# Étude sur les représentations continues de mots appliquées à la détection automatique des erreurs de reconnaissance de la parole

Sahar Ghannay

UNIVERSITE BRETAGNE LOIRE

Le Mans Université

# Thèse de Doctorat

Sahar GHANNAY

*Mémoire présenté en vue de l'obtention du*
***grade de Docteur de Le Mans Université***
*sous le sceau de l'Université Bretagne Loire*

**École doctorale : MathSTIC**

**Discipline :** Informatique
**Spécialité :** Informatique
**Unité de recherche :** ***LIUM***

**Soutenue le 20 Septembre 2017**
**Thèse N° : 2017LEMANS1019**

# Étude sur les représentations continues de mots appliquées à la détection des erreurs de reconnaissances de la parole

## JURY

| | |
|---|---|
| Rapporteurs : | **M. Frédéric BECHET**, Professeur, LIF, CNRS, Université Aix-Marseille<br>**Mme Sophie ROSSET**, Directrice de recherche, LIMSI, CNRS, Université Paris-Sud |
| Examinateurs : | **Mme Martine ADDA-DECKER,** Directrice de recherche, Université Paris 3 Sorbonne<br>**M. Benoit FAVRE**, Maître de conférence, LIF, CNRS, Université Aix-Marseille<br>**M. Benjamin LECOUTEUX**, Maître de conférence, LIG, CNRS, Université Grenoble Alpes |
| Directeur de Thèse : | **M. Yannick ESTÈVE**, Professeur, LIUM, Le Mans Université |
| Co-directeur de Thèse : | **Mme Nathalie CAMELIN**, Maître de conférence, LIUM, Le Mans Université |

UNIVERSITE BRETAGNE LOIRE

Le Mans Université

# Thèse de Doctorat

Sahar GHANNAY

# Étude sur les représentations continues de mots appliquées à la détection automatique des erreurs de reconnaissance de la parole

## A study of continuous word representations applied to the automatic detection of speech recognition errors


### Résumé

Nous abordons, dans cette thèse, une étude sur les représentations continues de mots (en anglais *word embeddings)* appliquées à la détection automatique des erreurs dans les transcriptions de la parole.
En dépit de la performance des systèmes de reconnaissance automatique de la parole actuels, de nombreuses erreurs sont encore générées. Cela s'explique par leur sensibilité aux diverses variabilités liées à l'environnement acoustique, au locuteur, au style de langage, à la thématique du discours, *etc.*
Notre étude se concentre sur l'utilisation d'une approche neuronale pour améliorer la détection automatique des erreurs dans les transcriptions automatiques, en exploitant les *word embeddings*. Ces représentations ont révélées être d'un grand atout dans différentes tâches de traitement automatique des langues naturelles (TALN).
L'exploitation des représentations continues de mots repose sur l'idée que la détection d'erreurs consiste à trouver les possibles incongruités linguistiques ou acoustiques au sein des transcriptions automatiques.
L'intérêt est donc de trouver la représentation appropriée du mot qui permet de capturer des informations pertinentes pour pouvoir détecter ces anomalies.
Notre contribution dans le cadre de cette thèse porte sur plusieurs axes.
D'abord, nous commençons par une étude préliminaire dans laquelle nous proposons une architecture neuronale capable d'intégrer différents types de descripteurs, y compris les *word embeddings*.
Ensuite, nous nous focalisons sur une étude approfondie des représentations continues de mots. Cette étude porte d'une part sur l'évaluation de différents types d'*embeddings* linguistiques puis sur leurs combinaisons, afin de tirer profit de leurs complémentarités. D'autre part, elle s'intéresse aux *embeddings* acoustiques de mots. Nous proposons une approche qui repose sur l'utilisation d'un réseau de neurones convolutif pour construire des *embeddings* acoustiques de signal, et un réseau de neurones profond pour construire des *embeddings* acoustiques de mots. De plus, nous présentons deux approches pour évaluer la performance des *embeddings* acoustiques de mots. Nous proposons également d'enrichir la représentation du mot en entrée d'un système de détection d'erreurs par des descripteurs prosodiques en plus des *embeddings* linguistiques et acoustiques.
L'intégration de ces informations dans notre architecture neuronale apporte un gain significatif en termes de réduction du taux d'erreur de classification, en comparaison à un système état de l'art fondé sur les champs aléatoires conditionnels (CRF). Puis, nous présentons une étude portant sur l'analyse des erreurs de classification, qui a pour objectif de percevoir les erreurs difficiles à détecter. Des perspectives pour améliorer la performance de notre système sont également proposées, en modélisant les erreurs au niveau de la phrase.
Finalement, nous exploitons les *embeddings* linguistiques et acoustiques ainsi que l'information fournie par notre système de détection d'erreurs dans plusieurs cadres applicatifs.



### Mots clés

Représentations continues de mots, Détection d'erreurs de reconnaissance de la parole, apprentissage profond, Post-édition des transcriptions automatiques



### Abstract

My thesis concerns a study of continuous word representations applied to the automatic detection of speech recognition errors.
Recent advances in the field of speech processing have led to significant improvements in speech recognition performances. However, recognition errors are still unavoidable.
This reflects their sensitivity to the variability, *e.g.* to acoustic conditions, speaker, language style, etc.
Our study focuses on the use of a neural approach to improve ASR error detection, using *word embeddings*. These representations have proven to be a great asset in various natural language processing tasks (NLP).

The exploitation of continuous word representations is motivated by the fact that ASR error detection consists on locating the possible linguistic or acoustic incongruities in automatic transcriptions.
The aim is therefore to find the appropriate word representation, which makes it possible to capture pertinent information in order to be able to detect these anomalies.
Our contribution in this thesis concerns several initiatives.
First, we start with a preliminary study in which we propose a neural architecture able to integrate different types of features, including *word embeddings*.
Second, we propose a deep study of continuous word representations. This study focuses on the evaluation of different types of linguistic *word embeddings* and their combination in order to take advantage of their complementarities. On the other hand, it focuses on acoustic *embeddings*.
The proposed approach relies on the use of a convolution neural network to build acoustic signal *embeddings*, and a deep neural network to build acoustic word *embeddings*. In addition, we propose two approaches to evaluate the performance of acoustic word *embeddings*.
We also propose to enrich the word representation, in input of the ASR error detection system, by prosodic features in addition to linguistic and acoustic *embeddings*.
Integrating this information into our neural architecture provides a significant improvement in terms of classification error rate reduction in comparison to a conditional random field (CRF) based state-of-the-art approach.
Then, we present a study on the analysis of classification errors, with the aim of perceiving the errors that are difficult to detect. Perspectives for improving the performance of our system are also proposed, by modelling the errors at the sentence level.
Finally, we exploit the linguistic and acoustic *embeddings* as well as the information provided by our ASR error detection system in several downstream applications.



### Key Words

Continuous word representations, Automatic detection of speech recognition errors, Deep learning, Post edition of automatic transcriptions

# *Remerciement*

L'aboutissement de ce travail de thèse n'est que le fruit d'échanges, de conseils et de soutiens d'un grand nombre de personnes auxquelles je tiens à adresser ma profonde reconnaissance.

Je tiens tout d'abord à exprimer ma gratitude et mes vifs remerciements à mes encadrants de thèse M. Yannick Estève, Professeur à Le Mans Université et Mme Nathalie Camelin, Maître de conférence à Le Mans Université, pour leur disponibilité, leur soutien, leurs conseils avisés, leurs encouragements ainsi que pour leur patience. Ce que j'ai appris en travaillant avec eux ne se limite pas à l'aspect scientifique mais s'étend aux aspects humain et relationnel. Je les remercie infiniment.

Je tiens aussi à exprimer mes remerciements à Mme Sophie Rosset, Chargé de recherche au LIMSI (CNRS) à l'Université Paris-Sud ainsi que M. Frédéric Béchet, Professeur à l'Université d'Aix-Marseille, qui ont accepté de juger ce travail et d'en être les rapporteurs. Je remercie également Mme Martine Adda-Decker, Directeur de recherche à l'Université de Paris 3 Sorbonne, M. Benoit Favre, Maître de conférence à l'Université d'Aix-Marseille et M. Benjamin Lecouteux, Maître de conférence à l'Université de Grenoble Alpes, pour avoir accepté de juger cette thèse et pour l'intérêt qu'ils ont porté à mon travail.

Mes remerciements sincères vont également à toutes les personnes que j'ai pu côtoyer quotidiennement au sein du LIUM, leur bonne humeur quotidienne sans faille et leur capacité de travail en équipe exemplaire. Je pense particulièrement à Loïc Barrault, Daniel Luzzati, Paul Deléglise, Fethi Bougares, Haithem Afli, Mercedes Garcia, Carole Lallier, Walid Aransa, Adrien Bardet, Kevin Vythelingum, Dominique Py, Teva Merlin et Étienne Micoulaut. Je terminerai cette partie en remerciant tous mes amis et amies, ceux et celles que j'ai eu la chance de côtoyer et qui m'ont toujours encouragé et supporté moralement.

Rien n'aurait été possible sans le soutien de mon jumeau Mohamed Ali et ma famille. Merci encore à vous, rien n'a su me motiver davantage que votre appui et la confiance que vous m'avez toujours accordée.

Mes pensées vont aussi et particulièrement à mon fiancé Akram qui était à mes côtés dans les moments de joie et de difficultés et qui m'a entouré d'affection. Qu'il trouve ici l'expression de mes remerciements pour sa patience et son soutien inestimable.

Je tiens à remercier très vivement mes amis Rihab, Mounira, Olfa, Yosra, Tarek, Anas et Mohamed pour leurs soutiens, leurs gentillesses et je leur souhaite une très

bonne continuation dans leurs vies professionnelles.

Maman et Papa, j'ai préféré finir les remerciements par vous. Je ne pourrai jamais assez-vous remercier. Vous avez su me protéger et me donner une éducation dont tout enfant rêverait de recevoir. À chaque fois que j'atteins une limite, votre présence et votre amour me poussent à aller encore plus loin. Merci infiniment, je vous aime sans limites.

Enfin, à tous ceux que je n'ai pas pu citer, auxquels je réitère mes sincères remerciements.

*À vous tous, Merci !*

# *Dédicace*

*À mes chers parents et beaux-parents,*
*À mon frère*
*À mon chéri,*
*pour la patience et le dévouement dont ils ont fait preuve.*

# *Résumé*

Nous abordons, dans cette thèse, une étude sur les représentations continues de mots (en anglais *word embeddings*) appliquées à la détection automatique des erreurs dans les transcriptions de la parole. En dépit de la performance des systèmes de reconnaissance automatique de la parole actuels, de nombreuses erreurs sont encore générées. Cela s'explique par leur sensibilité aux diverses variabilités liées à l'environnement acoustique, au locuteur, au style de langage, à la thématique du discours, *etc.* Notre étude se concentre sur l'utilisation d'une approche neuronale pour améliorer la détection automatique des erreurs dans les transcriptions automatiques, en exploitant les *word embeddings*. Ces représentations ont révélées être d'un grand atout dans différentes tâches de traitement automatique des langues naturelles (TALN).

L'exploitation des représentations continues de mots repose sur l'idée que la détection d'erreurs consiste à trouver les possibles incongruités linguistiques ou acoustiques au sein des transcriptions automatiques. L'intérêt est donc de trouver la représentation appropriée du mot qui permet de capturer des informations pertinentes pour pouvoir détecter ces anomalies.

Notre contribution dans le cadre de cette thèse porte sur plusieurs axes. D'abord, nous commençons par une étude préliminaire dans laquelle nous proposons une architecture neuronale capable d'intégrer différents types de descripteurs, y compris les *word embeddings*.

