluence, la reconnaissance d'entité nommée et la détection de frame sémantique en fonction du volume de l'ensemble d'apprentissage sur les transcriptions manuelles du corpus RATP-Decoda.

Nous avons par la suite étudié l'impact de la quantité de données utilisées en apprentissage sur les performances de chacune de nos tâches d'intérêt. Les résultats obtenus sont présentés dans le tableau 4.4. Ce qu'on constate, c'est qu'un volume de données minimal est nécessaire afin que le modèle multitâche égale puis supplante les modèles individuels. Cette quantité de données est propre à chaque tâche. Par exemple, on observe des gains significatifs en détection de disfluence dès 10%, alors qu'il faut attendre d'avoir injecté 60% du corpus afin que les performances de l'analyse en dépendances syntaxiques du modèle multitâche rattrapent celles du BiRNN. On peut, d'une part, interpréter ce résultat en considérant que certaines tâches sont plus "difficiles" que d'autre *i.e.* nécessitent davantage de données afin d'obtenir un *classifier* performant. D'autre part, il est possible d'imaginer que ces quantités de données minimales sont corrélées avec les poids  $\beta$  du critère d'apprentissage complexe, menant à un apprentissage multitâche optimal. En effet, du fait de nos observations, il semble logique de penser que le poids alloué à la détection de disfluence devrait être inférieur à celui dédié à l'analyse en dépendances syntaxiques. Cette tâche, nécessitant moins de ressources, semble plus simple à apprendre pour le réseau, par conséquent son influence dans le critère d'apprentissage devrait être moindre. Cependant, des expériences complémentaires sont nécessaires afin de valider cette intuition.

Enfin, on étudie conjointement l'impact de la quantité de données disponibles en apprentissage et de la distribution de poids  $\beta$  considérée au travers de la Fi-

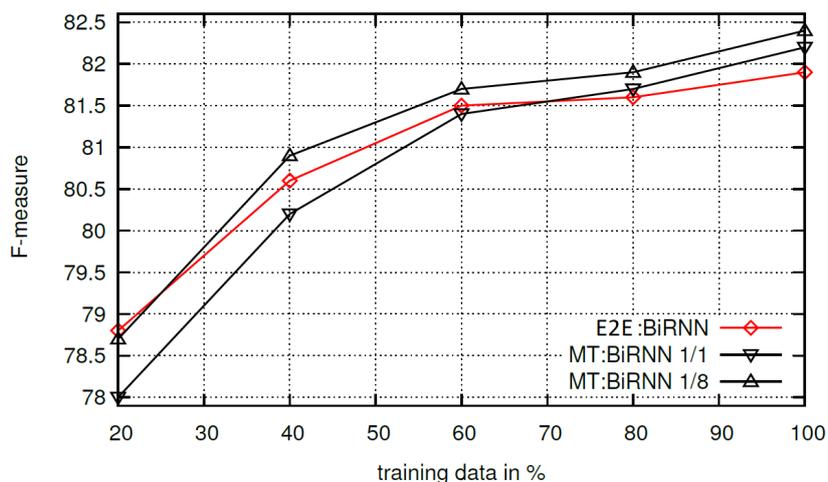


Figure 4.1 – Évaluation (F-score) sur la tâche de détection de cadre sémantique en fonction du volume de l'ensemble d'apprentissage pour trois configurations : apprentissage monotâche (E2E:BiRNN), apprentissage multitâche uniforme (MT:BiRNN 1/1) et avec un poids  $\beta$  dédié aux frames plus important (MT:BiRNN 1/8).

Figure 4.1. On compare trois modèles : un système monotâche (E2E:BiRNN), un modèle multitâche uniforme (MT:BiRNN 1/1), et un modèle multitâche apportant une attention particulière à la détection de cadre sémantique (MT:BiRNN 1/8). Comme on peut le constater, avec une distribution de poids uniforme, on peut espérer une amélioration des performances de reconnaissance à l'aide du MT:BiRNN, lorsque suffisamment de données d'apprentissage sont disponibles (dès environ 70% du corpus complet). En augmentant le poids du critère d'apprentissage alloué à la détection de cadre sémantique, on observe un gain supplémentaire au prix d'une faible baisse de performances sur les autres tâches d'intérêt. Qui plus est, seulement 25% du corpus est alors nécessaire afin d'observer les premières améliorations vis-à-vis de l'approche monotâche.

### 4.3.3 Évaluation sur les transcriptions automatiques

Dans cette partie, on évalue les différentes variations de notre modèle BiRNN sur les transcriptions automatiques du corpus RATP-DECODA. Nos résultats expérimentaux sont présentés dans les tableaux 4.5 et 4.6. On y compare cinq systèmes d'analyse sémantique : deux modèles *End-to-End* (CRF et BiRNN) réalisant directement la tâche de détection de cadre sémantique à partir des séquences de mots, deux approches *pipeline* (PL:CRF et PL:MACAON) réalisant une succession de tâches du traitement automatique de la langue (étiquetage

morpho-syntaxique, détection de disfluece, reconnaissance d’entités nommées, analyse en dépendances syntaxiques) avant de prédire les *frames*, et enfin notre modèle d’analyse multitâche (**MT:BiRNN**) réalisant l’ensemble des tâches citées précédemment directement à partir des mots.

Afin de pallier les possibles erreurs d’alignement dans les transcriptions automatiques, pour chaque segment détecté par un système, on vérifie si un segment, avec une étiquette de *frame* identique, apparaît dans le même intervalle de temps  $\pm\delta$  (avec  $\delta = 2s$ ). Afin de fournir des résultats comparables, on applique la même heuristique d’évaluation pour les transcriptions manuelles de référence. Le tableau 4.5 présente les mesures de Précision, Rappel et F-score ainsi obtenus par les différents modèles étudiés dans cette section :

- **E2E:CRF** : un *Conditional Random Field* entraîné à la détection de cadre sémantique en *End-to-End* i.e. en ne considérant que les mots ;
- **PL:MACAON** : un analyseur syntaxique état de l’art, basé sur le *pipeline* MACAON (A. NASR, F. BÉCHET, REY et al. 2011), adapté pour manipuler des transcriptions d’oral spontané (Alexis NASR, BECHET, FAVRE et al. 2014). La détection de cadre sémantique est réalisée grâce au même système à base de règles ayant servi à annoter les transcriptions manuelles du corpus (TRIONE, BECHET, FAVRE et al. 2015) ;
- **PL:CRF** : un *Conditional Random Field* tirant parti des *features* syntaxiques produites par le *pipeline* MACAON (parti de discours, disfluences, entités nommées et dépendances syntaxiques) ;
- **E2E:BiRNN** : un RNN bidirectionnel entraîné en *End-to-End*. Les mots sont projetés dans un espace d’*embedding* ;
- **MT:BiRNN** : un RNN bidirectionnel multitâche entraîné à détecter aussi bien les *frames* sémantiques que les classes morpho-syntaxiques, les disfluences, les entités nommées et des annotations de dépendances syntaxiques.

trans.	ref. transcriptions			ASR (WER=34.4)		
	P	R	F	P	R	F
<b>E2E:CRF</b>	78.4	72.5	75.3	74.2	44.0	55.3
<b>PL:CRF</b>	78.4	80.4	79.4	72.5	48.9	58.4
<b>PL:MACAON</b>	79.6	84.2	81.8	69.4	52.4	59.7
<b>E2E:BiRNN</b>	79.7	84.3	81.9	70.9	53.9	61.3
<b>MT:BiRNN</b>	79.9	85.0	<b>82.4</b>	71.1	54.3	<b>61.6</b>

Table 4.5 – Évaluation sur la détection de cadre sémantique sur les transcriptions manuelles et automatiques du corpus RATP-Decoda.

Dans les résultats présentés dans le tableau 4.5, on observe tout d’abord que le *pipeline* MACAON supplante les CRF, probablement car, au sein de **PL:MACAON**,

la détection de cadre sémantique est réalisée grâce au même système à base de règles ayant servi à annoter les transcriptions. Les modèles BiRNN égalent les performances de MACAON sur les transcriptions manuelles et sont tous les deux plus performants que les *pipelines* sur les transcriptions automatiques. L'apprentissage multitâche apporte un gain ténu mais constant (particulièrement en Rappel). Il est d'ailleurs intéressant de noter que l'approche **E2E:BiRNN** monotâche offre des performances en détection de *frames* proches de celles de l'approche multitâche, et ce bien qu'aucune information syntaxique ne fût fournie au réseau durant la phase d'entraînement. Il est également intéressant de remarquer que nos deux modèles BiRNN surpassent le *pipeline* MACAON dans le cas de transcriptions automatiques. Cela s'explique par la sensibilité des composants d'analyse de MACAON aux erreurs de reconnaissance. Cela peut signifier que nos réseaux bidirectionnels sont parvenus à suffisamment généraliser leurs prédictions, afin de mieux s'abstraire des possibles erreurs de reconnaissance dans les tâches syntaxiques.

Model	$\delta = 1s$	$\delta = 2s$	$\delta = 4s$
<b>PL:MACAON</b>	55.8	59.7	63.1
<b>E2E:BiRNN</b>	57.5	61.3	64.9
<b>MT:BiRNN</b> 1/1/1/1/1	57.5	61.5	64.8
<b>MT:BiRNN</b> 1/1/1/1/4	57.6	61.6	65.0
<b>MT:BiRNN</b> 1/1/1/1/8	57.9	61.8	65.5
<b>MT:BiRNN</b> 1/1/1/1/16	57.6	61.5	65.2

Table 4.6 – *Évaluation sur la détection de cadre sémantique sur les transcriptions automatiques du corpus RATP-Decoda, en fonction de la distribution  $\beta$  du critère d'apprentissage multitâche. Afin de pallier les erreurs d'alignement des transcriptions automatiques, pour chaque segment détecté, on vérifie si un segment avec une étiquette de frame identique apparaît dans le même intervalle de temps  $\pm\delta$*

De plus, il est possible d'améliorer les performances d'une tâche en particulier, en déterminant la meilleure distribution de poids  $\beta$  du critère d'apprentissage complexe, à partir des performances de reconnaissance obtenues sur l'ensemble de validation *dev*. Par exemple, comme explicité dans le tableau 4.6, les meilleures performances de notre système dans le cadre de la détection de cadre sémantique sont obtenues en considérant un poids  $\beta_{frame}$  huit fois plus important que le poids alloué à chacune des autres tâches. Cela laisse à penser qu'il existe une distribution de poids menant à un apprentissage optimal, *a minima* pour une tâche en particulier. Cependant, dans un souci d'efficacité et en raison du nombre important de distributions possibles à tester, une heuristique plus fine

d'approximation de ces poids devrait être envisagée.