Ensuite, nous nous focalisons sur une étude approfondie des représentations continues de mots. Cette étude porte d'une part sur l'évaluation de différents types d'*embeddings* linguistiques puis sur leurs combinaisons, afin de tirer profit de leurs complémentarités. D'autre part, elle s'intéresse aux *embeddings* acoustiques de mots. Nous proposons une approche qui repose sur l'utilisation d'un réseau de neurones convolutif pour construire des *embeddings* acoustiques de signal, et un réseau de neurones profond pour construire des *embeddings* acoustiques de mots. De plus, nous présentons deux approches pour évaluer la performance des *embeddings* acoustiques de mots. Nous proposons également d'enrichir la représentation du mot en entrée d'un système de détection d'erreurs par des descripteurs prosodiques en plus des *embeddings* linguistiques et acoustiques. L'intégration de ces informations dans notre architecture neuronale apporte un gain significatif en termes de réduction du taux d'erreur de classification, en comparaison à un système état de l'art fondé sur les champs aléatoires conditionnels (CRF).

Puis, nous présentons une étude portant sur l'analyse des erreurs de classification, qui a pour objectif de percevoir les erreurs difficiles à détecter. Des perspectives pour améliorer la performance de notre système sont également proposées, en modélisant les erreurs au niveau de la phrase. Finalement, nous exploitons les *embeddings* linguistiques et acoustiques ainsi que l'information fournie par notre système de détection d'erreurs dans plusieurs cadres applicatifs.

# *Abstract*

My thesis concerns a study of continuous word representations applied to the automatic detection of speech recognition errors. Recent advances in the field of speech processing have led to significant improvements in speech recognition performances. However, recognition errors are still unavoidable. This reflects their sensitivity to the variability, *e.g.* to acoustic conditions, speaker, language style, *etc.* Our study focuses on the use of a neural approach to improve ASR error detection, using word embeddings. These representations have proven to be a great asset in various natural language processing tasks (NLP).

The exploitation of continuous word representations is motivated by the fact that ASR error detection consists on locating the possible linguistic or acoustic incongruities in automatic transcriptions. The aim is therefore to find the appropriate word representation which makes it possible to capture pertinent information in order to be able to detect these anomalies. Our contribution in this thesis concerns several initiatives. First, we start with a preliminary study in which we propose a neural architecture able to integrate different types of features, including word embeddings.

Second, we propose a deep study of continuous word representations. This study focuses on the evaluation of different types of linguistic word embeddings and their combination in order to take advantage of their complementarities. On the other hand, it focuses on acoustic embeddings. The proposed approach relies on the use of a convolution neural network to build acoustic signal embeddings, and a deep neural network to build acoustic word embeddings. In addition, we propose two approaches to evaluate the performance of acoustic word embeddings. We also propose to enrich the word representation, in input of the ASR error detection system, by prosodic features in addition to linguistic and acoustic embeddings. Integrating this information into our neural architecture provides a significant improvement in terms of classification error rate reduction in comparison to a conditional random field (CRF) based state-of-the-art approach.

Then, we present a study on the analysis of classification errors, with the aim of perceiving the errors that are difficult to detect. Perspectives for improving the performance of our system are also proposed, by modelling the errors at the sentence level. Finally, we exploit the linguistic and acoustic embeddings as well as the information provided by our ASR error detection system in several downstream applications.

# TABLE DES MATIÈRES

# LISTE DES TABLEAUX

# TABLE DES FIGURES

# ACRONYMES

| | |
|---|---|
| **ACP** | Analyse en Composante Principale |
| **ANR** | Agence Nationale de la Recherche |
| **Bi-RNN** | Réseaux de Neurones Récurrents bidirectionnel |
| **CBOW** | Continuous bag of word |
| **CER** | Taux d'Erreur de Concept |
| **CMLLR** | Constrained Maximum Likelihood Linear Regression |
| **CN** | Confusion Network |
| **CNN** | Réseaux de Neurones Convolutifs (Convolutional Neural Network) |
| **CRF** | Champs Conditionnels Aléatoires (Conditional Random Fields-CRF) |
| **CSLM** | Continuous Space Language Models |
| **CVER** | Taux d'Erreur de Concept/Valeur |
| **DNN** | Deep neural network |
| **EUMSSI** | Event Understanding through Multimodal Social Stream Interpretation |
| **FSM** | Machine à états-finis (Finite-State Machine) |
| **fMLLR** | feature-space Maximum Likelihood Linear Regression |
| **GloVe** | VEcteur GLobal |
| **GRU** | Gated Recurrent Unit |
| **IA** | Intelligence Artificielle |
| **LDA** | Allocation Latente de Dirichlet |
| **LPCC** | Linear Prediction Ceptral Coefficients |
| **LSA** | Analyse Sémantique latente |
| **LSTM** | Long Short-Term Memory |
| **MAP** | Maximum A Posteriori |
| **MFCC** | Mel-scale Frequency Cepstral Coefficients |
| **MLP** | Perceptron Multicouches ( Multilayer Perceptron) |
| **MLP-MS** | Multicouches Multi Stream |
| **MLLR** | Maximum Likelihood Linear Regression |
| **MMC** | Modèles de Markov Cachés |
| **NCE** | Entropie Croisée Normalisée (Normalized Cross Entropy) |

| | |
|---|---|
| **PAP** | Probabilité *a Posteriori* |
| **PLP** | Perceptual Linear Prediction |
| **RAP** | Reconnaissance Automatique de la Parole |
| **RNN** | Réseaux de Neurones Récurrents (Recurrent Neural Network) |
| **RNN-EDA** | RNN de type encodeur-décodeur avec mécanisme d'attention |
| **SDAE** | Stacked Denoising Auto-Encoder |
| **SLU** | Spoken Langage Understanding |
| **SVD** | Singular Value Decomposition |
| **TALN** | Traitement Automatique des Langages Naturelles |
| **t-SNE** | t-Distributed Stochastic Neighbor Embedding |
| **VERA** | AdVanced ERror Analysis for speech recognition |
| **WER** | Taux d'Erreur-Mot (Word Error Rate) |

# INTRODUCTION

## Introduction

Les avancées scientifiques récentes dans le domaine de l'apprentissage profond ainsi que la disponibilité de grandes quantités de données et de dispositifs de calcul puissants ont provoqué ces dernières années un grand bouleversement dans le domaine de l'intelligence artificielle et ses applications.

Dans le domaine du traitement automatique des langues, les méthodes d'apprentissage profond doivent entre autres leur succès à leur faculté de construire des représentations continues de mots ou *word embeddings* pertinentes. Ces représentations continues, qui constituent la base des modèles de langage neuronaux, sont des vecteurs denses à valeurs réelles construits pour chacun des mots d'un vocabulaire. Elles constituent une projection de ces mots dans un espace continu de faible dimension. Les *word embeddings* sont au coeur de l'intérêt de nombreux travaux de recherche ces dernières années et plusieurs approches ont été proposées pour construire ces représentations. Elles s'appuient généralement sur la prise en compte des contextes d'apparition d'un mot, représentés sous différentes formes, comme les sacs de mots continus, les dépendances syntaxiques, les co-occurrences de mots, ..., et voire même sur le signal audio. Ainsi, ces *word embeddings* peuvent capturer différents types d'information entre les mots : similarités sémantiques, syntaxiques, acoustiques, *etc.*

Ces représentations se sont révélées être un atout dans de nombreuses tâches de traitement automatique des langues naturelles (TALN) telles que l'étiquetage morpho-syntaxique, le regroupement en syntagmes, la reconnaissance d'entités nommées, l'étiquetage de rôles sémantiques, *etc.*

Nous présentons dans cette thèse une étude portant sur les représentations continues de mots, appliquées à une tâche importante en traitement du langage parlé qu'est la détection automatique d'erreurs de reconnaissance de la parole.

Les systèmes de reconnaissance automatique de la parole (SRAP) peuvent actuellement fournir des transcriptions avec un niveau de performance permettant leur intégration dans de nombreuses applications. Néanmoins, les systèmes état de l'art

sont loin d'être parfaits, et produisent souvent des transcriptions qui contiennent des erreurs de reconnaissance. Ces erreurs sont liées principalement à la sensibilité des SRAPs aux diverses variabilités liées à l'environnement acoustique (type de microphone, bruits, ...), au locuteur (anatomie, état émotionnel ou physique, genre du locuteur, accent, ...), à l'élocution (discours spontané, hésitations, silences, ...), à la thématique du discours, *etc.*

Les erreurs rencontrées présentent un obstacle à l'exploitation des transcriptions automatiques, surtout pour les applications qui ont une tolérance très faible à l'erreur et nécessitent une transcription parfaite comme les applications de dictée vocale ou de sous-titrage. D'autres applications s'appuyant sur l'exploitation des transcriptions automatiques peuvent être développées, mais leurs performances sont plus ou moins influencées par les erreurs de reconnaissance : l'extraction d'information, la traduction de la parole, la compréhension de la parole, la recherche documentaire, *etc.*

Les erreurs de reconnaissance constituent une mauvaise interprétation du signal de parole. Elles peuvent provenir de la qualité du canal d'enregistrement, des bruits liés à l'environnement sonore tels que les claquements de portes et autres bruits de rue, *etc*, d'une mauvaise prononciation d'un mot, de locuteurs qui s'expriment en même temps, ... Elles peuvent également provenir d'une mauvaise adéquation du SRAP en terme de vocabulaire retenu, de modèles acoustiques inadaptés, de modèles de langage non pertinents, ...

Une génération sans erreur des transcriptions de parole, quelles que soient les conditions, reste un but ultime. À défaut d'être atteint, ce but peut être approché si l'on est capable de réduire et/ou de corriger la plupart des erreurs dans les transcriptions automatiques. Cependant, la détection d'erreurs n'est pas une tâche facile compte tenu du fait qu'il existe plusieurs types d'erreurs tant au niveau lexical que syntaxique. Ces erreurs peuvent aller d'une simple erreur d'accord en genre à l'insertion d'un mot non pertinent pour la compréhension globale de la séquence de mots. Elles peuvent aussi se répercuter sur les mots voisins et créer une zone de mots erronés.

La détection d'erreurs a pour but d'atténuer l'impact de ces erreurs sur toute application exploitant les transcriptions automatiques. Cette tâche consiste à détecter les incongruités linguistiques et/ou acoustiques dans les transcriptions automatiques. C'est pourquoi il faut utiliser une représentation appropriée du mot permettant de capturer des informations pertinentes pour pouvoir détecter ces anomalies.

De nombreuses études se focalisent sur la tâche de détection d'erreurs. La plupart de ces études récentes sont fondées sur les champs aléatoires conditionnels (*Conditional Random Fields - CRF*), qui constituaient au début de ces travaux de thèse la méthode de l'état de l'art pour cette tâche. Ces approches s'appuient sur l'intégration de plusieurs sources d'informations : syntaxiques, lexicales, contextuelles, *etc.* Nous proposons dans ce mémoire d'étudier l'utilisation d'une approche neuronale pour cette tâche. Cela se justifie par les performances prometteuses récentes de ce type d'approche dans le cadre de la modélisation acoustique et du TALN.

Cette thèse a été co-financée par la région Pays de la Loire et par la Commission

Européenne à travers le projet EUMSSI (*Event Understanding through Multimodal Social Stream Interpretation*). Ce projet (2013-2016) avait pour but de faciliter l'extraction d'informations multimedia (audio, video, texte, image) à partir de très grandes bases de données hétérogènes pour aider des (data-)journalistes à rédiger leurs articles. Le LIUM, impliqué dans ce projet, a développé des systèmes très rapides de transcription automatique en anglais, en français et en allemand. Dans ce contexte, la détection d'erreur est une tâche importante pour améliorer l'efficacité des outils de recherche documentaire appliqués sur de grands volumes de transcriptions automatiques.

Pendant ma thèse, le LIUM a été partenaire et coordinateur du projet ANR (Agence National de la Recherche) VERA (*AdVanced ERror Analysis for speech recognition*) qui s'est déroulé de 2014 à 2016. Dans ce projet, la tâche du LIUM était de fournir les transcriptions automatiques aux autres partenaires, de développer de nouvelles mesures de confiance et de les intégrer dans des cadres applicatifs.

C'est dans cet environnement collaboratif que ce travail a été réalisé, principalement en langue française.

## Structure du document

Ce document est organisé en deux grandes parties. Nous présentons dans un premier temps l'état de l'art des techniques d'apprentissage profond, des systèmes de reconnaissance de la parole et de la détection automatique d'erreurs dans les transcriptions de la parole. Nous développons ensuite le travail réalisé durant cette thèse, à savoir l'élaboration d'un système de détection d'erreurs, l'étude de différents types d'*embeddings* linguistiques et acoustiques, l'amélioration de la détection d'erreurs après analyse qualitative des sorties du système proposé, et enfin l'exploitation des embeddings et des sorties des systèmes de détection d'erreurs dans différents cadres applicatifs.