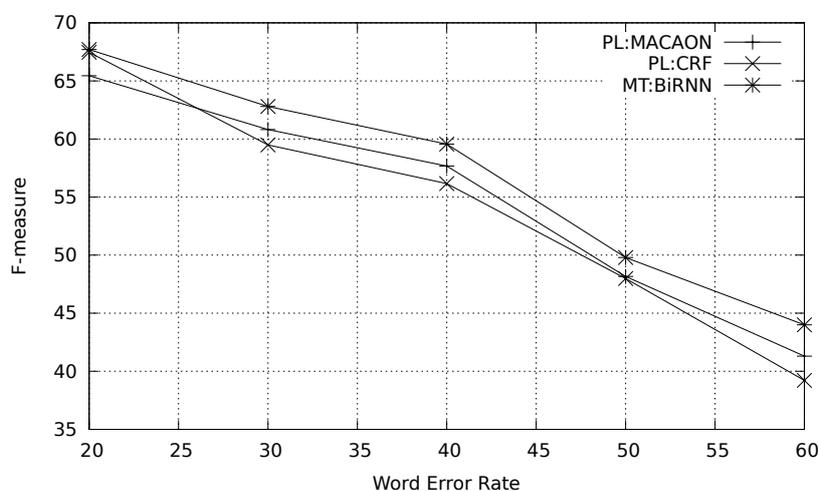


Figure 4.2 – Évaluation (*F*-score) sur la tâche de détection de cadre sémantique en fonction du taux erreurs mot WER des transcriptions automatiques, de trois modèles : les deux pipelines *PL:MACAON* et *PL:CRF*, et notre réseau multitâche (*MT:BiRNN*).

Enfin, la robustesse de nos modèles aux erreurs de transcriptions ASR est étudiée au travers de la Figure 4.2. Nous avons divisé le corpus d'évaluation en fonction du taux erreur mot (WER) des transcriptions automatiques. On considère cinq partitions respectivement à 20, 30, 40, 50 et plus de 50% de taux d'erreur. On compare les performances des trois systèmes *PL:MACAON*, *PL:CRF* et *MT:BiRNN* sur ces différentes partitions. Comme dans le cas des transcriptions manuelles, le *pipeline* *MACAON* est plus performant que le *CRF*. Bien que les gains restent ténus, on observe que notre modèle multitâche *MT:BiRNN* est plus robuste que ces deux homologues *pipeline* aux erreurs induites par les systèmes ASR. D'après le tableau 4.5, ces performances semblent tout autant dues à l'utilisation d'un modèle *BiRNN* qu'à l'apprentissage multitâche. Il serait intéressant d'étudier plus en détail l'apport respectif des *embeddings*, de la structure récurrente et de l'algorithme d'apprentissage, afin de déterminer précisément la source du gain. Cependant, si notre approche est plus robuste aux erreurs de transcription, elle ne cherche nullement à les corriger. Une approche classique consiste à entremêler reconnaissance de la parole et analyse syntaxique (JOUSSE, S. MEIGNIER, JACQUIN et al. 2009). Bien que les améliorations obtenues par addition de *features* syntaxiques aux modèles ASR soient généralement marginales, la structure syntaxique résultant de l'analyse en elle-même facilite la compréhension automatique, même sans diminution notable du taux d'erreur mots dans les transcriptions. Une perspective directe de notre travail consisterait à considérer un système *End-to-End* multitâche intégré au processus de reconnaissance afin

de permettre au système de prendre en compte différentes hypothèses de transcription durant l'analyse.

## 4.4 Problème des mots inconnus

Dans le chapitre 2, on est parti de l'hypothèse qu'on possédait un espace d'*embedding* présenté sous forme d'un dictionnaire ou d'un lexique, associant les mots à leurs représentations vectorielles. Différents choix d'espace de représentation de mots et leur injection dans des réseaux de neurones ont été décrits dans la section 2.4. On a notamment proposé d'initialiser les poids de la couche d'entrée à l'aide de l'espace d'*embedding*, afin de permettre au réseau de le transformer au cours de son apprentissage. En entraînement, chaque exemple raffine les représentations des mots dont il est composé.

Lorsqu'elles ne sont pas initialisées aléatoirement, ces représentations ont été calculées à partir d'un corpus  $C_{\text{emb}}$  de données textuelles couvrant un vocabulaire noté  $V_{\text{emb}}$ . Cet espace d'*embedding* est utilisé par un réseau de neurones dans le cadre de l'apprentissage d'une tâche du TAL. Cette tâche d'intérêt est représentée par un corpus **annoté**  $C_{\text{task}}$  couvrant un vocabulaire  $V_{\text{task}}$ . On cherche à reproduire et à généraliser l'annotation à partir des exemples observés dans  $C_{\text{task}}$ . Ce corpus se compose d'un ensemble d'apprentissage  $C_{\text{train}}$  à l'aide duquel est entraîné le réseau, et d'un ensemble de test  $C_{\text{test}}$  dédié à l'évaluation du modèle entraîné.

Cependant, il est important de garder à l'esprit qu'un lexique aussi important soit-il, de sources aussi variées soient-elles, et qu'on estime représentatif de l'ensemble des usages de la langue, n'est jamais complet. Un des problèmes essentiels en TAL est celui des mots inconnus, c'est-à-dire des mots qui ne sont pas répertoriés dans les lexiques de termes auxquels ont recours les systèmes d'analyse. Parmi les mots inconnus figurent des néologismes, des mots étrangers importés, des noms propres, des sigles, des mots mal orthographiés ou non accentués et enfin des mots propres à la terminologie du domaine ou du registre de langue. Ce phénomène est particulièrement sensible dans le cas du traitement de l'oral spontané. En effet, la collecte et l'annotation de données, sont des tâches ardues, et si de nos jours, on trouve en abondance des corpus d'écrits canoniques, les ressources disponibles relatives à la langue orale spontanée sont toujours limitées.

## 4.4.1 Caractérisation

De manière générale, un mot hors-vocabulaire  $w$  est un mot qui apparaît dans le corpus d'évaluation mais qui n'a aucune occurrence dans  $C_{\text{train}}$ . Les systèmes standards faisant l'amalgame entre un mot et son indice dans le lexique (cf. Chapitre 2) et  $w$  étant absent du dit-lexique, comment le représenter ? Dans le cadre d'un modèle statistique à base de  $N\text{-gram}$ , on tombe sur un écueil statistique classique. Par exemple, dans un CRF, les mots ou *features* non-observés en entraînement ont un poids nul, ce qui signifie que le modèle les ignore. On repose alors sur des *features* extraites à partir des caractères du mot (préfixes, suffixes), mais elles ne permettent pas de modéliser l'usage des mots inconnus<sup>d</sup>. Une solution triviale consiste à considérer que tous les mots de ce type sont peu fréquents. Si on considère comme lexique les mots les plus fréquents de  $V_{\text{train}}$  et qu'on y ajoute un symbole dédié aux mots peu fréquents, alors tout mot  $w$  même absent de  $V_{\text{train}}$  obtient de fait une représentation dédiée. Tous les mots peu fréquents sont ainsi considérés comme un seul et même *token*. Le prix de l'apparente simplicité de cette approche est une mauvaise gestion des mots peu ou pas observés, au profit de meilleures performances sur les mots plus fréquents allant de pair avec une meilleure robustesse générale du modèle. Cependant, la majorité des erreurs des outils d'analyses modernes, a lieu lorsqu'on considère des mots peu fréquents, pour lesquels on a peu d'exemples d'apprentissage, et qui donc conservent une certaine ambiguïté.

C'est en partie pour résoudre plus finement ce problème que la notion d'*embedding* a été développée ces dernières années. Cette approche consiste à considérer un corpus de taille importante  $C_{\text{emb}}$  sur lequel apprendre des représentations vectorielles denses de mots, en espérant que son lexique englobe celui du corpus d'entraînement, et celui de l'ensemble de test. On a ainsi accès à un premier corpus d'entraînement  $C_{\text{emb}}$  sur lequel est calculé les *embeddings*, et un second de taille plus modeste mais annoté  $C_{\text{task}}$ . Ce dernier est divisé en deux sous-corpus,  $C_{\text{train}}$  servant à l'apprentissage de la tâche considérée, et un ensemble d'exemples de test  $C_{\text{test}}$  dédié à l'évaluation du modèle. De cette hétérogénéité de corpus découle une généralisation du problème des mots inconnus – ou *Out Of Vocabulary words* - OOV – uniquement présents dans un sous-ensemble des corpus utilisés.

---

d. ou du moins font l'hypothèse que cet usage ne dépend que de la morphologie

## 4.4.2 Généralisation

Le fait d'utiliser deux corpus d'entraînement distincts, l'un pour apprendre les représentations  $C_{emb}$ , l'autre pour entraîner le réseau  $C_{train}$ , génère différents type de mots hors vocabulaire. Si on considère un mot  $w$  apparaissant dans un exemple de test, il peut être absent de l'ensemble d'apprentissage, de l'espace d'*embedding* ou des deux. Ce constat est d'autant plus flagrant lorsqu'on considère des corpus de domaine ou de registre de langue très différents, ce qui ne fait qu'augmenter la propension des vocabulaires associés à se distinguer. Dans la figure 4.3, la région  $w_1$  représente les mots connus du corpus de test, présents dans le corpus d'entraînement et associés à une représentation dédiée dans l'espace de *embeddings*. La région  $w_2$  représente les mots présents dans l'espace des *embeddings* mais absents du corpus d'entraînement ( $OOV_{train}$ ). Symétriquement,  $w_3$  correspond aux mots, certes présents dans le corpus d'entraînement, mais n'étant associés à aucun vecteur de l'espace d'*embedding* ( $OOV_{emb}$ ). Enfin,  $w_4$  fait référence aux mots uniquement présents dans le corpus de test ( $OOV_{all}$ ).

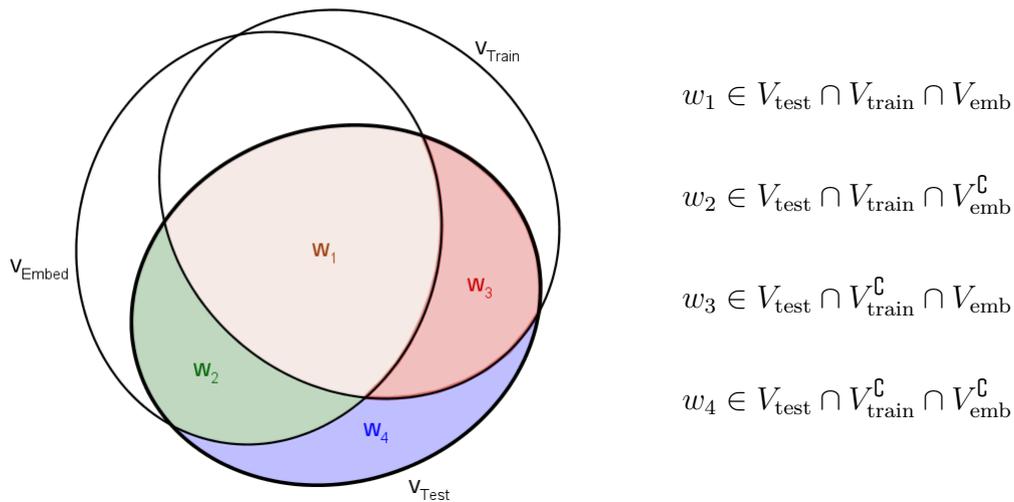


Figure 4.3 – Classification des mots inconnus : l'ensemble de test  $V_{test}$  est composé de mots connus (région  $w_1$ ), de trois types de mots inconnus (régions  $w_2$ ,  $w_3$  et  $w_4$ )

Considérons un cas pratique afin d'étudier la répartition de ces différents mots hors vocabulaire. Deux corpus seront plus tard utilisés lors des expérimentations. D'une part, une extraction automatique du contenu français de Wikipedia, contenant environ 357 millions de mots, est utilisée comme base d'apprentissage générique de nos représentations distributionnelles  $C_{emb}$ . D'autre part, le RATP-DECODA corpus, contenant environ 600 000 mots, est composé de transcriptions

manuelles de conversations téléphoniques annotées. Il s’agit d’un jeu de données spécifique à la tâche et au domaine considérés, dissocié pour les besoins de l’apprentissage en un ensemble d’entraînement  $C_{\text{train}}$  de 575 000 mots, et un ensemble de test  $C_{\text{test}}$  de 25 000 mots. Afin d’étudier la distribution d’OOV dans cet ensemble de test, on considère des sous-ensembles de taille croissante de notre ensemble d’entraînement, de  $D_0$  à  $D_9$ . Comme on peut s’y attendre, initialement la proportion d’ $OOV_{\text{train}}$  est sans pareille lorsqu’on considère un petit ensemble d’apprentissage, avant de décroître rapidement lorsqu’on augmente progressivement la taille de ce dernier. Ce qu’on observe de moins trivial, c’est que la somme des  $OOV_{\text{emb}}$  et des  $OOV_{\text{all}}$  reste constante. Lorsqu’on augmente progressivement la taille du corpus d’apprentissage, un nombre croissant de mots initialement absents de  $C_{\text{emb}}$  voient leur première occurrence apparaître dans  $C_{\text{train}}$ . Cela signifie qu’une part des  $OOV_{\text{all}}$  deviennent progressivement des  $OOV_{\text{emb}}$ .