Nous présentons dans le premier chapitre les techniques d'apprentissage profond, en nous focalisant sur les principales architectures neuronales existantes, ainsi que sur les représentations continues de mots.

Dans le deuxième chapitre, nous présentons les composants d'un système de reconnaissance automatique de la parole, et nous décrivons toutes les étapes nécessaires au système pour passer d'un signal de parole à une transcription.

Nous détaillons dans le chapitre 3 les différentes catégories des mesures de confiance, plus particulièrement la combinaison de différents descripteurs. Cela nous amène à définir plusieurs types de descripteurs, à faire un point sur les différents types de classifieurs permettant la combinaison de ces descripteurs, et présenter les approches à l'état de l'art pour la détection automatique d'erreurs. Ces approches s'appuient sur les CRF et plus récemment sur les réseaux de neurones.

La première partie de mes contributions est présentée dans le chapitre 4. Dans cette partie nous proposons une architecture neuronale pour la détection d'erreurs. Nous proposons également une étude sur l'utilisation de plusieurs types de *word*

*embeddings* linguistiques provenant de différentes implémentations disponibles.

L'évaluation des *word embeddings* linguistiques sur la tâche de détection d'erreurs a montré une complémentarité potentielle. C'est pourquoi, nous présentons dans le chapitre 5 une comparaison systématique de leur impact sur plusieurs tâches d'évaluation pour étudier cette complémentarité. Nous proposons également, suite aux résultats d'évaluation, d'étudier et d'évaluer différentes approches pour combiner ces *embeddings* afin de tirer profit de leurs complémentarités pour la tâche de détection d'erreurs.

Nous nous intéressons dans le chapitre 6 à enrichir le système de détection d'erreurs avec des informations acoustiques, notamment des *embeddings* acoustiques et des descripteurs prosodiques. Nous présentons une approche pour construire des *embeddings* acoustiques de signal et de mots et différentes tâches d'évaluation pour évaluer la performance des *embeddings* acoustiques de mots.

Nous présentons, dans le chapitre 7, une étude portant sur l'analyse d'erreurs des sorties de notre système de détection d'erreurs, selon plusieurs critères. Suite aux résultats de ces analyses, nous proposons également une étude sur la modélisation d'erreurs au niveau de la phrase.

Dans le chapitre 8, nous étudions l'exploitation des *word embeddings* et des informations fournies par les systèmes de détection d'erreurs proposés dans différents cadres applicatifs.

Finalement, un résumé des points clés de la thèse est présenté dans le dernier chapitre, ainsi que quelques perspectives pour de futurs travaux de recherche.

# Première partie

# Contexte de travail et état de l'art

CHAPITRE 1

# TECHNIQUES D'APPRENTISSAGE PROFOND

**Sommaire**

L'apprentissage profond (*deep learning* en anglais) a provoqué ces dernières années un grand bouleversement dans le domaine de l'intelligence artificielle (IA), ainsi que ses applications. L'apprentissage profond a permis le franchissement de palliers importants dans de nombreuses applications, comme la reconnaissance et la classification d'images [LeCun *et al.*, 2010, Tompson *et al.*, 2014, Zbontar et LeCun, 2016], la modélisation acoustique, [Hinton *et al.*, 2012, Jaitly *et al.*, 2012, Abdel-Hamid *et al.*, 2012], la modélisation du langage [Mikolov *et al.*, 2011b, Schwenk, 2013], le traitement automatique des langues naturelles (TALN ou NLP pour *Natural Language Processing*) [Turian *et al.*, 2010, Collobert *et al.*, 2011, Bansal *et al.*, 2014a], *etc.*

Ces performances s'expliquent par les avancés récentes dans le domaine de l'apprentissage automatique ainsi que par la disponibilité de grandes quantités de données d'apprentissage et de dispositifs de calcul puissants. Ces avancées se traduisent par des méthodes plus efficaces pour l'apprentissage de réseaux de neurones profonds. Ces derniers sont capables à travers leurs architectures de modéliser des abstractions de haut niveau au sein des données. En effet, l'architecture profonde représente une cascade de modules (représentations) qui sont entrainés simultanément pour résoudre une tâche donnée. Ceci permet au réseau de neurones de construire des abstractions les unes à la suite des autres : des plus simples aux plus complexes [Hinton *et al.*, 2006, Bengio *et al.*, 2007a]. Par exemple, dans le cas du traitement d'images, le premier module vise à extraire des caractéristiques basiques, comme la présence ou l'absence d'un contour, d'orientations particulières, *etc.* Ensuite, le module suivant peut permettre de détecter des conjonctions de ces contours pour former des formes élémentaires comme des cercles, des angles, *etc.* Enfin, les derniers modules manipulent ces formes élémentaires sur un exemple pour détecter des objets [Lee *et al.*, 2011, Zeiler et Fergus, 2014].

Ce chapitre présente, d'une manière succincte, les techniques d'apprentissage profond, en se focalisant sur les architectures neuronales ainsi que sur les représentations continues de mots. Nous présenterons dans un premier temps les réseaux de neurones et exposerons les architectures neuronales utilisées dans cette thèse. Ensuite, nous nous attacherons à définir les représentations continues de mots, qui nous amèneront à faire un point sur l'état de l'art des différentes approches utilisées pour construire ces représentations. Une description des techniques de visualisation des représentations continues de mots clôturera ce chapitre.

## 1.1 Réseaux de neurones

Les réseaux de neurones constituent aujourd'hui une des techniques de classification les plus utilisées et les plus performantes dans différentes tâches, y compris les tâches de TALN [Turian *et al.*, 2010, Collobert *et al.*, 2011, Bansal *et al.*, 2014a].

### 1.1.1 Définition

L'idée de construire des réseaux de neurones artificiels remonte à la fin des années 50. Elle s'inspire du fonctionnement du cortex visuel des animaux. Les premiers réseaux de neurones artificiels apparaissent sous le terme de perceptron, introduit en 1958 par Rosenblatt [Rosenblatt, 1958].

Le perceptron est la forme la plus simple d'une architecture neuronale, puisqu'il est composé d'un seul neurone. Il a été à l'origine développé pour la reconnaissance de formes. À cause de son architecture simple, le perceptron ne peut résoudre que des problèmes linéairement séparables et n'est pas capable de traiter des problèmes de classification complexes, comme l'ont prouvé Marvin Minsky et Papert dans leur étude [Minsky et Papert, 1969]. Ce problème a été résolu à la suite du développement de nouveaux types de réseaux de neurones formels [Hopfield, 1982, Hopfield, 1984] et de nouvelles méthodes d'apprentissage : Le Cun [LeCun *et al.*, 1988, LeCun, 1985], Rumelhart [Rumelhart *et al.*, 1985, Rumelhart *et al.*, 1988], *etc.*

Un réseau de neurones est l'association de plusieurs neurones formels en un graphe plus au moins complexe. Un neurone formel est une approximation mathématique et informatique d'un neurone biologique, comme le montre la figure 1.1. Cette figure illustre le parallèle entre un neurone biologique et un neurone artificiel.

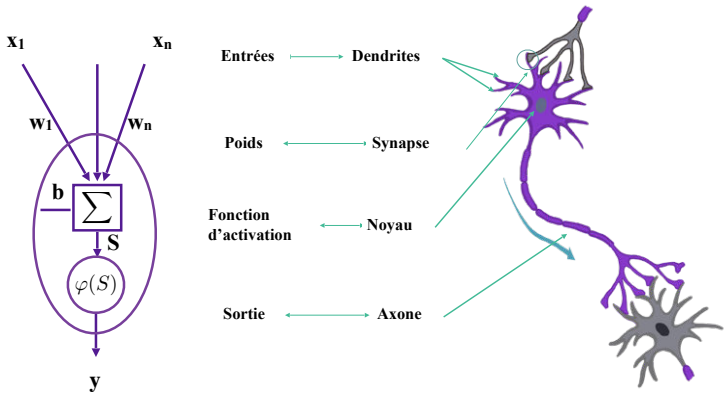



FIGURE 1.1 – Comparaison entre un neurone biologique et un neurone formel.

Un neurone formel possède généralement plusieurs entrées $x_1 \dots x_n$. À chaque entrée est associé un point synaptique $w_i$ où $1 \leq i \leq n$, qui représente l'importance de l'information $x_i$. Le neurone formel possède également une sortie $y$ qui peut être utilisée comme entrée pour d'autres neurones. Elle est définie par l'équation 1.1.

$$y = \varphi(W.X + b) \tag{1.1}$$

La sortie $y$ d'un neurone correspond à l'application d'une fonction d'activation $\varphi$ sur la somme des vecteurs d'entrées $X = (x_1 \dots x_n)$ pondérés par le vecteur de poids $W = (w_1 \dots w_n)$ et un paramètre additionnel appelé biais. Le biais $b$ peut être considéré comme le poids d'une entrée constante égale à 1. Il permet d'ajouter de

la flexibilité au réseau en agissant sur la position de la frontière de décision (pour plus de détails, se référer à [Rosenblatt, 1957]).

La fonction d'activation $\varphi$ est généralement définie par l'une des fonctions suivantes :

— Identité :

$$\varphi(Z) = Z \tag{1.2}$$

— Sigmoïde :

$$\varphi(Z) = \frac{1}{1 + e^{-Z}} \tag{1.3}$$

— Tangente hyperbolique :

$$\varphi(Z) = tanh(Z) = \frac{1 - e^{-Z}}{1 + e^{-Z}} \tag{1.4}$$

— ReLu (Rectified linear unit) :

$$\varphi(Z) = ReLu(Z) = \max(0, Z) \tag{1.5}$$

avec $Z = W.X + b$.

### 1.1.2 Architectures

L'architecture d'un réseau de neurones détermine la façon par laquelle les neurones sont ordonnés et connectés au sein d'un même réseau. Quelque soit l'architecture, un réseau de neurones est composé de plusieurs couches de neurones successives : les entrées, les couches suivantes dites couches cachées (du fait qu'elles n'ont aucun contact direct avec *l'extérieur* du réseau) jusqu'à la couche de sorties. La profondeur d'une telle architecture dépend du nombre de couches cachées.

Nous allons dans cette section nous intéresser aux familles de réseaux de neurones utilisées dans cette thèse, à savoir le perceptron multicouches [Rumelhart *et al.*, 1988], les réseaux de neurones convolutifs [LeCun *et al.*, 1990], ainsi que les auto-encodeurs [Rumelhart *et al.*, 1985]. À noter toutefois qu'il existe d'autres types de réseaux de neurones, par exemple les réseaux de neurones récurrents [Medsker et Jain, 1999], le *Deep Belief Network* [Hinton *et al.*, 2006], *Restricted Boltzmann Machine* [Salakhutdinov et Hinton, 2009], *Recursive Neural Network* [Socher *et al.*, 2011] qui ne seront pas abordés. Nous suggérons au lecteur intéressé de se référer aux ouvrages suivants [Bishop, 2006, Bengio, 2009a, Ian Goodfellow et Courville, 2016].

#### 1.1.2.1 Perceptron multicouches

**Topologie**

Un perceptron multicouches (MLP pour *Multilayer Perceptron*) est un réseau de neurones de type *feedforward*, où l'information qui le traverse ne circule que dans

un sens unique : de l'entrée vers la sortie [Rumelhart *et al.*, 1988].

Un réseau de neurones *feedforward* est constitué de plusieurs couches successives : on intercale entre la couche d'entrée et celle de la sortie une ou plusieurs couches cachées. Chaque neurone d'une couche est relié à tous ceux de la suivante. Les sorties de la couche inférieure sont propagées à travers les différentes couches, de l'entrée vers la sortie.