$C_{\text{train}}$	$ C_{\text{train}} $	$w_2 - OOV_{\text{train}}$		$w_3 - OOV_{\text{emb}}$		$w_4 - OOV_{\text{all}}$	
$D_0$	1,667	4601	19.27%	1261	5.28%	1250	5.24%
$D_1$	11,273	1951	8.17%	1814	7.60%	697	2.92%
$D_2$	23,752	1381	5.78%	2013	8.43%	498	2.09%
$D_3$	65,057	406	1.70%	2308	9.67%	203	0.85%
$D_4$	151,910	406	1.70%	2308	9.67%	203	0.85%
$D_5$	230,950	347	1.45%	2354	9.86%	157	0.66%
$D_6$	311,400	312	1.31%	2371	9.93%	140	0.59%
$D_7$	387,689	291	1.22%	2379	9.96%	132	0.55%
$D_8$	477,729	270	1.13%	2391	10.01%	120	0.50%
$D_9$	576,056	244	1.02%	2403	10.06%	108	0.45%

Table 4.7 – Distribution des OOV dans  $C_{\text{test}}$  en fonction de différentes partitions de  $C_{\text{train}}$ , le corpus  $C_{\text{emb}}$  étant fixé.

La gestion des mots inconnus est un problème important auquel doit répondre tout système de traitement automatique de la langue. Comme mentionné précédemment, l’approche générique consiste à extraire systématiquement des *features* morphologiques mais d’autres solutions, plus spécifiques, existent. Par exemple, les étiqueteurs en POS recourent souvent à un *guesser* morphologique : un *classifieur* entraîné à prédire une classe grammaticale uniquement à partir de *features* morphologiques. Dans le cas spécifique où on utilise un espace d’*embedding*, il est possible d’utiliser un point arbitraire de l’espace comme représentation d’un mot hors vocabulaire. Enfin, si l’espace d’*embedding* considéré est issu d’un algorithme de type *Skip-Gram*, il est envisageable d’utiliser le contexte courant d’un mot inconnu en entrée du *Skip-Gram* afin de lui fournir un *embedding*.

### 4.4.3 Stratégies envisagées

Cette section détaille les stratégies que nous avons envisagées pour gérer chacun de ces trois types de mots problématiques. Il est important de noter qu'ils n'ont que deux causes possibles (absence de  $C_{\text{emb}}$ , absence de  $C_{\text{train}}$ ) et on se propose de décrire une stratégie propre à chacune d'elle.

S'affranchir des mots  $w \notin C_{\text{emb}}$  (ou  $OOV_{\text{emb}}$ ) absents de l'espace de représentation est possible en leur fournissant une représentation initiale. La stratégie envisagée afin d'obtenir cette représentation approximée n'a qu'un faible impact, lorsque suffisamment de données d'apprentissage sont disponibles, car la représentation choisie sera modifiée et optimisée lors de l'apprentissage. Elle est en revanche primordiale lorsqu'on se trouve dans le cas de langues, de domaines ou de registres peu dotés en terme de corpus annotés. Nous avons choisi d'utiliser la représentation associée à un mot apparaissant effectivement dans  $C_{\text{train}}$ , "proche" de notre mot inconnu. Nous développerons par la suite le critère de similarité distributionnelle que nous avons envisagé.

S'affranchir des mots  $w \notin C_{\text{train}}$  (ou  $OOV_{\text{train}}$ ) est une tâche plus ardue car nous aimerions approximer avec précision l'*embedding* raffiné qu'aurait eu ce mot, s'il était apparu dans l'ensemble d'apprentissage. Nous avons choisi d'approximer le raffinement d'un mot inconnu à partir des raffinages de mots "proches" dans l'espace d'*embedding* initial et effectivement raffinés lors de l'apprentissage car apparaissant dans  $C_{\text{task}}$ . L'hypothèse sous-jacente est que l'apprentissage modifie l'espace d'*embedding* sans créer de discontinuités. Il est donc possible d'approximer localement les effets du raffinement. Afin de raffiner artificiellement la représentation vectorielle d'un mot absent du corpus d'apprentissage, on se propose d'appliquer à sa représentation initiale  $\Phi_0(w)$  la moyenne des transformations de ses plus proches voisins dans l'espace d'*embedding* initial.

Enfin, afin de s'affranchir des mots  $w \notin C_{\text{train}} \cup C_{\text{emb}}$  (ou  $OOV_{\text{all}}$ ), on applique successivement les deux méthodes précédentes, afin d'approximer une représentation initiale avant de la raffiner artificiellement.

#### 4.4.3.1 Cas des mots $\notin C_{\text{emb}}$ : Out-of-Embedding

Si l'espace de représentation utilisé pour les mots est établi sur un corpus donné, alors tout mot en étant absent n'aura initialement pas de représentation. On se propose d'approximer la représentation qu'aurait eu ce mot s'il avait été présent lors de l'apprentissage des *embeddings*. Dans cette section, nous considérons un  $OOV_{\text{emb}}$  : un mot  $w$  apparaissant dans l'ensemble de test  $C_{\text{test}}$  mais

absent du corpus ayant servi à l'apprentissage des représentations distributionnelles.  $w \notin C_{\text{emb}}$  signifie qu'on n'a pas accès à une représentation convenable de  $w$  encodée dans un *embedding* dédié. Même si  $w$  apparaît effectivement dans l'ensemble d'apprentissage  $C_{\text{train}}$ , ce phénomène conduit à des erreurs proches des inconsistances statistiques qu'on observe dans les modèles probabilistes lorsqu'on s'intéresse à un événement ponctuel absent de l'ensemble ayant servi de base aux calculs des probabilités. Il est donc nécessaire de fournir une représentation pertinente à de tels mots.

Ceci peut, par exemple, être réalisé en utilisant une représentation commune, associée à tous les mots inconnus  $OOV_{\text{emb}}$  susceptibles d'être rencontrés dans  $C_{\text{train}} \cup C_{\text{test}}$ . Cette représentation peut aussi bien être fixée *a priori* ou apprise à partir des mots  $w \in C_{\text{emb}}$  ayant une faible fréquence d'apparition dans le corpus servant à l'apprentissage des *embeddings* (COLLOBERT, WESTON, BOTTOU et al. 2011). En plus d'être l'idée la plus simple, l'intuition sous-jacente est de tirer profit du fait qu'un  $OOV_{\text{emb}}$  est *a priori* peu fréquent, sans quoi on l'aurait justement observé. On espère qu'en regroupant les mots peu fréquents, on minimisera l'effort de généralisation demandé au réseau, afin de traiter ces derniers. Par exemple, si la langue, le domaine et le registre s'accordent pour que les énoncés contiennent énormément d'allusions à beaucoup de noms propres, alors, considérer un mot inconnu préférentiellement comme un nom propre, a du sens. Une autre alternative est d'assigner une représentation individuelle, initialisée aléatoirement, à chaque mot  $w \notin C_{\text{emb}}$ . Les méthode d'apprentissage de représentations n'étant réellement effectives que si on a accès à un certain nombre d'occurrences d'un mot considéré, il n'est pas aberrant de considérer que la représentation d'un mot, n'apparaissant pas de manière significative dans le corpus de référence, est essentiellement déterminé par l'initialisation aléatoire du réseau en charge de l'apprentissage de ces représentations.

On se propose de mettre au point une méthode basée sur l'hypothèse distributionnelle qui stipule que des mots apparaissant dans des contextes similaires ont des fonctions syntaxiques et/ou sémantiques similaires. Nous calculons des profils d'apparition des mots dans tous les contextes possibles. L'idée est d'utiliser, pour un mot  $w \notin C_{\text{emb}}$  inconnu, l'*embedding* du mot ayant un profil proche. Dans un premier temps, on recherche et cumule toutes les occurrences de  $w$  dans  $C_{\text{train}} \cup C_{\text{test}}$ . À l'aide des contextes des occurrences collectées, on détermine  $m \in C_{\text{emb}}$  le mot le plus "proche" de  $w$  au sens d'une mesure de similarité. Il est ainsi possible d'utiliser l'*embedding* de  $m$  afin de représenter  $w$ . La mesure de similarité que nous avons envisagée est définie comme la similarité empirique entre les distributions des occurrences d'un mot dans tous les contextes possibles. Un contexte d'occurrence est alors défini comme la suite des  $K$  mots précédents  $c_p$  et des  $K$  mots suivants  $c_s$ . Plus formellement, on considère un mot  $w$  et toutes ses occurrences dans tous les contextes possibles comme la distribution probabi-

liste des  $N$ -grammes centrés sur  $w$ , avec  $N = 2K + 1$ . Cette distribution est définie comme  $\{P_w(c_p, c_s), \forall (c_p, c_s) \in C_{\text{train}}^{2K}\}$  avec :

$$\forall (c_p, c_s) \in C_{\text{train}}^{2K} P_w(c_p, c_s) = P(\langle c_p, w, c_s \rangle | w) = \frac{\text{count}\langle c_p, w, c_s \rangle}{\text{count}\langle w \rangle} \quad (4.1)$$

La similarité entre deux mots  $u$  et  $v$  est ensuite déterminée comme la *divergence de Kullback-Leibler* entre leur distribution d'occurrences respective. L'*embedding* d'un mot  $w \notin C_{\text{emb}}$  est initialisé à la valeur de l'*embedding* du mot lui étant le plus semblable  $m = \underset{u \in C_{\text{emb}} \cap C_{\text{train}}}{\text{argmin}} D_{KL}(P_w || P_u)$ .

$$D_{KL}(P_u || P_v) = \sum_{c_p, c_s} P_u(c_p, c_s) \log \frac{P_u(c_p, c_s)}{P_v(c_p, c_s)} \quad (4.2)$$