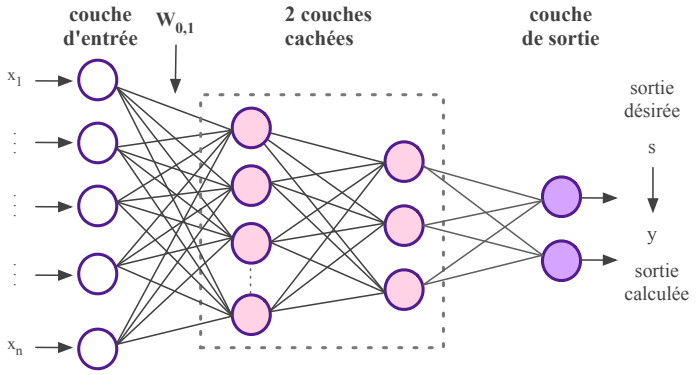



FIGURE 1.2 – Réseau de neurones *feedforward* de deux couches cachées

### Calculs

Supposons que l'on dispose d'un MLP de deux couches cachées comme celui de la figure 1.2 avec $n$ neurones d'entrée, activés par un vecteur d'entrée $X$ et par la matrice de poids $W_{0,1}$. Cette matrice de poids correspond à la connexion entre les neurones de la couche d'entrée 0 et les neurones de la première couche cachée 1. Le vecteur des sorties $h_1$ de la première couche cachée est calculé par la fonction suivante :

$$h_1 = \varphi(W_{0,1}.X + b_1) \tag{1.6}$$

où $\varphi$ est l'une des fonctions d'activation définie précédemment (sous-section 1.1.1). D'une manière générale, le vecteur des sorties $h_l$ d'une couche $l$ est ensuite utilisé comme entrée pour la couche suivante $l+1$. Ainsi, nous pouvons généraliser l'équation 1.6 à toutes les couches comme suit :

$$h_{l+1} = \varphi(W_{l,l+1}.h_l + b_{l+1}) \tag{1.7}$$

Comme pour l'équation 1.7, nous calculons la sortie $y$ en fonction de vecteur $h_{L-1}$ par l'équation 1.8.

$$y = h_L = \sigma(W_{L-1,L}.h_{L-1} + b_L) \tag{1.8}$$

où $L$ est le nombre de couches du réseau et $\sigma$ est la fonction d'activation de la couche de sortie.

**Exemples d'application**

La seule différence entre les deux dernières équations réside dans la fonction d'activation. Son choix pour la couche de sortie est déterminé par la nature des données et la distribution des sorties désirée.

Dans le cas des problèmes de régression standard, la fonction d'activation est l'identité (équation 1.2). Pour les problèmes de classification binaire, on peut utiliser la fonction sigmoïde (équation 1.3). Finalement, pour les problèmes de classification multi-classes, la fonction $Softmax$ est la plus souvent utilisée pour représenter une distribution de probabilités sur $k$ classes différentes.

Soit $X$ un vecteur d'entrée et $i \in \{s_1 \dots s_k\}$ une classe de sortie, la fonction $Softmax$ est exprimée par :

$$Softmax(Z^i) = \frac{e^{Z^i}}{\sum_{j=1}^{k} e^{Z^j}} \quad avec \quad Z^i = W^i_{L-1,L}.h_{L-1} + b^i_L \tag{1.9}$$

où $W$ et $b$ sont respectivement les poids et le biais de la dernière couche cachée.

En notant $S$ la variable aléatoire désignant la classe correspondant au vecteur d'entrée $X$, la fonction $Softmax$ est un estimateur de $P(S = i|X)$, la probabilité d'appartenance du vecteur d'entrée $X$ à la classe $i$. On vérifie que :

$$\sum_{i \in \{s_1 \dots s_k\}} P(S = i|X) = 1 \tag{1.10}$$

Finalement, pour les problèmes de classification, la prédiction $y$ du modèle est la classe qui obtient la probabilité maximale :

$$y = \underset{i \in \{s_1 \dots s_k\}}{\operatorname{argmax}} \left(Softmax(Z_i)\right) \tag{1.11}$$

Les réseaux de neurones *feedforward* sont utilisés pour la modélisation acoustique [Jaitly *et al.*, 2012], la modélisation du langage [Bengio *et al.*, 2003, Schwenk *et al.*, 2006, Schwenk, 2013], la tâche de détection d'erreurs dans les transcriptions de la parole [Yik-Cheung *et al.*, 2014, Jalalvand et Falavigna, 2015], *etc.*

#### 1.1.2.2 Réseaux de neurones convolutifs

Les réseaux de neurones convolutifs (CNN pour *Convolutional Neural Network*) sont un type spécialisé de réseaux de neurones généralement utilisés quand l'entrée est structurée selon une grille (*grid-like topology* [Ian Goodfellow, 2016]), par exemple une grille de pixels pour une image.

Ces réseaux ont été inspirés par les travaux de Hubel et Wiesel [Hubel et Wiesel, 1962] sur le système visuel du chat, et plus particulièrement sur leurs propriétés : les champs récepteurs locaux et le partage des poids. Les

CNNs sont apparus dans les années 1980 avec les travaux de K. Fukushima [Fukushima, 1980] sur le Necognitron, et ont été popularisés avec les travaux de Y. LeCun [LeCun *et al.*, 1990] sur la reconnaissance des caractères dans les années 1990.

Généralement, l'architecture d'un CNN est composée d'une suite de couches de convolution et de sous-échantillonnage (*pooling* en anglais), suivie par des couches de neurones qui sont totalement connectées sous la forme d'un MLP, comme illustré dans la figure 1.3.

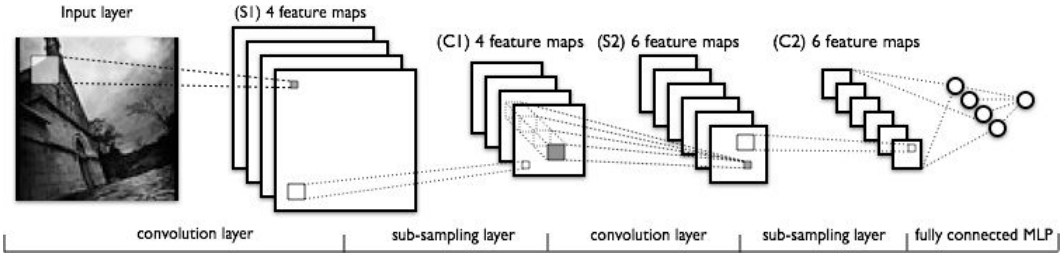



FIGURE 1.3 – Réseau de neurones convolutif composé de deux couches de convolution et de pooling, suivi de deux couches cachées et une couche Softmax [LeCun *et al.*, 1998].

Une convolution est une opération mathématique qui consiste à multiplier, ou « convoluer » une matrice (qui représente par exemple une image) par une autre appelée matrice de convolution, ou « filtre ». Les paramètres de la couche de convolution consistent en un ensemble de filtres apprenables. Chaque filtre est petit spatialement (le long de la largeur et de la hauteur), mais s'étend à travers toute la profondeur de l'image d'entrée. Pendant l'apprentissage, chaque filtre est convolué avec l'image. Ceci produit une carte de caractéristiques (*features map*) de 2-dimensions qui donnent les réponses de ce filtre à chaque position spatiale dans l'image. Intuitivement, le réseau va apprendre des filtres qui s'activent lorsqu'ils voient un type de caractéristique visuelle comme le contour, les orientations, *etc.*, dans la première couche du réseau. L'ensemble de filtres dans la couche de convolution produisent chacun une carte de caractéristiques, qui sont ensuite empilées le long de la dimension de profondeur pour former la sortie.

Les réseaux de neurones convolutifs sont caractérisés par les critères suivants :

— Des champs récepteurs locaux (*local receptive fields*) associés à des convolutions qui permettent d'extraire des formes élémentaires sur les données en entrées (image, son, texte, *etc.*), formant ainsi une *features map*. Autrement dit, chaque neurone dans cette carte n'est connecté qu'à une sous-région (champ récepteur local), correspondant à un certain nombre de neurones voisins dans la couche de neurones précédente.

— Un partage de poids d'une convolution est proposé afin de réduire le nombre de paramètres à apprendre dans le réseau. Il consiste à partager les poids entre tous les neurones d'une *features map,* c'est-à-dire appliquer les mêmes

poids d'une convolution (filtre) pour toutes les positions sur l'entrée. Celui-ci est convolué sur toute l'entrée en utilisant une fenêtre glissante, qui doit apprendre à capturer des caractéristiques locales.

— Des opérations de sous-échantillonnage (*pooling*) qui permettent d'une part de réduire le nombre de paramètres en réduisant la taille des *features map* et d'autre part, d'introduire de l'invariance aux (faibles) rotations et translations pouvant apparaître en entrée. Lorsqu'une opération de sous-échantillonnage est appliquée sur une région, la sortie reste approximativement la même, même si l'image est décalée ou pivotée de quelques pixels. Ceci s'explique par le fait que les opérations de sous-échantillonnage choisissent la même valeur dans une région (le maximum ou la moyenne) indépendamment de la position de l'image. Cette propriété est intéressante en traitement d'image, étant donné que la classification doit être invariante aux rotations et aux translations.

  Habituellement, deux opérations de *pooling* peuvent être utilisées : le maximum (max-pooling) ou la somme pondérée. Pour un voisinage donné, le max-pooling consiste à transférer en entrée de la couche suivante les valeurs des maxima locaux, alors que la somme pondérée consiste à transférer la moyenne pondérée. Ainsi le choix de l'opération dépend de la tâche elle-même. Par exemple pour des tâches de TALN, les auteurs dans [Collobert et Weston, 2008] ont utilisé le max-pooling. Ils considèrent que l'utilisation de la moyenne a pour conséquence un lissage peu opportun sur l'ensemble des mots d'une phrase, alors que le maximum tend à donner plus d'importance aux mots les plus saillants.

Cette architecture est utilisée avec succès dans différentes tâches, notamment la reconnaissance de caractères [LeCun *et al.*, 1990, LeCun *et al.*, 1998, Sermanet *et al.*, 2012], la modélisation acoustique [Hinton *et al.*, 2012, Abdel-Hamid *et al.*, 2012, Bengio et Heigold, 2014], ou comme déjà évoqué, pour des tâches de TALN [Collobert et Weston, 2008, Collobert *et al.*, 2011], *etc.*

#### 1.1.2.3 Réseaux de neurones récurrents

Les réseaux de neurones récurrents (RNN pour *Recurrent Neural Network*) [Medsker et Jain, 1999] sont une extension des réseaux de neurones *feedforward*. Ils sont capables de traiter une entrée de longueur variable. Ainsi, ils sont particulièrement utiles pour la classification automatique de séquences.

Un RNN traite la variabilité de la taille de l'entrée en utilisant des états cachés récurrents, aussi appelés nœuds de contexte. Cela permet de conserver des informations sur le contexte en apportant des éléments de l'instant $t-1$ lors du traitement de l'instant $t$. Le RNN présente un caractère dynamique du fait que ses poids dépendent non seulement des entrées apprises, mais également des sorties précédentes. Ces sorties peuvent être les sorties d'une couche cachée, dans ce cas on parle d'un RNN de type Elman [Elman, 1990] comme celui de la figure 1.4, ou bien les sorties principales et dans ce cas le RNN est de type Jordan [Jordan, 1997] comme celui de la figure 1.5.