On a ainsi trouvé un *embedding* de substitution pour notre mot inconnu  $w \notin C_{\text{emb}}$  absent de l'espace de représentation initial. Cela revient à confondre  $w$  et  $m$  lors de l'initialisation, tout en permettant à leurs représentations associées d'être optimisées et raffinées indépendamment, lors de l'apprentissage sur  $C_{\text{train}}$ . La principale limitation de cette approche reste toutefois l'estimation de  $P_u$  et  $P_v$  quand on a peu de données. Nos expérimentations ont montré que, si le mot inconnu  $w$  était trop peu fréquent dans  $C_{\text{task}}$  (moins de 5 occurrences), alors le mot  $m$  prédit devenait arbitraire, généralement au centre de l'espace tel que "et" ou "hum". Il s'agit d'un problème particulièrement impactant pour les  $OOV_{\text{all}}$  (uniquement observable dans  $C_{\text{test}}$ ). Une alternative, probablement efficace dans ce cas de figure, nécessite que les *embeddings* initiaux proviennent d'une approche *Skip-gram*. Il est alors possible d'utiliser le contexte courant des occurrences de  $w$  observées dans  $C_{\text{task}}$ . En injectant ces contextes en entrée du réseau *Skip-gram* entraîné, ce dernier prédit alors un mot de substitution effectivement présent dans  $C_{\text{emb}}$ , dont l'*embedding* peut-être utilisé pour représenter notre mot inconnu. Enfin, dans le cas où on n'a pas d'exemples, mais juste le mot, la morphologie pourrait aider à obtenir une bonne représentation. Une approche simple consisterait à utiliser la *distance de Levenstein* sur les caractères afin de choisir un mot connu de substitution. Une autre solution considère l'utilisation d'une couche d'entrée *character-level* (SANTOS et ZADROZNY 2014). La séquence de caractères composant un mot est utilisée en entrée d'un DNN dont la sortie sert de représentation pour le mot dans une architecture *word-level* classique (LABEAU et ALLAUZEN 2017).

#### 4.4.3.2 Cas des mots $\notin C_{\text{train}}$ : Out-of-Train

La méthode de raffinement des *embeddings* exposée précédemment est uniquement applicable à des mots effectivement présents dans le corpus d'apprentissage spécifique à la tâche d'intérêt  $C_{\text{train}}$ . Ainsi, les représentations associées à tout mot, n'apparaissant dans aucun exemple d'apprentissage, n'est jamais remise en cause par l'algorithme d'optimisation par rétro-propagation du gradient, et reste donc parfaitement inchangées au cours de l'apprentissage. Ce phénomène conduit à un espace de représentation où certains *embeddings* ont été modifiés, raffinés, tandis que d'autres sont restés inchangés. On se propose ici de raffiner artificiellement ces mots inconnus  $w \notin V_{\text{train}}$  en espérant ainsi conserver certaines propriétés de compositionnalité et de linéarité de l'espace d'*embedding*, perdues lors des transformations discontinues induites par l'optimisation par rétro-propagation du gradient. L'idée sous-jacente est d'approximer la modification qu'aurait subi la représentation  $\Phi_0(w)$ , associée initialement au mot  $w$ , s'il avait été présent au cours de l'apprentissage spécifique à la tâche. Nous faisons l'hypothèse que si un mot  $w$  était effectivement apparu dans l'ensemble d'apprentissage  $C_{\text{train}}$ , il aurait subi un raffinement similaire à celui subi par ses proches voisins dans l'espace d'*embedding* initial. Cette hypothèse s'appuie sur le fait qu'on considère alors le raffinement comme un morphisme permettant de passer de  $\mathcal{E}_0$  à  $\mathcal{E}_r$ .

On peut approximer le raffinement qu'aurait subi  $\Phi_0(w)$ , l'*embedding* initial de  $w$ , à l'aide des  $K$  plus proches voisins de  $w$  dans l'espace d'*embedding* initial  $\mathcal{E}_0$  effectivement raffinés i.e présents dans  $C_{\text{train}}$ . On note  $(\eta_k)_{k=1..K}$  les  $K$  mots  $\in V_{\text{emb}} \cap V_{\text{train}}$  dont les représentations initiales associées  $(\Phi_0(\eta_k))_{k=1..K}$  sont les plus proches de  $\Phi_0(w)$ . On considère que ces représentations vectorielles ont subi une translation  $(\Phi_r(\eta_k) - \Phi_0(\eta_k))$  au cours de l'apprentissage. On se propose d'approximer la translation qu'aurait subi  $w$  comme une somme pondérée des transformations subies par ses plus proches voisins dans la distribution initiale  $\mathcal{E}_0$  de l'espace d'*embedding* :

$$\Phi_r(w) = \Phi_0(w) + \sum_{k=1}^K \alpha_k (\Phi_r(\eta_k) - \Phi_0(\eta_k)) \quad (4.3)$$

Les coefficients de pondération  $\alpha$  sont des scalaires réels positifs sommant à 1. Dans le cadre de nos expérimentations, bien qu'aucun gain substantiel n'ait été observé vis-à-vis d'une somme uniforme, nous avons défini ces coefficients comme proportionnels à la *similarité cosine* entre  $w$  et  $\eta_k$  i.e au cosinus de l'angle formé par les vecteurs  $\Phi_0(w)$  et  $\Phi_0(\eta_k)$  :

$$\alpha_k \propto s(w, \eta_k) = \frac{\Phi_0(w) \cdot \Phi_0(\eta_k)}{|\Phi_0(w)| \times |\Phi_0(\eta_k)|} \quad (4.4)$$

Il ne s'agit là que de propositions *ad hoc* afin de servir cette étude. On sait que l'hypothèse selon laquelle les mots proches dans l'espace d'*embedding* initial subissent un raffinement similaire est fautive. C'est par exemple le cas de l'analyse de sentiment où *good* et *bad* sont proches dans l'espace d'origine mais lointains dans l'espace *fine-tuned*. D'ailleurs, d'autres méthodes d'approximation (linéaire, MLP) sont possibles. En réalité, on s'intéresse essentiellement à jauger la viabilité et la difficulté du problème de raffinement artificiel des représentations d'entrée d'un réseau de neurones.

#### 4.4.4 Application

Dans cette partie, on cherche à évaluer l'impact de nos stratégies de gestion des mots hors-vocabulaire. Plus précisément, on étudie l'impact de nos stratégies sur les performances en détection de cadres sémantiques sur les transcriptions manuelles du corpus RATP-DECODA en fonction du volume de données d'apprentissage disponibles. Nos résultats expérimentaux sont rassemblés dans la figure 4.4. On y compare six systèmes. D'une part, deux *Conditional Random Field*, l'un ne considérant que les séquences mots dans une approche *End-to-End* tandis que l'autre a accès à des *features* syntaxiques supplémentaires (Part-Of-Speech, Named Entities). Et d'autre part, quatre réseaux de neurones, le premier considérant un espace d'*embedding* initial aléatoire, le deuxième ayant accès à un espace d'*embedding* initial appris sur  $C_{emb}$  et les derniers implémentant nos stratégies d'adaptation de cet espace d'*embedding*.

- **CRF** est un *Conditional Random Field* état de l'art entraîné à la détection de cadre sémantique en *End-to-End* i.e. en ne considérant que les mots. Il utilise un contexte lexical sous forme d'une fenêtre de 5 mots pour prédire la meilleure séquence d'étiquette *FrameNet*. Ce CRF entraîné à l'aide de *crf\_suite* (OKAZAKI 2007) fait partie du *pipeline MACAON NLP tool suite* (A. NASR, F. BÉCHET, REY et al. 2011) dont la partie dédiée à l'analyse syntaxique a été adaptée au traitement de l'oral (BÉCHET, Alexis NASR et FAVRE 2014).
- **CRF++** correspond au même CRF, ayant cette fois accès à des annotations morpho-syntaxiques et relatives aux entités nommées. Ces *features* sont prépondérantes dans la bonne gestion des mots hors-vocabulaire. Elles permettent de passer outre les écueils statistiques, en fournissant au CRF des *features* auxquelles se rattacher dans le cas de l'analyse d'un mot ou d'une structure absents des données d'entraînement.
- **NN** et **Rand** sont tous les deux des réseaux de neurones. Le premier ayant

accès à un espace d'*embedding* calculé sur  $C_{emb}$ . Le second utilisant cette fois un espace d'*embedding* initialisé aléatoirement, plutôt qu'appris de façon non-supervisée. À ce stade, on ne considère aucune stratégie d'adaptation, et les mots hors-vocabulaire sont associés à des représentations vectorielles aléatoires.

- **NN++** et **Rand++** intègrent tous les deux nos stratégies d'adaptation de l'espace d'*embedding*, décrites dans la section 4.4.3.

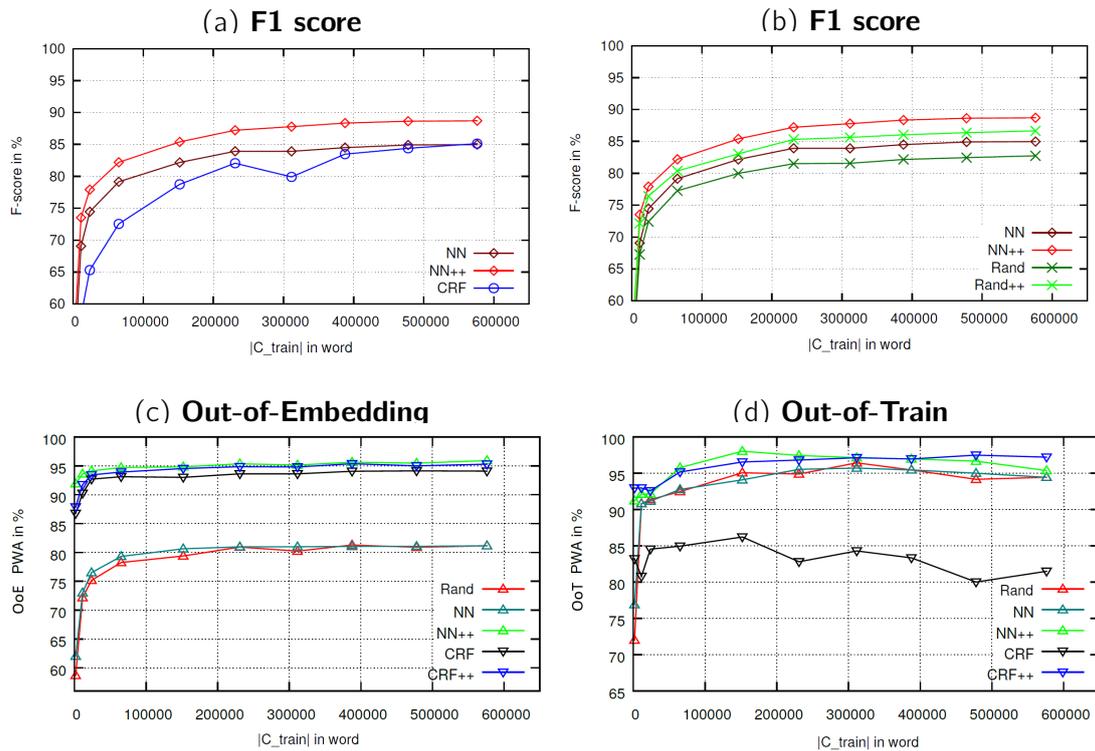


Figure 4.4 – Évaluations (*F1*-score et *PWA* restreintes aux **OoE** et aux **OoT**) en fonction du volume de données disponibles durant l'apprentissage.

Tout d'abord, si on se concentre sur les courbes invoquant le **CRF** et le **NN** dans la figure 4.4, on constate qu'ils affichent des performances comparables (cf.4.4a). Cependant, d'après les résultats restreints aux mots hors-vocabulaire, ils ne commettent pas les mêmes erreurs. En effet, le **CRF** a des difficultés à appréhender les mots absents de son corpus d'entraînement (cf.4.4d). Tandis que le **NN** peine à traiter de rares représentations aléatoires qui se seraient glissées dans un espace de représentation cohérent (cf.4.4c). Enfin, le réseau de neurones, intégrant nos stratégies d'adaptation des *embeddings*, domine l'ensemble des autres modèles en compétition. Le gain est particulièrement significatif dans

le cas où on a accès à peu de données d’entraînement, mais on observe tout de même des améliorations mineures lorsque la totalité du corpus est utilisé lors de l’apprentissage.

Lorsque la totalité du corpus d’apprentissage ( $D_9$ ) est considérée, le CRF entraîné en *End-to-End*, c’est-à-dire ne considérant que des mots en entrée, affiche une F-mesure de 84,6% en détection de cadres sémantiques. Il est à peine dépassé (85,1%) par son homologue ayant, quant à lui, accès à des *features* supplémentaires (annotations POS et NE). L’impact de ces *features* sur les performances est particulièrement visible si on s’intéresse aux performances restreintes aux mots inconnus  $w \notin C_{\text{task}}$  (cf. figure 4.4d). L’ajout de ces *features* additionnelles permet au CRF de gérer convenablement 97,2% des mots inconnus, contre seulement 81,5% en *End-to-End*. En d’autres termes, ces *features* permettent au CRF de mieux généraliser ses prédictions concernant les mots inconnus absents du corpus d’apprentissage *i.e.* pour lesquels le CRF *End-to-End* n’a aucune information statistique pertinente. Ce phénomène est particulièrement marqué lorsque peu de données d’entraînement sont considérées, ce qui augmente la proportion de mots inconnus dans l’ensemble de test.

L’initialisation de l’espace d’*embedding* à l’aide de représentations calculées sur  $C_{\text{emb}}$  (cf. figure 4.4b) ne s’accompagne de gains substantiels que dans le cas où peu de données d’apprentissage sont utilisées *i.e.* dans le cas où les procédures d’initialisation des paramètres internes du réseau influencent encore les performances du modèle. En effet, lorsque suffisamment de corpus d’apprentissage est disponible, le processus de raffinement des représentations semble compenser l’absence de représentations initiales. Les effets de cette initialisation sont donc particulièrement visibles si on se concentre sur les mots inconnus  $w \notin C_{\text{task}}$  (cf. figure 4.4d) puisqu’il s’agit des mots dont les *embeddings* ne sont pas raffinés durant l’apprentissage. De nouveau, l’amélioration observée dans ces conditions n’est significative que dans le cas où très peu de données sont considérées en entraînement ( $D_0$ ) *i.e.* dans le cas où la proportion de mots  $w \notin C_{\text{task}}$  est maximale dans l’ensemble de test.

En revanche, l’application de nos stratégies d’adaptation conduit à une amélioration des performances passant de 86,6% à 90,2% de F-mesure, lorsque la totalité du corpus d’apprentissage ( $D_9$ ) est considérée. Cette amélioration est du même ordre de grandeur que celle observée dans le cas des CRF. L’impact de nos stratégies d’adaptation est particulièrement visible si on s’intéresse aux performances restreintes aux mots inconnus  $w \notin C_{\text{emb}}$  (cf. figure 4.4c) c’est-à-dire sans *embedding* initial. L’adaptation de l’espace d’*embedding* permet au réseau de neurones de gérer convenablement 95,9% des mots sans représentation initiale contre seulement 81,2% lorsque aucune stratégie d’adaptation n’est considérée. Fournir de meilleures représentations initiales à ces mots améliore significative-

ment les performances du réseau, même dans le cas où la totalité du corpus est utilisé. Ceci est probablement dû au fait que les OOE sont en général des mots peu fréquents. La fréquence d'un mot dans le corpus d'apprentissage va influencer le nombre de fois où son *embedding* associé sera raffiné. Par conséquent, les *embeddings* de mots peu fréquents (dont les OOE) sont peu modifiés durant l'apprentissage, même lorsque la totalité du corpus est considérée. Autrement dit, les performances restreintes à ces mots sont fortement impactées par les procédures d'initialisation des représentations.

Bien que les modèles *baselines* CRF et NN affichent tous deux des performances globales comparables, on se rend compte, en observant leur comportement respectif sur les différentes classes de mots inconnus, qu'ils ne commettent pas les mêmes erreurs. Le *Conditional Random Field* ne considérant que les mots  $w \in C_{\text{task}}$  a du mal à généraliser ses prédictions aux mots  $w \notin C_{\text{task}}$ . On doit pour cela faire appel à des *features* additionnelles afin de pallier l'absence d'occurrence de référence. Similairement, le réseau de neurones sans stratégie d'adaptation de l'espace de représentation a des difficultés à appréhender des mots  $w \notin C_{\text{emb}}$  sans *embedding* initial. Fournir de meilleures représentations initiales à ces mots améliore significativement les performances du réseau, même dans le cas où la totalité du corpus est utilisé. Cependant, l'utilisation de *features* additionnelles dans le cas du CRF requiert nécessairement des annotations additionnelles généralement coûteuses, là où l'apprentissage et l'adaptation d'un espace d'*embedding* dans le cas du réseau de neurones requièrent uniquement d'importantes quantités de corpus *blanc*, disponibles en abondance de nos jours.

## Conclusion

Dans ce chapitre, nous proposons d'appliquer le modèle multitâche décrit dans le chapitre précédent à l'analyse de transcriptions manuelles et automatiques de conversations téléphoniques. On y a présenté le corpus RATP-DECODA de transcriptions provenant d'un centre d'appel des transports publics de la ville de Paris. Ce corpus contient un très grand nombre de phénomènes spécifiques à la langue orale tels que les disfluences (répétitions, marqueurs de discours, faux-départs, etc.). À cause de cet important degré de spontanéité dans les énoncés à traiter, les modèles d'étiquetage morpho-syntaxique ou d'analyse en dépendances syntaxiques, développés à partir de corpus de textes écrits canoniques, sont difficilement applicables en l'état. De plus, quand les transcriptions automatiques contiennent trop d'erreurs, même les systèmes d'analyse les plus robustes peuvent échouer. Ces détériorations de performances sont particulièrement observables dans le cas des approches *pipelines*, courantes en traitement automa-

tique de la langue.

Concernant les transcriptions manuelles, nous avons réalisé une évaluation comparative entre des BiRNNs monotâches *End-to-End*, un pipeline de BiRNNs et l'approche multitâche sur cinq analyses : l'étiquetage morpho-syntaxique, la détection de disfluences, la reconnaissance d'entités nommées, l'analyse en dépendances syntaxiques et la détection de cadres sémantiques. Ces résultats montrent que le modèle **MT:BiRNN** égale, voire surpasse, le **E2E:BiRNN** et le **PL:BiRNN** sur l'ensemble de ces tâches. Lorsqu'on étudie l'impact de la quantité de données d'apprentissage sur les performances, on constate qu'un volume de données minimal est nécessaire afin que le modèle multitâche égale puis surpasse les modèles *End-to-End*. On peut, d'une part, interpréter ce résultat en considérant que certaines tâches sont plus "difficiles" que d'autres *i.e.* nécessitant davantage de données afin d'obtenir un *classifier* performant. D'autre part, il est possible d'imaginer que ces quantités de données minimales sont corrélées avec les poids  $\beta$  du critère d'apprentissage complexe menant à un apprentissage multitâche optimal.

Nous avons étudié par la suite les performances en détection de *frames* sémantiques sur les transcriptions automatiques. On évalue cinq systèmes d'analyse sémantique : Deux modèles *End-to-End* (**CRF** et **BiRNN**) ; Deux *pipelines* (**PL:CRF** et **PL:MACAON**) et enfin le modèle d'analyse multitâche (**MT:BiRNN**). Les deux approches BiRNN obtiennent de meilleures performances (en particulier un meilleur rappel) que les autres modèles mis en jeu, tout en offrant une meilleure flexibilité. Ils surpassent même notre modèle de référence MACAON, qui est pourtant à l'origine des annotations utilisées en apprentissage. Dans le cas du **MT:BiRNN**, il est possible d'améliorer les performances d'une tâche en particulier en déterminant la distribution de poids  $\beta$  du critère d'apprentissage multitâche, maximisant les performances obtenues sur l'ensemble de validation. Puis, on réalise une analyse en fonction du taux d'erreur mot des transcriptions automatiques. Bien que les gains restent ténus, on observe que le modèle multitâche est plus robuste que les *pipelines* aux erreurs induites par les systèmes ASR. Il serait intéressant d'étudier l'apport respectif des *embeddings*, de l'architecture récurrente et de l'algorithme d'apprentissage multitâche, afin de déterminer précisément la source du gain. Cependant, si notre approche est plus robuste aux erreurs de transcriptions, elle ne cherche nullement à les corriger et une perspective directe consisterait à inclure l'ASR dans l'ensemble des tâches.

Enfin, notre modèle repose sur le partage de représentations et donc notamment sur la bonne qualité des *embeddings* utilisés. Lorsqu'on considère des petits corpus de langue non-canonique en apprentissage, les proportions de mots hors-vocabulaire (OOV) ont tendance à exploser. Nous nous sommes penchés sur le problème particulier de l'adaptation de l'espace de représentation dans lequel sont plongés les mots, dans le cas où une part importante des termes, propres au

domaine et au registre de langue, n'ont pas de représentation initiale de bonne qualité. Nos stratégies d'adaptation peuvent se résumer en la recherche d'une meilleure représentation à fournir au réseau, lorsqu'on en possède pas initialement (OOE) ou que cette représentation ne subit pas le raffinement naturel de l'espace d'*embedding*, car elle est associée à un mot n'apparaissant pas dans le corpus d'entraînement (OOT). Ce type de stratégie d'adaptation peut être utile lorsque peu de données sont disponibles afin d'entraîner le modèle *i.e.* lorsque les proportions de ces mots inconnus sont maximales dans l'ensemble de test.