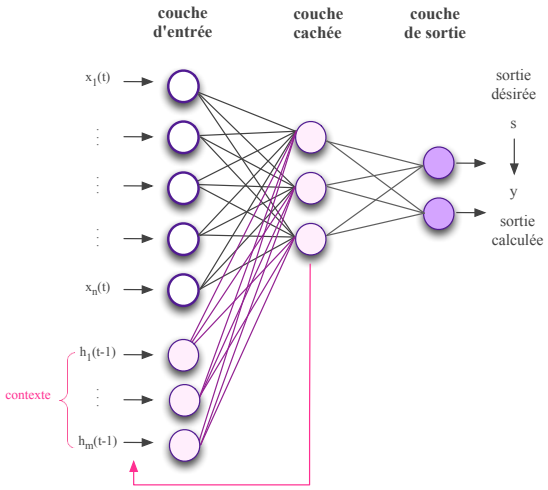



FIGURE 1.4 – Réseau de neurones récurrent de type Elman

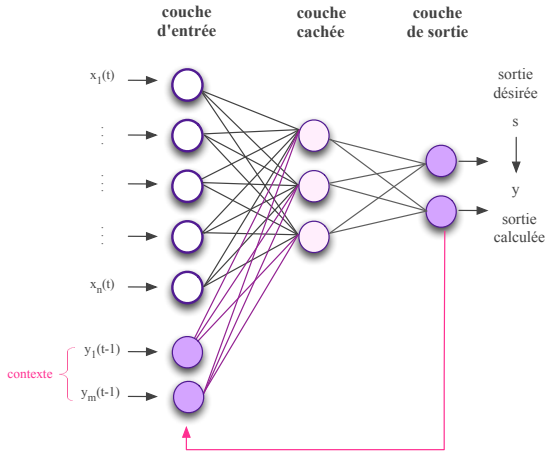



FIGURE 1.5 – Réseau de neurones récurrent de type Jordan

Plus précisément, ces deux architectures diffèrent dans la mesure où les nœuds de contexte $h_1(t-1), ..., h_m(t-1)$ des réseaux de type Elman sont alimentés par la couche cachée précédente alors que ceux de type Jordan $y_1(t-1), ..., y_m(t-1)$ sont alimentés par la couche de sortie.

Les performances des réseaux de neurones récurrents sur une tâche dépendent de la quantité d'information contextuelle à conserver. Cette information peut être étendue en permettant au réseau à l'instant $t$ l'accès aux informations futures $t+1$ et passées $t-1$. En pratique, ceci est réalisé en utilisant un réseau de neurones bidirectionnel (Bi-RNN) [Schuster et Paliwal, 1997] composé de deux couches cachées : une couche issue d'une passe « avant » (*forward*) qui lit l'entrée dans l'ordre (de $x_1$ à $x_T$), l'autre couche est issue d'une passe « arrière » (*backward*) qui lit l'entrée dans l'ordre inverse (de $x_T$ à $x_1$).

En reprenant les notations de l'équation 1.7 et en désignant par $t$ l'indice temporel, les séquences des états cachés « avant » (f) et « arrière » (b) sont calculés comme suit :

$$h_t^f = \varphi(W_h^f h_{t-1}^f + W_X^f X_t + b_h^f) \tag{1.12}$$

$$h_t^b = \varphi(W_h^b h_{t+1}^b + W_X^b X_t + b_h^b) \tag{1.13}$$

et donc la couche de sortie est calculée en fonction de $h_t^f$ et $h_t^b$ :

$$y(t) = \sigma(W_h^f . h_t^f + W_h^b . h_t^b + b_y) \tag{1.14}$$

où $\sigma$ est la fonction d'activation de la couche de sortie et $b_y$ est le biais.

Comme évoqué dans le paragraphe précédent, l'intérêt principal des RNNs est leur capacité à utiliser des informations contextuelles lors de l'apprentissage. Mal-

heureusement, il a été observé par Bengio [Bengio *et al.*, 1994] qu'il est difficile d'entrainer un RNN « classique » pour capturer les dépendances à long terme à cause de la variation exponentielle de l'erreur (gradient). En effet, l'influence d'une entrée sur les couches cachées augmente ou se dissipe de manière exponentielle au fur et à mesure qu'elle passe par les connexions récurrentes. Ceci s'explique par le fait que l'erreur locale à un instant $t$ rétro-propagée dans le temps s'exprime de manière récursive en fonction des erreurs rétro-propagées aux instants passés.

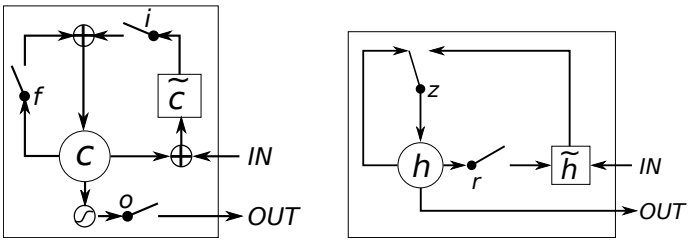
Plusieurs solutions ont été proposées dans la littérature pour pallier ce problème. La solution la plus utilisée dans l'état de l'art consiste à remplacer l'unité récurrente classique soit par une unité récurrente appelée LSTM pour *Long Short-Term Memory* (longue mémoire à court terme) [Hochreiter et Schmidhuber, 1997, Graves *et al.*, 2013] soit par une unité appelée GRU pour *Gated Recurrent Unit* [Cho *et al.*, 2014a].

La particularité de ces unités est liée à l'utilisation de portes multiplicatives. Ces portes sont des fonctions d'activation permettant de moduler le flux d'information à l'intérieur de l'unité.

Les deux unités sont illustrées dans la figure 1.6, et sont détaillées comme suit :

**LSTM** est composée d'une mémoire ($c$) et de trois portes. La partie d'entrée $i$ (input) doit choisir les informations pertinentes qui seront transmises à la mémoire. La sortie $o$ (output) protège le réseau du contenu de sa mémoire. La porte d'oubli $f$ (forget) permet à l'unité de remettre à zéro le contenu de sa mémoire.

**GRU** n'est composée que de deux portes. La porte de réinitialisation $r$ (reset) décide si l'état précédent de l'unité est ignoré ou non. La porte de modification $u$ (update) permet de décider si l'état caché $h$ doit être mis à jour avec le nouvel état caché $\tilde{h}$ ou non.




(a) Long Short-Term Memory (b) Gated Recurrent Unit

FIGURE 1.6 – Illustration des unités LSTM (a) et GRU (b). (a) $c$ et $\tilde{c}$ sont la mémoire et le nouveau contenu de la mémoire. (b) $h$ et $\tilde{h}$ sont l'activation et l'activation candidate [Cho *et al.*, 2014a].

Les différentes architectures de réseaux de neurones récurrents sont utilisées avec succès pour la modélisation du langage [Mikolov *et al.*, 2011b], la reconnaissance de la parole [Graves *et al.*, 2013, Graves et Jaitly, 2014], la compréhension de

la parole [Mesnil *et al.*, 2013, Mesnil *et al.*, 2015, Liu et Lane, 2016], pour la détection des erreurs dans les transcriptions de la parole [Ogawa et Hori, 2015], et à travers une architecture d'encodeur/décodeur pour faire de la traduction automatique [Cho *et al.*, 2014a, Bahdanau *et al.*, 2014, Firat *et al.*, 2016], *etc.*

#### 1.1.2.4 Auto-encodeurs

Les auto-encodeurs sont caractérisés par leur algorithme d'apprentissage classique qui vise à reproduire ses propres entrées. L'intérêt n'est pas la prédiction des entrées en tant que telles, mais les représentations latentes apprises par l'auto-encodeur au niveau de la couche cachée, qui sont capables de capturer suffisamment d'informations pour reconstruire les entrées. Ces représentations sont utilisées comme pré-entraînement de réseaux de neurones profonds [Vincent *et al.*, 2010, Erhan *et al.*, 2009, Bengio *et al.*, 2007b, Hinton et Zemel, 1994, Bengio, 2009b, Ian Goodfellow, 2016].

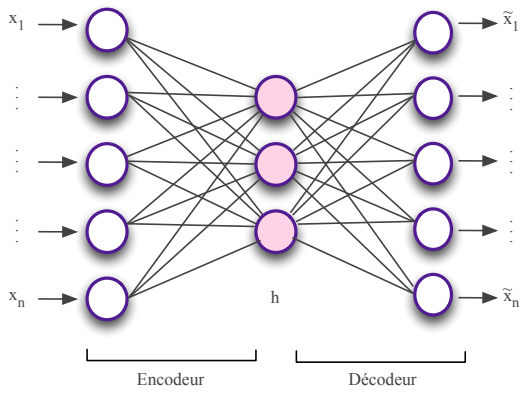



FIGURE 1.7 – Illustration de l'architecture d'un auto-encodeur. $X$ représente l'entrée, $h$ la couche de projection générée par l'encodeur alors que $\tilde{X}$ définit le vecteur d'entrée reconstruit par le décodeur.

L'architecture d'un auto-encodeur ordinaire est illustrée dans la figure 1.7. Il est composé de deux modules : un encodeur et un décodeur. L'encodeur sert à projeter l'entrée $X \in \mathbb{R}^n$ dans un espace de faible dimension. De cette projection résulte une nouvelle représentation $h$ de $X$ qui est susceptible de préserver les informations utiles de $X$. La couche de projection $h$ est obtenue en appliquant la formule suivante :

$$h = f_{enc}(W_{enc}.X + b_{enc}) \tag{1.15}$$

Dans cette équation $W_{enc}$ représente la matrice de poids, $b_{enc}$ le vecteur de biais, et $f_{enc}$ est une fonction d'activation.

Le décodeur est ensuite utilisé pour obtenir une reconstruction de l'entrée $X$ à partir de la représentation $h$. Le but du décodage est de vérifier si l'encodeur a

capturé l'information utile contenue dans les données fournies en entrée. La reconstruction est réalisée à l'aide de la formule suivante :

$$\tilde{X} = f_{dec}(W_{dec}.h + b_{dec}) \tag{1.16}$$

Les poids $W_{enc}$ et $W_{dec}$ peuvent être distincts ou les mêmes, avec $W_{dec} = W_{enc}^T$. Les paramètres $W_{enc}$, $W_{dec}$, $b_{enc}$ et $b_{dec}$ sont optimisés pour minimiser l'erreur $E$ de reconstruction (le coût). Le choix des fonctions de reconstruction $f_{dec}$ et de coût $(E)$ dépend de la tâche et de la distribution des données [Memisevic, 2011, Rudy et Taylor, 2014]. Si les données sont continues, on peut utiliser une fonction linéaire de reconstruction et l'erreur moyenne quadratique comme fonction de coût :

$$E(X, \tilde{X}) = \frac{1}{m} \sum_{i=1}^{m} \|X_i - \tilde{X}_i\|^2 \tag{1.17}$$

Par contre, si les données sont binaires ou dans l'intervalle $[0, 1]$, il est préférable d'utiliser une fonction sigmoid de reconstruction et l'entropie croisée moyenne comme fonction de coût :

$$E(X, \tilde{X}) = \frac{1}{m} \sum_{i=1}^{m} (X_i \log(\tilde{X}_i) + (1 - X_i) \log(1 - \tilde{X}_i)) \tag{1.18}$$

où $m$ est le nombre d'exemples d'apprentissage.

L'auto-encodeur peut apprendre deux représentations latentes différentes, cela dépend de la taille de la couche cachée. Dans le cas où la taille de la couche cachée est plus petite que l'entrée, l'auto-encodeur applique une compression des entrées pour réduire la redondance. Le réseau dans ce cas essaie d'apprendre une représentation compacte de l'entrée qui conserve l'information la plus pertinente. En revanche, dans le cas où la taille de la couche cachée est plus grande que l'entrée, le réseau apprend une représentation sur-complète (ou *overcomplete*) [Bengio *et al.*, 2007a]. Cette représentation peut être dans certains cas la fonction identité. L'une des solutions proposées pour éviter d'apprendre la fonction identité est d'utiliser un auto-encodeur débruitant (*denoising auto-encoder*) [Vincent *et al.*, 2008]. Ce dernier vise à apprendre une représentation robuste d'une entrée partiellement corrompue. L'idée est de forcer l'auto-encodeur à capter des informations suffisantes à la construction de l'entrée malgré sa corruption.

La différence entre l'auto-encodeur ordinaire et l'auto-encodeur débruitant vient de la notion de corruption aléatoire des entrées au cours de l'apprentissage dans le dernier cas. Cette corruption rend l'auto-encodeur plus généraliste en découvrant des paramètres plus robustes qu'un auto-encodeur ordinaire.

Les entrées peuvent être corrompues selon plusieurs manières : le plus simple est de masquer une partie de l'entrée en l'initialisant à zéro. On peut aussi utiliser le bruit « poivre et sel » pour les données binaires qui consiste à mettre un sous-ensemble aléatoire des entrées à 0 ou à 1 avec probabilité égale. Pour les entrées continues, le bruit « gaussien centré » est conseillé [Thibodeau-Laufer, 2014].