# Conclusion

Les objectifs de cette thèse étaient l'analyse, la conception, le développement et l'implémentation de modèles issus de l'apprentissage automatique et plus spécifiquement du *Deep Learning*, appliqués à la résolution de tâches d'analyse linguistique du Traitement Automatique de la Langue. Les systèmes traditionnels du TAL considèrent une analyse comme une hiérarchie de sous-problèmes. L'analyse est réalisée par des *pipelines* de modèles spécifiques à chaque tâche. Ces modèles sont généralement basés sur des heuristiques d'apprentissage automatique supervisées (CRF, SVM) s'appuyant sur une sélection de *features* réalisée par un expert. Ces chaînes de traitement présentent un certain nombre de limitations : la dépendance des modèles à la sélection empirique des traits, le cumul des erreurs dans le *pipeline* et la sensibilité au changement de domaine. Ces limitations peuvent conduire à des pertes de performances particulièrement importantes, lorsqu'il existe un décalage entre les conditions d'apprentissage des modèles et celles d'utilisation. Un tel décalage existe lors de l'analyse de transcriptions automatiques de parole spontanée, comme, par exemple, les conversations téléphoniques enregistrées dans des centres d'appels. En effet, l'analyse d'une langue non-canonique pour laquelle il existe peu de données d'apprentissage, la présence de disfluences et de constructions syntaxiques spécifiques à l'oral ainsi que la présence d'erreurs de reconnaissance dans les transcriptions automatiques, mènent à une détérioration importante des performances des systèmes d'analyse. C'est dans ce cadre que se déroule cette thèse, en visant à mettre au point des systèmes d'analyse à la fois robustes et flexibles permettant de dépasser les limitations des systèmes actuels.

Appliquer les méthodes du *Deep Learning* au Traitement Automatique de la Langue est devenu omniprésent dans la littérature. L'apprentissage automatique à l'aide de réseaux de neurones profonds se base sur l'entraînement de modèles composés de nombreuses couches de neurones, formant une fonction paramétrique. Il sont notamment appliqués à la réalisation de tâches de classification (étiquetage, reconnaissance de motif, prévision de séries temporelles, *etc.*). Il est ainsi devenu possible d'entraîner avec moins de supervision des réseaux neuronaux sur d'importants volumes de données, qui généralement égalent, voire surpassent, les méthodes probabilistes classiques. Les modèles issus de ces recherches se sont montrés tout aussi efficaces que les systèmes traditionnels sur les tâches d'analyse syntaxique et sémantique. Notre travail propose de s'abstraire des limitations inhérentes aux modèles *pipelines* à l'aide de modèles issus du *Deep Learning*. Les réseaux de neurones peuvent notamment tirer profit des *word embeddings* afin d'augmenter leur capacité de généralisation et de se passer, en partie, de connaissances *a priori* des structures syntaxiques et sémantique

du langage. Nous avons tenté de montrer que des choix simples d'optimisation des architectures profondes, comme la procédure d'initialisation des poids, les paradigmes de l'algorithme d'optimisation considérés et les régularisations appliquées, jouent un rôle important dans la dynamique de l'apprentissage. Par exemple, il est possible de se passer de la procédure de pré-entraînement non-supervisée des *word embeddings*, à condition d'avoir suffisamment de données annotées. Cependant, à l'inverse, lorsque la quantité de données est insuffisante, il est primordial d'utiliser une méthode non-supervisée afin d'obtenir une représentation des données facilitant l'apprentissage.

Nous proposons d'utiliser un modèle multitâche, basé sur l'apprentissage de représentations communes, à la résolution d'un ensemble d'analyses syntaxiques du TAL. Notre approche se base sur un ensemble de *LSTM Bidirectional Recurrent Neural Networks* partageant une partie de leurs paramètres et capables de réaliser conjointement des tâches d'analyse syntaxique du texte. Ce modèle tire profit de corpus annotés pour différentes tâches. On a notamment illustré son efficacité sur des tâches d'analyse syntaxique à l'aide de corpus de référence composés de textes journalistiques annotés manuellement. Bien que notre approche multitâche n'offre pas d'amélioration significative, son avantage principal réside dans le fait de produire une annotation enrichie, à différent niveau d'analyse linguistique, et sans coût additionnel, à la différence de l'approche *pipeline* traditionnelle. Enfin, l'étude systématique de l'impact de la quantité de données disponibles en entraînement, que nous avons réalisée tout au long de ces évaluations, a permis de déterminer que notre approche est particulièrement performante lorsque peu d'exemples d'apprentissage sont disponibles. Cependant, un volume de données minimal est nécessaire afin que le réseau multitâche égale puis surpasse les modèles *End-to-End*.

Par la suite, ce modèle multitâche a été appliqué à l'analyse en cadres sémantiques du corpus RATP-DECODA de transcriptions manuelles et automatiques de conversations téléphoniques. Ce corpus contient un très grand nombre de phénomènes spécifiques à la langue orale tels que les disfluences (répétitions, marqueurs de discours, faux-départs, *etc.*). À cause de cet important degré de spontanéité dans les énoncés à traiter, les modèles d'étiquetage morpho-syntaxique ou d'analyse en dépendance syntaxiques, développés à partir de corpus de textes écrits canoniques, sont difficilement applicables en l'état. De plus, quand les transcriptions automatiques contiennent trop d'erreurs, même les systèmes d'analyse les plus robustes peuvent échouer. Ces détériorations de performances sont particulièrement observables dans le cas des approches *pipelines*. Nous avons ainsi pu montrer que notre approche multitâche présente une meilleure robustesse aux erreurs de transcription, induites par les systèmes ASR de reconnaissance de la parole.

Lorsqu'on considère des petits corpus de langue non-canonique en apprentissage, les proportions de mots hors-vocabulaire (OOV) ont tendance à exploser durant l'évaluation sur des données inconnues. Nous nous sommes penchés sur le problème particulier de l'adaptation de l'espace de représentation dans lequel sont plongés les mots, dans le cas où une part importante des termes propres au domaine et au registre de langue n'ont pas de représentation initiale de bonne qualité. Nos stratégies d'adaptation peuvent se résumer en la recherche d'une meilleure représentation à fournir au réseau, lorsqu'on en possède pas initialement (OOE) ou que cette représentation ne subit pas le raffinement naturel de l'espace d'*embedding* car elle est associée à un mot n'apparaissant pas dans le corpus d'entraînement (OOT). Ainsi, nous avons montré que ce type de stratégie d'adaptation peut être utile afin d'entraîner le modèle lorsque peu de données sont disponibles *i.e.* lorsque les proportions de ces mots inconnus sont maximales dans l'ensemble de test.

L'engouement de ces dernières années autour des applications du *Deep Learning* au TAL s'est accompagné d'un foisonnement de publications, d'outils et d'approches. Dans ces conditions, se maintenir à l'état de l'art devient une véritable gageure et une course contre la montre pour le doctorant s'attaquant à ces thèmes de recherche. Même si certains modèles étudiés en début de thèse sont maintenant, soit obsolètes, soit considérés comme étant de la pratique courante, toutes ces études successives ont chacune contribué à conforter mon appréhension de ce domaine dans toute sa complexité, jusqu'à l'obtention, début 2016, d'un système dont les performances et les propriétés étaient satisfaisantes.

Néanmoins de nombreuses pistes restent encore à explorer, notamment concernant l'ensemble des tâches et la distribution de poids  $\beta$  du critère d'apprentissage à considérer. En effet, de par la définition même de notre erreur globale sous forme de moyenne des erreurs individuelles, on est en droit de se poser la question suivante : *Existe-t-il un sous-ensemble de tâches secondaires menant à un apprentissage optimal d'une tâche principale ?*

Considérer des tâches supplémentaires pourrait permettre d'apprendre de meilleures représentations cachées (par exemple à l'aide d'un modèle de langage), de détecter des invariants naturels au sein des données (par exemple avec une tâche de segmentation en phrases) ou encore d'améliorer la robustesse du modèle (par exemple en réalisant conjointement la reconnaissance de la parole et les analyses linguistiques).

Une fois les tâches secondaires choisies, il conviendrait de déterminer la distribution  $\beta$  associée dans le critère d'apprentissage multitâche. À l'heure actuelle, il s'agit d'un hyper-paramètre du modèle. Cependant, on peut imaginer des heuristiques déterminant automatiquement cette distribution, à l'aide de critères tels

que l'évolution individuelle de l'erreur de chaque tâche au cours de l'entraînement, ou la quantité de données nécessaires au modèle multitâche afin de surpasser les performances du modèle monotâche.

Il est également envisageable d'enrichir l'architecture du réseau, notamment afin de prendre en compte les spécificités de certaines tâches. Par exemple, il est possible de prédire des séquences de taille variable à l'aide d'une approche *Sequence-to-Sequence* en ajoutant un RNN spécifique à une tâche, utilisé comme *decoder* et prenant en entrée (*via.* un modèle d'attention) les représentations cachées du BiRNN partagé. Il s'agirait d'une première étape permettant de rendre notre approche multitâche compatible avec des réseaux plus complexes tels que les *Pointer Networks*.

Réciproquement, on peut envisager d'utiliser en entrée du BiRNN l'historique d'un RNN *encoder* prenant en entrée des caractères, afin de produire automatiquement des représentations morphologiques des mots.

Enfin, on aurait pu considérer des apprentissages sur des corpus différents par tâche, dans le cas où on ne posséderait pas de corpus annoté pour l'ensemble des tâches d'intérêt. Le modèle actuel peut être adapté en considérant des distributions  $\beta$  propres à chaque exemple, en fonction des annotations disponibles. Il requiert toutefois une bonne couverture des lexiques des différents corpus, afin de favoriser la mutualisation des représentations entre les différentes tâches.

# Bibliographie

- [1] Dzmitry BAHDANAU, Jan CHOROWSKI, Dmitriy SERDYUK et al. « End-to-End Attention-based Large Vocabulary Speech Recognition ». In : *CoRR* abs/1508.04395 (2015) (cf. p. 51).
- [2] Collin F BAKER, Charles J FILLMORE et John B LOWE. « The berkeley frame-net project ». In : *Proceedings of the 17th international conference on Computational linguistics-Volume 1*. Association for Computational Linguistics. 1998, p. 86–90 (cf. p. 102).
- [3] P. BALDI et K. HORNIK. « Neural Networks and Principal Component Analysis : Learning from Examples Without Local Minima ». In : *Neural Netw.* 2.1 (jan. 1989), p. 53–58. ISSN : 0893-6080 (cf. p. 28).
- [4] Laura BANARESCU, Claire BONIAL, Shu CAI et al. « Abstract meaning representation (AMR) 1.0 specification ». In : *Parsing on Freebase from Question-Answer Pairs.* In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle : ACL. 2012, p. 1533–1544 (cf. p. 102).
- [5] Thierry BAZILLON, Melanie DEPLANO, Frederic BECHET et al. « Syntactic annotation of spontaneous speech : application to call-center conversation data. » In : *LREC*. 2012, p. 1338–1342 (cf. p. 104, 107).
- [6] Frederic BECHET, Benjamin MAZA, Nicolas BIGOUROUX et al. « DECODA : a call-centre human-human spoken conversation corpus ». In : *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*. 2012 (cf. p. 103).
- [7] Frederic BECHET, Alexis NASR et Benoit FAVRE. « Adapting dependency parsing to spontaneous speech for open domain spoken language understanding ». In : *Interspeech, Singapore*. 2014 (cf. p. 102, 104, 107, 122).
- [8] Yoshua BENGIO, Réjean DUCHARME, Pascal VINCENT et al. « A Neural Probabilistic Language Model ». In : *J. Mach. Learn. Res.* 3 (2003), p. 1137–1155 (cf. p. 51, 57, 63, 64).
- [9] Yoshua BENGIO, Patrice SIMARD et Paolo FRASCONI. « Learning Long-Term Dependencies with Gradient Descent is Difficult ». In : *IEEE Transactions on Neural Networks* 5.2 (1994), p. 157–166 (cf. p. 39, 76).
- [10] Bernd BOHNET. « Very high accuracy and fast dependency parsing is not a contradiction ». In : *Proceedings of the 23rd International Conference on Computational Linguistics*. 2010, p. 89–97 (cf. p. 56, 97).
- [11] Rich CARUANA. « Multitask Learning ». In : *Mach. Learn.* 28.1 (1997), p. 41–75 (cf. p. 73).

- [12] Christophe CERISARA, Pavel KRAL et Ladislav LENC. « On the Effects of Using word2vec Representations in Neural Networks for Dialogue Act Recognition ». In : *Computer Speech and Language* 47 (juil. 2017) (cf. p. 62).
- [13] Wanxiang CHE, Zhenghua LI, Yongqiang LI et al. « Multilingual Dependency-based Syntactic and Semantic Parsing ». In : *Proceedings of the Thirteenth Conference on Computational Natural Language Learning : Shared Task*. CoNLL '09. Association for Computational Linguistics, 2009, p. 49–54 (cf. p. 56, 97).
- [14] Jan K CHOROWSKI, Dzmitry BAHDANAU, Dmitriy SERDYUK et al. « Attention-Based Models for Speech Recognition ». In : *Advances in Neural Information Processing Systems* 28. 2015, p. 577–585 (cf. p. 41).
- [15] Dan Claudiu CIRESAN, Ueli MEIER, Jonathan MASCI et al. « Flexible, High Performance Convolutional Neural Networks for Image Classification ». In : *IJCAI*. 2011, p. 1237–1242 (cf. p. 16).
- [16] Maximin COAVOUX et Benoît CRABBÉ. « Comparaison d'architectures neuronales pour l'analyse syntaxique en constituants ». In : *TALN 2015*. Caen, France, 2015. URL : <https://hal.inria.fr/hal-01174613> (cf. p. 56).
- [17] COLLOBERT et WESTON. « A Unified Architecture for Natural Language Processing : Deep Neural Networks with Multitask Learning ». In : *Proceedings of the 25th International Conference on Machine Learning*. ICML '08. 2008, p. 160–167 (cf. p. 74–76, 80, 86).
- [18] COLLOBERT, WESTON, BOTTOU et al. « Natural Language Processing (Almost) from Scratch ». In : *the Journal of Machine Learning Research* 12. 2011, p. 2461–2505 (cf. p. 16, 51, 53–55, 57, 73, 91–93, 119).
- [19] Corinna CORTES et Vladimir VAPNIK. « Support-Vector Networks ». In : *Mach. Learn.* 20.3 (1995), p. 273–297. ISSN : 0885-6125 (cf. p. 33).
- [20] Koby CRAMMER et Yoram SINGER. « On the Algorithmic Implementation of Multiclass Kernel-based Vector Machines ». In : *J. Mach. Learn. Res.* 2 (mar. 2002), p. 265–292 (cf. p. 33).
- [21] Paul DELÉGLISE, Yannick ESTÈVE, Sylvain MEIGNIER et al. « The LIUM speech transcription system : a CMU Sphinx III-based system for French broadcast news. » In : *Interspeech*. 2005, p. 1653–1656 (cf. p. 107).
- [22] Trinh-Minh-Tri DO et Thierry ARTIERES. « Neural conditional random fields ». In : *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Sous la dir. d'Yee Whye TEH et Mike TITTE-RINGTON. T. 9. Proceedings of Machine Learning Research. Chia Laguna Resort, Sardinia, Italy : PMLR, 13–15 May 2010, p. 177–184 (cf. p. 57).

- [23] Daxiang DONG, Hua WU, Wei HE et al. « Multi-Task Learning for Multiple Language Translation ». In : *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1 : Long Papers*. 2015, p. 1723–1732 (cf. p. [52](#), [74](#), [76–78](#), [85](#), [86](#)).
- [24] John DUCHI, Elad HAZAN et Yoram SINGER. « Adaptive Subgradient Methods for Online Learning and Stochastic Optimization ». In : *J. Mach. Learn. Res.* 12 (juil. 2011), p. 2121–2159 (cf. p. [35](#)).
- [25] Charles J. FILLMORE. « FRAME SEMANTICS AND THE NATURE OF LANGUAGE ». In : *Annals of the New York Academy of Sciences* 280.1 (1976), p. 20–32 (cf. p. [105](#)).
- [26] Sylvain GALLIANO, Guillaume GRAVIER et Laura CHAUBARD. « The ester 2 evaluation campaign for the rich transcription of French radio broadcasts. » In : *INTERSPEECH*. ISCA, 2009, p. 2583–2586 (cf. p. [17](#)).
- [27] Daniel GILDEA et Daniel JURAFSKY. « Automatic Labeling of Semantic Roles ». In : *Comput. Linguist.* 28.3 (2002), p. 245–288 (cf. p. [101](#)).
- [28] Dan GILLICK, Oriol VINYALS et Amarmag SUBRAMANYA. « Multilingual Language Processing From Bytes ». In : (2015) (cf. p. [57](#)).
- [29] J. GIMÉNEZ et L. MÁRQUEZ. « SVMTool : A general POS tagger generator based on Support Vector Machines ». In : *In Proceedings of the 4th International Conference on Language Resources and Evaluation*. 2004, p. 43–46 (cf. p. [53](#)).
- [30] Yoav GOLDBERG. « A Primer on Neural Network Models for Natural Language Processing ». In : *CoRR* abs/1510.00726 (2015) (cf. p. [60](#)).
- [31] GRAFF, KONG, CHEN et al. « English gigaword ». In : *Linguistic Data Consortium, Philadelphia* (2003) (cf. p. [90](#)).
- [32] Alex GRAVES, Abdel-rahman MOHAMED et Geoffrey E. HINTON. « Speech Recognition with Deep Recurrent Neural Networks ». In : *CoRR* abs/1303.5778 (2013) (cf. p. [41](#)).
- [33] Jiang GUO. « Backpropagation through time ». In : (2013) (cf. p. [38](#), [43](#), [44](#)).
- [34] Dilek HAKKANI-TUR, Yun-Cheng JU, Geoffrey ZWEIG et al. « Clustering Novel Intents in a Conversational Interaction System with Semantic Parsing ». In : *Interspeech 2015 Conference*. 2015 (cf. p. [102](#)).
- [35] Zellig S HARRIS. *Distributional structure*. Springer, 1981 (cf. p. [61](#)).
- [36] Geoffrey E. HINTON, Simon OSINDERO et Yee-Whye TEH. « A Fast Learning Algorithm for Deep Belief Nets ». In : *Neural Comput.* 18.7 (juil. 2006), p. 1527–1554. ISSN : 0899-7667 (cf. p. [15](#)).

- [37] HINTON, Li DENG, Dong YU et al. « Deep Neural Networks for Acoustic Modeling in Speech Recognition ». In : *Signal Processing Magazine* (2012) (cf. p. 16).
- [38] Sepp HOCHREITER. « The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions ». In : *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.* 6.2 (1998), p. 107–116 (cf. p. 39).
- [39] Sepp HOCHREITER et Jurgen SCHMIDHUBER. « Long Short-Term Memory ». In : *Neural Comput.* 9.8 (1997), p. 1735–1780 (cf. p. 39).
- [40] Eric H. HUANG, Richard SOCHER, Christopher D. MANNING et al. « Improving Word Representations via Global Context and Multiple Word Prototypes ». In : *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics : Long Papers - Volume 1*. ACL '12. 2012, p. 873–882 (cf. p. 64).
- [41] Zhiheng HUANG, Wei XU et Kai YU. « Bidirectional LSTM-CRF Models for Sequence Tagging. » In : *CoRR* abs/1508.01991 (2015) (cf. p. 51, 53–55, 91–93).
- [42] V. JOUSSE, S. MEIGNIER, C. JACQUIN et al. « Analyse conjointe du signal sonore et de sa transcription pour l'identification nommée de locuteur ». In : *Traitement automatique des langues* 50.1 (2009). URL : <http://www.atala.org/IMG/pdf/TAL-2009-50-1-08-Jousse.pdf> (cf. p. 113).
- [43] Diederik P. KINGMA et Jimmy BA. « Adam : A Method for Stochastic Optimization ». In : *CoRR* abs/1412.6980 (2014) (cf. p. 35).
- [44] Ryan KIROS, Yukun ZHU, Ruslan R SALAKHUTDINOV et al. « Skip-Thought Vectors ». In : *Advances in Neural Information Processing Systems* 28. 2015, p. 3294–3302 (cf. p. 65).
- [45] Alex KRIZHEVSKY, I. SUTSKEVER et HINTON. « ImageNet Classification with Deep Convolutional Neural Networks ». In : *Advances in Neural Information Processing Systems* 25. Sous la dir. de F. PEREIRA, C. J. C. BURGESS, L. BOTTOU et al. 2012, p. 1097–1105 (cf. p. 37).
- [46] Taku KUDOH et Yuji MATSUMOTO. « Use of Support Vector Learning for Chunk Identification ». In : *Proceedings of the 2Nd Workshop on Learning Language in Logic and the 4th Conference on Computational Natural Language Learning - Volume 7*. ConLL '00. Lisbon, Portugal : Association for Computational Linguistics, 2000, p. 142–144 (cf. p. 55).
- [47] Matthieu LABEAU et Alexandre ALLAUZEN. « Character and Subword-Based Word Representation for Neural Language Modeling Prediction ». In : *Proceedings of the First Workshop on Subword and Character Level Models in NLP, Copenhagen, Denmark, September 7, 2017*. 2017, p. 1–13. URL : <http://aclanthology.info/papers/W17-4101/w17-4101> (cf. p. 120).