L'empilement et l'entraînement successif de ce modèle, connus sous le nom de *Stacked Denoising Auto-Encoder* (SDAE), servent à initialiser les poids d'une architecture profonde [Vincent *et al.*, 2010].

L'auto-encodeur contractant (contractive) [Rifai *et al.*, 2011] possède des propriétés semblables à celles des auto-encodeurs débruitants. Son principe consiste à ajouter une pénalité de contraction à la fonction de coût. Celle-ci pénalise la sensibilité du modèle aux variations des données fournies en entrée. Cela permet d'apprendre des représentations plus robustes aux variations des entrées.

### 1.1.3 Apprentissage des réseaux de neurones

Cette section présente de manière succincte les fondements théoriques des algorithmes d'apprentissage de réseaux de neurones.

Les réseaux de neurones sont généralement utilisés pour des tâches de classification supervisée. Ceci implique l'existence d'une base d'apprentissage composée de paires d'entrées-sorties $(x_1, ..., x_n; s_1, ..., s_k)$ liées par une certaine relation, que le réseau va apprendre en ajustant ses paramètres. Plusieurs méthodes d'apprentissage sont proposées dans la littérature. Le choix de l'algorithme d'apprentissage dépend non seulement de l'architecture du réseau, mais aussi des données d'apprentissage et du problème à traiter. Toutefois, l'algorithme de *rétro-propagation du gradient* introduit par Rumhelhart et al [Rumelhart *et al.*, 1986] reste l'algorithme le plus utilisé pour entrainer un MLP ou d'autres architectures. L'algorithme de rétro-propagation s'appuie sur la minimisation de l'erreur moyenne quadratique $E$ par la méthode de descente du gradient. L'erreur est calculée entre la sortie désirée $s$ et la sortie $y$ du MLP (la prédiction) comme défini dans l'équation 1.17.

Cet algorithme comporte deux étapes, la première est la « propagation avant » (*forward pass*) lors de laquelle les valeurs de sorties du réseau sont calculées. Une fois que l'erreur entre les valeurs de sorties estimées et les valeurs désirées est calculée, on passe à la deuxième étape qui est la« propagation arrière » (*backward pass*). Elle consiste à rétro-propager la dérivée partielle de l'erreur $\frac{\partial E}{\partial W}$ par rapport aux poids du réseau. Enfin, les poids sont mis à jour en fonction de cette dérivée partielle :

$$W' = W - \Delta W \tag{1.19}$$

$$\Delta W = -\lambda \frac{\partial E}{\partial W} \tag{1.20}$$

où $\lambda > 0$ est le taux d'apprentissage (*learning rate*), $W'$ est la matrice de poids mise à jour, et $W$ la matrice de poids utilisée lors de la passe avant.

On retrouve généralement trois variantes d'algorithme d'apprentissage :

— La rétro-propagation stochastique (*stochastic backpropagation*) consiste à mettre à jour les poids à chaque présentation d'un exemple d'apprentissage au système.

— Si la mise à jour des poids est effectuée selon la moyenne des gradients sur tous les exemples d'apprentissage qui forment un lot, on parlera de rétro-

propagation par lot (*batch ou deterministic backpropagation*).

— L'algorithme le plus utilisé pour l'apprentissage profond se situe quelque part entre les deux, en utilisant plus d'un exemple, mais moins que tous les exemples d'apprentissage. Celui-ci est appelé la rétro-propagation par mini-lots (*minibatch backpropagation*). Cet algorithme est avantageux par rapport à la méthode stochastique : d'une part parce qu'il y a moins de risque de surapprentissage vu que le gradient est moyenné ; d'autre part, car il est plus rapide du fait qu'il y a moins de mises à jour de poids.

On appelle *itération* l'application de l'algorithme sur un exemple ou sur un lot d'apprentissage, et *époque* son application sur tous les exemples d'apprentissage. Le processus décrit précédemment est répété sur plusieurs époques, jusqu'à la convergence vers la bonne solution, c'est-à-dire quand $E$ devient quasi-nulle, ce qui n'est pas toujours possible. Les paramètres retenus comme étant les plus performants sont ceux ayant obtenu le meilleur score d'évaluation sur un ensemble de données non utilisé pendant l'apprentissage, appelé corpus de validation. Notons que la métrique utilisée sur le corpus de validation est parfois différente de la fonction de coût utilisée lors de l'apprentissage. Le procédé, consistant à arrêter l'apprentissage lorsque le score d'évaluation sur le corpus de validation ne s'améliore pas au bout d'un certain nombre d'époques, s'appelle le *early stopping*.

À noter que le taux d'apprentissage, la taille de mini-batch, le taille et le nombre de couches cachées et le nombre d'époques sont un ensemble minimum d'hyperparamètres à optimiser. Ils ont un grand impact sur les performances finales.

Le taux d'apprentissage détermine le changement relatif des poids d'une itération à une autre. Il sert à contrôler la convergence de l'algorithme : plus sa valeur est petite, plus la convergence sera longue. Par contre, si on lui donne une valeur plus grande pour accélérer l'apprentissage, l'apprentissage risque de rendre le réseau instable : il est possible que des phénomènes d'oscillation apparaissent pendant l'apprentissage. Pour contrer ce phénomène qui peut également survenir avec un taux d'apprentissage petit, plusieurs solutions ont été proposées :

— L'adaptation du taux d'apprentissage consiste à faire varier la valeur du taux d'apprentissage pendant l'apprentissage. De nombreuses méthodes ont été proposées pour modifier et adapter le taux d'apprentissage afin d'accélérer et d'améliorer la convergence, on peut citer par exemple AdaDelta (*Adaptive learning rate*) [Bottou, 2010, Bergstra et Bengio, 2012, Zeiler, 2012, Schaul *et al.*, 2013].

— Le *Momentum* est une variante de l'algorithme de rétro-propagation [Plaut *et al.*, 1986, Hinton, 1978]. Il consiste à prendre en compte les mises à jour précédentes pour faire la mise à jour des poids à l'instant $t$. Ceci est fait en ajoutant un second terme appelé *momentum* à l'équation 1.20 :

$$\Delta W(t) = -\lambda \frac{\partial E}{\partial W}(t) + \alpha.\Delta W(t-1) \tag{1.21}$$

où $\alpha$ est un coefficient de pondération compris entre 0 et 1. Plu-

sieurs variantes de l'algorithme *Momentum* ont été introduites récemment, par exemple dans [Sutskever *et al.*, 2013].

L'accélération de la convergence du réseau pour atteindre le minimum local est importante, mais cela ne garantit pas que le réseau va généraliser correctement, et ne pas sur-apprendre. C'est pourquoi, il existe plusieurs méthodes pour prévenir le sur-apprentissage du réseau, dont :

— La régularisation, connue aussi sous le terme anglais de *Weight Decay* [Collobert et Bengio, 2004, Bengio *et al.*, 2005], est proposée afin d'éviter le sur-apprentissage et améliorer la généralisation du réseau. Cette méthode consiste à ajouter à la fonction de coût $E$ un terme de régularisation. Ce terme va contrôler les valeurs des poids pendant l'apprentissage. En effet, il pénalise les poids qui ont de grandes valeurs, en les poussant vers de plus faibles valeurs. Les choix les plus populaires sont les deux termes de régularisation :

  - $L1$ : somme des poids au carré.
  - $L2$ : somme des valeurs absolues des poids.

  Ces termes vont croitre en fonction des valeurs des poids. De plus, ils sont pondérés par des coefficients de régularisation pour contrôler leurs valeurs et éviter des réductions inutiles sur les valeurs de poids [Bengio, 2012].

— Le pré-entraînement (*pretraining*) consiste à initialiser les poids du réseau à l'aide d'auto-encodeurs ou de variantes des machines de *Boltzmann* [Salakhutdinov et Hinton, 2009]. Ceci permet de converger vers un minimum local plus rapidement qu'avec une initialisation aléatoire des poids. L'apprentissage du réseau de neurones avec cette méthode consiste à initialiser les couches cachées par des poids déjà appris, puis de faire l'apprentissage de tout le réseau (*fine-tuning*). Les auteurs dans [Hinton *et al.*, 2006, Bengio *et al.*, 2007a, Erhan *et al.*, 2010] ont montré que cette méthode donne généralement une meilleure généralisation et un minimum local plus intéressant que l'apprentissage du réseau avec des poids initialisés aléatoirement.

— La normalisation par lots (*Batch-Normalisation*) consiste à normaliser les mini-lots au moment de l'apprentissage. Les auteurs dans [Ioffe et Szegedy, 2015] proposent cette approche pour résoudre le problème de *Internal Covariante Shift*, défini comme le changement de la distribution des entrées des couches du réseau dû à la modification de ces paramètres au moment de l'apprentissage. Cela ralentit l'apprentissage en exigeant des taux d'apprentissage plus faibles et une initialisation prudente des paramètres. L'ajout de la normalisation par lots à un système état de l'art pour la classification d'image, permet d'obtenir une accélération substantielle de l'apprentissage, et une amélioration de sa performance en augmentant les taux d'apprentissage et modifiant d'autres paramètres.

## 1.2 Représentations continues de mots

Dans cette section, nous allons aborder la deuxième partie de ce chapitre qui concerne l'état de l'art sur les représentations continues de mots. Le succès des méthodes d'apprentissage profond dépend beaucoup de la représentation des données, puisque ces représentations peuvent coder différents types de relations cachées (syntaxiques, sémantiques, *etc.*) dans les données. Les tâches conventionnelles de traitement de langage naturel (TALN) utilisent souvent la représentation *one-hot*, qui consiste à représenter chaque mot du vocabulaire par un vecteur de la taille du vocabulaire contenant un seul élément non nul. Toutefois, cette représentation simple du mot se heurte à plusieurs difficultés. La plus critique est que cette représentation ne peut pas capter de relations entre les mots, même s'il existe une forte corrélation sémantique ou syntaxique entre certains d'entre eux.

D'autres stratégies ont émergé pour capturer les similarités sémantiques et syntaxiques des mots. La plupart s'appuie sur un paradigme très commun pour l'acquisition de ces représentations, qui repose sur *l'hypothèse de distribution* de [Harris, 1954], indiquant que les mots dans des contextes similaires ont des significations semblables.

En se focalisant sur l'hypothèse de distribution, de nombreuses méthodes ont été étudiées dans la communauté TALN pour la construction de représentations de mots, qui permettent de capturer des liens entre les mots. Ces méthodes se répartissent en trois catégories : clustering, représentations distributionnelles et représentations distribuées [Turian *et al.*, 2010, Levy et Goldberg, 2014]. Nous allons présenter dans les sections suivantes ces trois catégories, en détaillant plus particulièrement les représentations distribuées, sur lesquelles s'appuient ces travaux de thèse.

### 1.2.1 Clustering

Cette méthode consiste à regrouper les mots en clusters en s'appuyant sur leurs contextes [Brown *et al.*, 1992, Uszkoreit et Brants, 2008]. L'approche de Brown s'appuie sur l'utilisation d'un algorithme hiérarchique qui regroupe les mots pour maximiser l'information mutuelle des bigrammes. La nature hiérarchique du regroupement signifie que nous pouvons choisir la classe de mots à plusieurs niveaux dans la hiérarchie, ce qui peut ne pas pénaliser les clusters ayant un petit nombre de mots. Un inconvénient de cette méthode est qu'elle se fonde uniquement sur les statistiques bigrammes, et ne considère pas l'usage des mots dans un contexte plus large.

### 1.2.2 Représentations distributionnelles

La construction de ces représentations se base sur une matrice de co-occurrences de mots, c'est pourquoi elles sont connues sous le terme de *count based models* ou *global matrix factorization*. Le mot est représenté par un vecteur *sparse* de très haute dimension, dans lequel chaque entrée est une mesure d'association entre le mot et un contexte particulier. La mesure la plus populaire de cette association est

l'information mutuelle PMI (Pointwise Mutual Information [Church et Hanks, 1990] (voir [Turney *et al.*, 2010] pour plus de détails). Dans certains travaux, la dimension de ce vecteur est réduite en utilisant soit la décomposition en valeurs singulières SVD (Singular Value Decomposition) [Bullinaria et Levy, 2007], soit l'allocation latente de Dirichlet (LDA) [Blei *et al.*, 2003], soit l'analyse sémantique latente (LSA) [Dumais, 2004].

### 1.2.3 Représentations distribuées : *Word embeddings*

Plus récemment, il a été proposé de représenter le mot par un vecteur dense, de faible [1] dimension avec valeurs réelles. Ces représentations sont appelées *word embeddings*, *neural embeddings* ou *prediction-based embeddings*, ou en français *représentation vectorielle du mot*, *représentation continue du mot* et aussi *plongement de mot*. Elles ont été introduites à travers la construction de modèles de langage neuronaux [Bengio *et al.*, 2003, Schwenk *et al.*, 2006, Schwenk, 2013].

Les *word embeddings* ont été utilisés avec succès en tant qu'informations supplémentaires dans plusieurs tâches, notamment pour les tâches TALN : l'étiquetage morpho-syntaxique, le regroupement en syntagme, la reconnaissance d'entités nommées, la détection de mention [Bansal *et al.*, 2014a, Turian *et al.*, 2010, Collobert *et al.*, 2011], la compréhension de la parole [Mesnil *et al.*, 2013, Yao *et al.*, 2014, Mesnil *et al.*, 2015, Liu et Lane, 2016], *etc.*

Les *word embeddings* constituent une projection des mots du vocabulaire dans un espace de faible dimension de manière à préserver les similarités sémantiques et syntaxiques. Ainsi, si les vecteurs de mots sont proches les uns des autres en terme de distance, les mots doivent être sémantiquement ou syntaxiquement proches. Chaque dimension représente une caractéristique latente du mot, qui peut capter des propriétés syntaxiques et sémantiques.

Plusieurs approches neuronales ont été proposées dans la littérature pour la construction des *word embeddings*, dont certaines sont détaillées ci-dessous.

#### 1.2.3.1 Modèles de langage neuronaux

Les *word embeddings* ont été introduits à travers la construction de modèles de langages neuronaux [Bengio *et al.*, 2003, Schwenk, 2007]. Ce modèle consiste à apprendre un réseau de neurones *feedforward* pour estimer la probabilité du prochain mot, en s'appuyant sur la représentation continue des mots précédents. Cette représentation est apprise au fur et à mesure au moment de l'apprentissage du réseau de neurones.

Cette approche doit effectuer deux tâches : d'abord, la projection des indices des mots du contexte dans un espace continu afin d'obtenir leurs représentations vectorielles. Ces représentations sont ensuite concaténées pour construire l'entrée de

1. C'est un peu étrange de dire faible dimension pour un vecteur qui possède une centaine de dimensions, mais c'est le terme le plus employé dans la littérature. En effet, on dit faible dimension par rapport à la dimension d'un vecteur one-hot.

la couche cachée. Deuxièmement, la probabilité n-gramme de chaque mot dans une liste de mots est calculée en sortie du réseau.

Ces deux tâches peuvent être effectuées par un réseau de neurones composé de deux couches cachées. La première correspond à la représentation continue de tous les mots du contexte et la deuxième couche cachée est nécessaire pour obtenir une estimation de probabilité (voir figure 1.8).

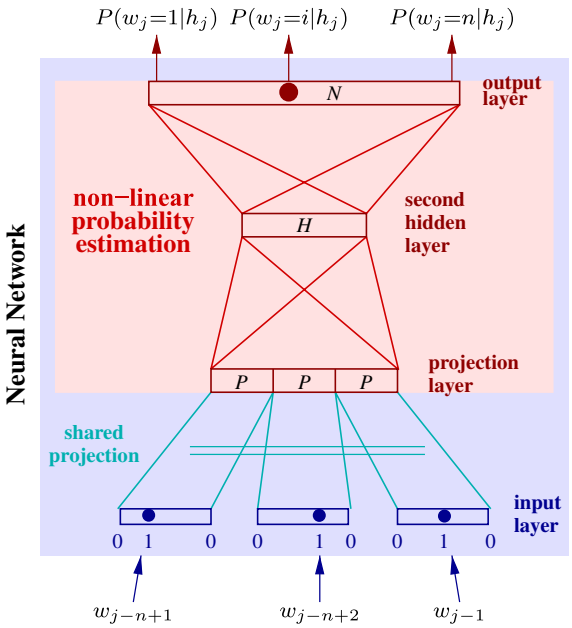



FIGURE 1.8 – Architecture neuronale pour la modélisation du langage [Schwenk, 2007].

Dans la couche de sortie, chaque neurone correspond à la probabilité d'un mot en fonction du contexte (calculée par la fonction Softmax afin de garantir que la somme de toutes les probabilités soit égale à 1). Dans l'approche de Bengio [Bengio *et al.*, 2003] la couche de sortie contient tous les mots du vocabulaire. Avec cette approche, l'utilisation d'un grand vocabulaire peut être très couteuse en termes de temps d'exécution, le temps de calcul étant linéaire par rapport à la taille du vocabulaire. C'est pour cette raison que Schwenk [Schwenk, 2007] a proposé dans son modèle nommé CSLM (*Continuous Space Language Models*), de se restreindre à une liste des mots fréquents (*short list*).

#### 1.2.3.2 Collobert et Weston

Collobert et Weston (C&W) [Collobert et Weston, 2008] présentent une autre approche pour la modélisation neuronale du langage. Ils proposent une solution au problème soulevé dans la section précédente. Pour éviter le calcul de la couche de sortie sur tout le vocabulaire, ils ont proposé d'utiliser une autre fonction de

coût pour l'apprentissage. En effet, au lieu d'utiliser l'entropie croisée comme dans [Bengio *et al.*, 2003, Schwenk, 2013], qui maximise la probabilité du prochain mot sachant les mots précédents, Collobert et Weston [Collobert et Weston, 2008] entraînent le modèle de langage à discriminer entre deux classes : si le mot du milieu d'un n-gramme en entrée du réseau est lié à son contexte ou non. Autrement dit, le réseau de neurones est entrainé pour discriminer les scores entre un n-gramme correct et un n-gramme corrompu. La corruption consiste à remplacer aléatoirement le mot central par un autre mot du vocabulaire $V$. L'entrainement du réseau de neurones consiste à minimiser la fonction de coût (équation 1.22), qui est de type classement (*ranking-type cost*), sur tous les n-grammes $G$ dans le corpus :

$$E(g) = \sum_{g \in G} \sum_{\tilde{g} \in V} max(0, 1 - s(g) + s(\tilde{g})) \tag{1.22}$$

où $s(.)$ est le score d'existence du n-gramme, $g$ est le n-gramme correct choisi à partir d'un ensemble de tous les n-grammes possibles dans le corpus $G$, $\tilde{g}$ est le n-gramme corrompu choisi pour chaque n-gramme $g$.

L'architecture du modèle est illustrée dans la figure 1.9. Elle est composée de trois couches cachées : une couche de projection (*Lookup Table*) suivie d'une couche linéaire et une non linéaire (Tanh). La dernière couche calcule en sortie un score pour un n-gramme donné.

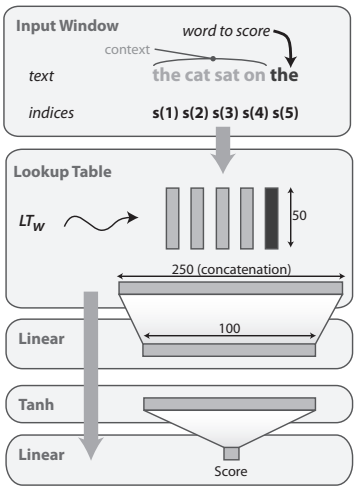



FIGURE 1.9 – Architecture d'un réseau de neurone profond qui calcule le score d'un mot donné sachant les mots en contexte [Collobert et Weston, 2008].

Cette approche est ensuite revisitée par Turian [Turian *et al.*, 2010]. Il a proposé pour la corruption du n-gramme de remplacer le dernier mot au lieu de celui au milieu. Il a proposé également d'utiliser des taux d'apprentissage différents pour les *embeddings* et pour les poids du réseau de neurones.

#### 1.2.3.3 Word2vec

Les *word embeddings* actuellement les plus populaires dans la littérature sont fournis par la boite à outils *Word2vec* [Mikolov *et al.*, 2013a, Mikolov *et al.*, 2013b].

[Mikolov *et al.*, 2013a] ont proposé d'abord deux architectures (CBOW et Skip-gram) pour l'apprentissage des *word embeddings* qui sont moins coûteuses en termes de temps de calcul que les modèles précédents. Puis, ils ont amélioré ces modèles en employant des stratégies supplémentaires pour améliorer la vitesse d'apprentissage et la performance des *word embeddings* [Mikolov *et al.*, 2013b].

Ces architectures offrent deux principaux avantages par rapport aux modèles CSLM et C&W :

— ils ne contiennent pas de couches cachées ;

— ils permettent au modèle de langage de prendre en compte un contexte supplémentaire.

Pour des raisons de complexité algorithmique de la couche Softmax, les auteurs dans [Mikolov *et al.*, 2013b] ont proposé d'autres alternatives appelées échantillons négatifs (en anglais *Negative Sampling*) et Softmax hiérarchique.

Le succès de *Word2vec* est non seulement dû à ces changements dans l'architecture du réseau, mais surtout en raison des algorithmes d'apprentissage (*Negative Sampling* [Mnih et Teh, 2012, Gutmann et Hyvärinen, 2012] et Softmax hiérarchique [Morin et Bengio, 2005]) qui ne sont pas utilisés dans les approches décrites dans les sections précédentes.

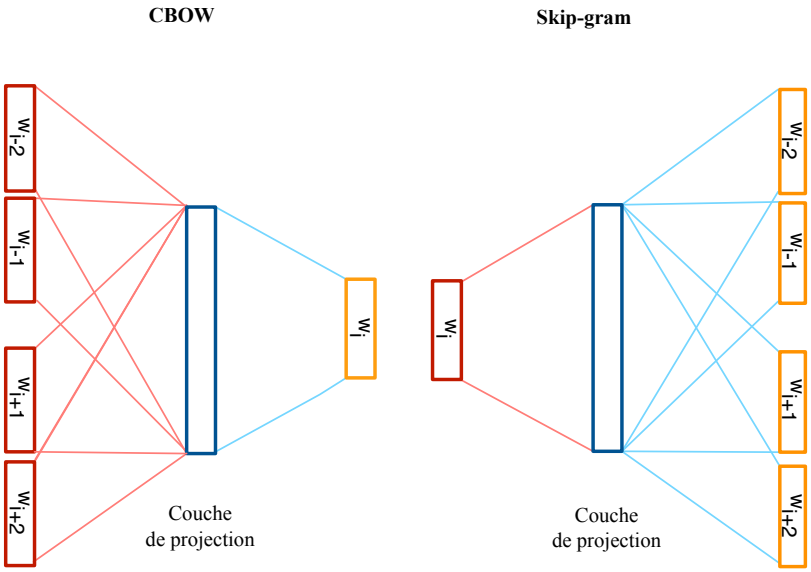



FIGURE 1.10 – L'architecture CBOW prédit le mot courant en fonction du contexte, et Skip-gram prédit les mots en contexte en fonction du mot courant.

Dans ce qui suit, nous allons présenter les deux architectures CBOW et Skip-gram illustrées par la figure 1.10.

## CBOW

La première architecture permet de prédire un mot $w_i$ en fonction de son contexte, qui correspond aux mots précédents et aux mots suivants. Dans cette architecture, la couche de projection est partagée par tous les mots : tous les mots sont projetés dans la même position. Ceci est différent de l'approche utilisée pour le modèle de langage (CSLM), où les mots sont projetés dans des positions différentes.

Ici, l'apprentissage des *word embeddings* consiste à prédire un mot en fonction de son contexte. Ceci est fait en calculant la somme des *word embeddings* du contexte, puis en appliquant sur le vecteur résultant un classifieur log-linéaire pour prédire le mot cible. Enfin, le modèle compare sa prédiction avec la réalité et corrige la représentation vectorielle du mot par rétro-propagation du gradient. Ce modèle cherche à maximiser l'équation suivante :

$$S = \frac{1}{I} \sum_{i=1}^{I} \log p(m_i | m_{i-n}, ..., m_{i-1}, m_{i+1}..., m_{i+n}) \tag{1.23}$$

où $I$ correspond au nombre de mots dans le corpus, le modèle reçoit une fenêtre de $n$ mots autour du mot cible $m_i$.