- [48] John D. LAFFERTY, Andrew MCCALLUM et Fernando C. N. PEREIRA. « Conditional Random Fields : Probabilistic Models for Segmenting and Labeling Sequence Data ». In : *Proceedings of the Eighteenth International Conference on Machine Learning*. ICML '01. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 2001, p. 282–289. ISBN : 1-55860-778-1 (cf. p. 57).
- [49] Carole LAILLER, Anaïs LANDEAU, Frédéric BÉCHET et al. « Enhancing the RATP-DECODA corpus with linguistic annotations for performing a large range of NLP tasks ». In : *Proceedings of LREC*. 2016 (cf. p. 107).
- [50] William LEHELLE et Philippe LANGLAIS. « Utilisation de représentations de mots pour l'étiquetage de rôles sémantiques suivant FrameNet ». In : *TALN*. 2014, p. 36–45 (cf. p. 62).
- [51] Y. LECUN, L. BOTTOU, Y. BENGIO et al. « Gradient-Based Learning Applied to Document Recognition ». In : *Intelligent Signal Processing*. IEEE Press, 2001, p. 306–351 (cf. p. 32, 75).
- [52] LECUN, BOTTOU, ORR et al. « Efficient BackProp ». In : *Neural Networks : Tricks of the Trade, This Book is an Outgrowth of a 1996 NIPS Workshop*. London, UK, UK : Springer-Verlag, 1998, p. 9–50 (cf. p. 32).
- [53] Omer LEVY et Yoav GOLDBERG. « Dependency-Based Word Embeddings ». In : *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. 2014, p. 302–308 (cf. p. 65, 90).
- [54] Minh-Thang LUONG, Quoc V. LE, Ilya SUTSKEVER et al. « Multi-task Sequence to Sequence Learning ». In : *CoRR abs/1511.06114* (2015). URL : <http://arxiv.org/abs/1511.06114> (cf. p. 74, 78, 86).
- [55] MANNING. « The Stanford CoreNLP Natural Language Processing Toolkit. » In : *ACL (System Demonstrations)*. 2014, p. 55–60 (cf. p. 51).
- [56] C. MANNING. « Part-of-Speech Tagging from 97% to 100% : Is it time for some linguistics? » In : *Computational Linguistics and Intelligent Text Processing, 12th International Conference (CICLing), Part I. Lecture Notes in Computer Science 6608*. Sous la dir. d'A. GELBUKH. 2011, p. 171–189 (cf. p. 53, 91).
- [57] Ryan T MCDONALD, Joakim NIVRE, Yvonne QUIRMBACH-BRUNDAGE et al. « Universal Dependency Annotation for Multilingual Parsing. » In : *ACL (2)*. 2013, p. 92–97 (cf. p. 55, 95).
- [58] T. MIKOLOV, I. SUTSKEVER, K. CHEN et al. « Distributed Representations of Words and Phrases and their Compositionality ». In : *Deep Learning Workshop at the 2013 Conference on Neural Information Processing Systems (NIPS)*. 2013 (cf. p. 51, 63).
- [59] A. NASR, F. BÉCHET, J.F. REY et al. « MACAON : An NLP tool suite for processing word lattices ». In : *Proceedings of the ACL 2011 System Demonstration* (2011), p. 86–91 (cf. p. 104, 111, 122).

- [60] Alexis NASR, Frederic BECHET, Benoit FAVRE et al. « Automatically enriching spoken corpora with syntactic information for linguistic studies ». In : *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*. 2014, p. 854–858 (cf. p. 17, 111).
- [61] NASR et BÉCHET. « MACAON : An NLP Tool Suite for Processing Word Lattices ». In : *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics : Human Language Technologies : Systems Demonstrations*. 2011, p. 86–91 (cf. p. 51, 102).
- [62] Joakim NIVRE. « Algorithms for Deterministic Incremental Dependency Parsing ». In : *Computational Linguistics* 34.4 (2008), p. 513–553. URL : <http://www.aclweb.org/anthology/J/J08/J08-4003.pdf> (cf. p. 55, 96).
- [63] Joakim NIVRE et Ryan MCDONALD. « Integrating GraphBased and Transition-Based Dependency Parsers ». In : *In Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics : Human Language Technologies (ACL-08 : HLT)*. 2008, p. 950–958 (cf. p. 55).
- [64] Naoaki OKAZAKI. *CRFsuite : a fast implementation of Conditional Random Fields (CRFs)*. 2007. URL : <http://www.chokkan.org/software/crfsuite/> (cf. p. 122).
- [65] Daniel POVEY, Arnab GHOSHAL, Gilles BOULIANNE et al. « The Kaldi speech recognition toolkit ». In : *IEEE 2011 workshop on automatic speech recognition and understanding*. 2011 (cf. p. 107).
- [66] Sameer PRADHAN, Wayne WARD, Kadri HACIOGLU et al. « Semantic role labeling using different syntactic views ». In : *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics. 2005, p. 581–588 (cf. p. 102).
- [67] Lawrence R. RABINER. « A tutorial on hidden Markov models and selected applications in speech recognition ». In : *PROCEEDINGS OF THE IEEE*. 1989, p. 257–286 (cf. p. 57).
- [68] Han REN, Donghong JI, Jing WAN et al. « Parsing Syntactic and Semantic Dependencies for Multiple Languages with a Pipeline Approach ». In : *Proceedings of the Thirteenth Conference on Computational Natural Language Learning : Shared Task*. CoNLL '09. 2009, p. 97–102 (cf. p. 56, 97).
- [69] David E. RUMELHART, Geoffrey E. HINTON et Ronald J. WILLIAMS. « Neurocomputing : Foundations of Research ». In : sous la dir. de James A. ANDERSON et Edward ROSENFELD. Cambridge, MA, USA : MIT Press, 1988. Chap. Learning Representations by Back-propagating Errors, p. 696–699 (cf. p. 35).

- [70] Cicero D. SANTOS et Bianca ZADROZNY. « Learning Character-level Representations for Part-of-Speech Tagging ». In : *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*. 2014, p. 1818–1826 (cf. p. 57, 120).
- [71] SCHMIDHUBER. *Learning Algorithms for Networks with Internal and External Feedback*. 1990 (cf. p. 37).
- [72] M. SCHUSTER et K.K. PALIWAL. « Bidirectional Recurrent Neural Networks ». In : *Trans. Sig. Proc.* 45.11 (1997), p. 2673–2681 (cf. p. 40).
- [73] Iulian Vlad SERBAN, Alessandro SORDONI, Yoshua BENGIO et al. « Hierarchical Neural Network Generative Models for Movie Dialogues ». In : *CoRR abs/1507.04808* (2015) (cf. p. 52).
- [74] Fei SHA et Fernando PEREIRA. « Shallow Parsing with Conditional Random Fields ». In : *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1. NAACL '03*. Edmonton, Canada : Association for Computational Linguistics, 2003, p. 134–141 (cf. p. 54, 55, 92, 93).
- [75] Hong SHEN et Anoop SARKAR. « Voting Between Multiple Data Representations for Text Chunking ». In : *Advances in Artificial Intelligence : 18th Conference of the Canadian Society for Computational Studies of Intelligence, Canadian AI 2005, Victoria, Canada, May 9-11, 2005. Proceedings*. Sous la dir. de Balázs KÉGL et Guy LAPALME. Berlin, Heidelberg : Springer Berlin Heidelberg, 2005, p. 389–400 (cf. p. 54, 55, 92, 93).
- [76] Richard SOCHER, Alex PERELYGIN, Jean WU et al. « Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank ». In : *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2013, p. 1631–1642 (cf. p. 51).
- [77] Xu SUN. « Structure Regularization for Structured Prediction : Theories and Experiments ». In : *CoRR abs/1411.6243* (2014) (cf. p. 53, 91).
- [78] Martin SUNDERMEYER, Hermann NEY et Ralf SCHLUTER. « From Feed-forward to Recurrent LSTM Neural Networks for Language Modeling ». In : *IEEE/ACM Transactions on Audio, Speech & Language Processing* 23.3 (2015), p. 517–529 (cf. p. 39, 52).
- [79] Swabha SWAYAMDIPTA, Miguel BALLESTEROS, Chris DYER et al. « Greedy, Joint Syntactic-Semantic Parsing with Stack LSTMs ». In : *CoRR abs/1606.08954* (2016) (cf. p. 56, 97).
- [80] Christian SZEGEDY, Wei LIU, Yangqing JIA et al. « Going Deeper with Convolutions ». In : *Computer Vision and Pattern Recognition (CVPR)*. 2015. URL : <http://arxiv.org/abs/1409.4842> (cf. p. 16).

- [81] K. TOUTANOVA, C. MANNING, D. KLEIN et al. « Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network ». In : *HLT-NAACL*. 2003 (cf. p. 53).
- [82] Jeremy TRIONE, Frederic BECHET, Benoit FAVRE et al. « Rapid FrameNet annotation of spoken conversation transcripts ». In : *Proceedings of the 11th Joint ACL-ISO Workshop on Interoperable Semantic Annotation*. 2015 (cf. p. 104–106, 111).
- [83] Gokhan TUR et Renato DE MORI. *Spoken language understanding : Systems for extracting semantic information from speech*. John Wiley & Sons, 2011 (cf. p. 101).
- [84] Joseph TURIAN, Lev RATINOV et Yoshua BENGIO. « Word representations : a simple and general method for semi-supervised learning ». In : *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics. 2010, p. 384–394 (cf. p. 62).
- [85] Oriol VINYALS, Meire FORTUNATO et Navdeep JAITLY. « Pointer Networks ». In : *Advances in Neural Information Processing Systems 28*. 2015, p. 2692–2700 (cf. p. 98).
- [86] Oriol VINYALS, Łukasz KAISER, Terry KOO et al. « Grammar as a Foreign Language ». In : *Advances in Neural Information Processing Systems 28*. 2015, p. 2773–2781 (cf. p. 56).
- [87] WESTON, CHOPRA et BORDES. « Memory Networks ». In : *CoRR abs/1410.3916* (2014) (cf. p. 52).

# ANNEXES

## Publications

L'ensemble des recherches présentées dans cet ouvrage ont été réalisées entre 2014 et 2016. Elles ont donné lieu à un certain nombre de publications et présentations d'article à l'occasion de conférences.

- TALN 2014 - "*Réseaux de neurones profonds pour l'étiquetage morpho-syntaxique*" est un article présentant les résultats de nos études préliminaires. On y considère un réseau de neurones, plus précisément un *Perceptron multi-couches*, entraîné au *POS-tagging* sur le *PennTree Bank*. On étudie l'utilisation de *word embeddings* afin de représenter les mots et on s'intéresse à la notion de raffinement des représentations au cours de l'apprentissage.

- INTERSPEECH 2015 - "*Lexical embedding adaptation for open-domain spoken language understanding*" est un article abordant le problème des mots hors-vocabulaire. En particulier, on s'intéresse aux mots absents du corpus d'entraînement (OOT). La méthode de raffinement des *embeddings* est uniquement applicable à des mots effectivement présents dans le corpus d'entraînement. Ainsi, les représentations associées à tout mot n'apparaissant dans aucun exemple d'apprentissage n'est jamais remise en cause par l'algorithme. Ce phénomène conduit à un espace de représentation où certains *embeddings* ont été modifiés, raffinés, tandis que d'autres sont restés inchangés. On étudie l'impact d'une stratégie d'adaptation visant à raffiner artificiellement les représentations des OOT.

- NIPS 2015 - "*Adapting lexical representation and OOV handling from written to spoken language with word embedding*" est un second article abordant le problème des mots hors-vocabulaire. On s'intéresse cette fois aux mots absents du corpus sur lequel ont été appris les *embeddings* (OOE). Si l'espace de représentation utilisé pour vectoriser les mots est établi sur un corpus donné, alors tout mot en étant absent n'aura initialement pas de représentation. On se propose d'approximer la représentation qu'aurait eu ce mot s'il avait été présent lors de l'apprentissage des *embeddings*. On étudie l'impact d'une stratégie d'adaptation basée sur l'hypothèse distributionnelle afin de trouver des *embeddings* de substitution pour les OOE.

- INTERSPEECH 2016 - "*Joint Syntactic and Semantic Analysis with a Multi-task Deep Learning Framework for Spoken Language Understanding*" est un article reprenant en détail nos travaux sur l'apprentissage de réseaux multitâches appliqués à l'analyse syntaxique et sémantique, et plus particulièrement à la détection de cadre sémantique sur le corpus DECODA de transcriptions automatiques de conversations téléphoniques. Une version longue intitulée "*Combining linguistic analysis with end-to-end methods for open domain Spoken Language Understanding*" est en cours de soumission auprès du journal *Computer Speech & Language*.