Ce modèle est nommé CBOW pour sac-de-mots continus ou *Continuous Bag-of-Words* pour deux raisons. D'une part, l'appelation sac-de-mots indique que l'ordre des mots du contexte n'a pas d'influence sur la projection. D'autre part, l'objectif souligne l'utilisation d'une représentation continue (*word embeddings*) des mots en contexte, ce qui diffère des modèles sac-de-mots standards.

## Skip-gram

L'architecture skip-gram est l'inverse de CBOW. Le mot cible $m_i$ est maintenant utilisé comme entrée, et les mots de contexte sont utilisés en sortie. Elle consiste à prédire, pour un mot donné, le contexte dont il est issu.

Dans cette architecture le mot en entrée est projeté dans la couche cachée puis dans la couche de sortie pour produire un vecteur. Ce vecteur est ensuite comparé à chacun des mots du contexte et le réseau se corrige par rétro-propagation du gradient. De cette manière, la représentation vectorielle du mot d'entrée va être proche de celles des mots qui produisent le même contexte que lui. Cette architecture maximise l'équation suivante :

$$S = \frac{1}{I} \sum_{i=1}^{I} \sum_{-n \leq j \leq n, \neq 0} \log p(m_i | m_{i+j}) \tag{1.24}$$

L'architecture Skip-gram avec échantillons négatifs (*Negative Sampling*) cherche à représenter chaque mot $m_i \in V_I$ et chaque contexte $c \in V_C$ par des vecteurs de d-dimensions ($\overrightarrow{m_i}$, $\overrightarrow{c}$), afin d'avoir des représentations vectorielles similaires pour les mots similaires. Ceci est fait d'une part, en maximisant le produit scalaire $\overrightarrow{m_i}.\overrightarrow{c}$ pour les paires ($m_i$,c) qui apparaissent dans le corpus $D$ ; d'autre part, en minimisant les exemples négatifs ($m_i$,$c_{Neg}$), qui ne sont pas nécessairement observés dans D. Les exemples négatifs sont créés en corrompant stochastiquement les paires ($m_i$,c) observées dans D, d'où le nom d'échantillons négatifs.

En outre, le contexte ne se limite pas au contexte immédiat, et les instances d'apprentissage peuvent être créées en ignorant (sautant) un nombre de mots dans son contexte, par exemple, $m_{i-4}, m_{i-3}, m_{i+3}, m_{i+4}$, d'où le nom de *Skip-gram*.

#### 1.2.3.4 Word2vec-deps

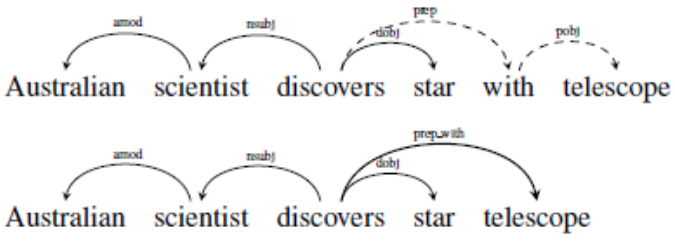
La méthode *Word2vec-deps* (*dependency based word embeddings*), introduite par [Levy et Goldberg, 2014], est une extension de *Word2vec* qui propose d'extraire les mots en contexte en se basant sur les relations syntaxiques.

Ce modèle est une généralisation du modèle *skip-gram* avec échantillons négatifs, qui nécessite un corpus annoté pour l'apprentissage, contrairement aux approches précédentes. En effet, le corpus d'apprentissage doit être enrichi par des informations syntaxiques (mots racines, lien de dépendance entre le mot racine et les mots dépendants, *etc.*), produites par exemple par un analyseur de dépendances. Cela revient à construire des arbres syntaxiques de dépendance pour extraire les mots en contexte d'un mot cible en fonction des relations syntaxiques. Cette méthode permet d'avoir une fenêtre de contexte de taille variable.

Après avoir analysé chaque phrase, les contextes du mot sont obtenus comme suit : pour chaque mot cible $m$, ces dépendants $dep_1, ..., dep_k$ et la racine $h$ de la phrase, on obtient les contextes $((dep_1, lbl_1), ..., (dep_k, lbl_k), (dep_k, lbl_h^{-1}))$, où $lbl$ est le type de relation de dépendance entre la racine et le dépendant (*nsubj*, *dobj*, *prep-with*, *amod*, *etc.*), et $lbl^{-1}$ est utilisé pour marquer la relation inverse.

Il est à noter que les relations syntaxiques de type préposition sont omises et les mots de types prépositions ne sont pas pris en compte au moment de l'extraction de contextes du mot cible. Dans ce cas, la racine est liée directement à l'objet de la préposition, et la préposition elle-même est incorporée avec l'étiquette de dépendance, comme illustré dans la figure 1.11.

Les dépendances syntaxiques sont à la fois plus inclusives et plus ciblées que l'approche sac-de-mots. Ells permettent de capter des relations entre mots éloignés qu'il serait impossible de détecter dans une fenêtre de contexte de taille réduite. De plus, elles filtrent les contextes qui se trouvent dans la fenêtre, mais pas directement liés au mot cible. C'est à dire que seuls les mots de contexte directement liés au mot cible sont utilisés.




| WORD | CONTEXTS |
|---|---|
| australian | scientist/amod$^{-1}$ |
| scientist | australian/amod, discovers/nsubj$^{-1}$ |
| discovers | scientist/nsubj, star/dobj, telescope/prep_with |
| star | discovers/dobj$^{-1}$ |
| telescope | discovers/prep_with$^{-1}$ |

FIGURE 1.11 – Un exemple d'extraction de contexte. En Haut : les relations de prépositions sont regroupées en arcs simples, ce qui rend le « télescope » en lien direct avec « discovers ». En bas : les contextes extraits pour chaque mot dans la phrase [Levy et Goldberg, 2014].

#### 1.2.3.5 GloVe

*GloVe* est une approche récente proposée par [Pennington *et al.*, 2014], qui réunit deux approches : la factorisation de matrice « count-based » (section 1.2.3) et les modèles prédictifs ou neuronaux. Cette approche est considérée comme modèle prédictif par [Levy *et al.*, 2015, Li et Jurafsky, 2015] et comme modèle « count-based » par [Arora *et al.*, 2016].

Cette approche repose sur la construction d'une matrice $MG$ de co-occurrence globale des mots, en traitant le corpus en utilisant une fenêtre contextuelle glissante. Ici, chaque élément $MG_{ij}$ représente le nombre de fois où le mot $m_j$ apparaît dans le contexte du mot $m_i$.

*GloVe* est un modèle d'apprentissage non supervisé qui prend en compte toute l'information portée par le corpus et non pas la seule information portée par une fenêtre de mots, d'où le nom *GloVe*, pour VEcteur GLObal.

Une fois la matrice $MG$ calculée, un modèle de regression par moindres carrés est entrainé pour construire des représentations vectorielles $\overrightarrow{m_i}$ et $\overrightarrow{m_j}$. Ces représentations doivent conserver des informations utiles sur la co-occurrence de paire de mots $m_i$ et $m_j$, telles que :

$$\overrightarrow{m_i}^T \overrightarrow{m_j} + b_i + b_j = logMG_{ij} \tag{1.25}$$

où $b_i$ et $b_j$ sont les vecteurs biais associés respectivement pour les mots $m_i$ et $m_j$.

Ceci est accompli en minimisant la fonction de coût, qui évalue la somme des

erreurs au carré :

$$E = \sum_{i,j=1}^{nv} f(MG_{ij})(\overrightarrow{m_i}^T \overrightarrow{m_j} + b_i + b_j - logMG_{ij})^2 \tag{1.26}$$

où $nv$ est la taille du vocabulaire, $f(.)$ est une fonction de pondération, qui pondère le coût en fonction de la fréquence du nombre de co-occurrence $MG_{ij}$. Celle-ci est définie comme suit :

$$f(MG_{ij}) = \begin{cases} (\frac{MG_{ij}}{MG_{max}})^{\alpha} & si\ MG_{ij} < MG_{max} \\ 1 & sinon \end{cases}$$

où $MG_{max} = 100$ et $\alpha = 3/4$.

Lorsque la valeur de co-occurence $MG_{ij}$ d'une paire de mots est très élevée, c'est-à-dire supérieure à une valeur maximale $MG_{max}$, la fonction ne prend pas en compte cette valeur et simplement retourne 1. Sinon, pour les autres paires, elle envoie un poids entre (0 et 1), où la distribution de poids dans cette plage est décidée par $\alpha$.

### 1.2.4 Techniques de visualisation

Pour avoir une idée plus claire sur les représentations vectorielles de mots, nous pouvons les visualiser de deux manières différentes.

La première consiste à utiliser la technique t-SNE (*t-Distributed Stochastic Neighbor Embedding*) [Maaten et Hinton, 2008]. C'est une technique non linéaire de réduction de dimensions, qui est particulièrement adaptée pour la projection des données à haute dimension dans un espace de deux ou trois dimensions. Ces représentations peuvent être visualisées sous forme de nuage de points. La figure 1.12 illustre une partie d'une représentation $2D$ des *word embeddings*. On peut voir des îlots qui portent des informations différentes, le nuage de mots à gauche porte une information de type numérique, alors que celui à droite porte une information sur l'emploi.

La deuxième est une méthode qualitative, qui se base sur le calcul de la distance cosinus pour rechercher les mots les plus proches (similaires) du *word embedding* d'un mot donné. La figure 1.13 présente un exemple de mots et leurs cinq mots les plus proches. Ces mots similaires sont extraits à partir de différents *word embeddings*, qui sont : *CBOW* avec une fenêtre de contexte de taille 5 (BoW5) et 2 (BoW2) et *word2vec-deps* (Deps).

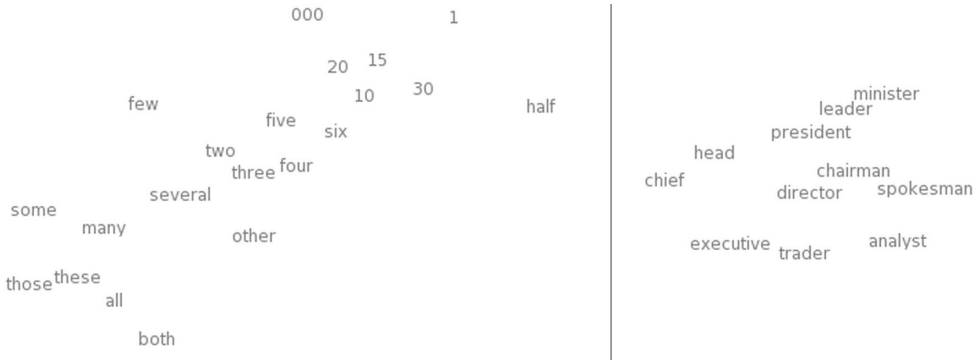



FIGURE 1.12 – Visualisation de *word embeddings* [Turian *et al.*, 2010] avec t-SNE. Gauche : Nuage de mots qui portent une information de type numérique ; Droit : Nuage de mots qui portent une information sur l'emploi. [Turian *et al.*, 2010]

| **Target Word** | BOW5 | BOW2 | DEPS |
|---|---|---|---|
| batman | nightwing<br>aquaman<br>catwoman<br>superman<br>manhunter | superman<br>superboy<br>aquaman<br>catwoman<br>batgirl | superman<br>superboy<br>supergirl<br>catwoman<br>aquaman |
| hogwarts | dumbledore<br>hallows<br>half-blood<br>malfoy<br>snape | evernight<br>sunnydale<br>garderobe<br>blandings<br>collinwood | sunnydale<br>collinwood<br>calarts<br>greendale<br>millfield |
| turing | nondeterministic<br>non-deterministic<br>computability<br>deterministic<br>finite-state | non-deterministic<br>finite-state<br>nondeterministic<br>buchi<br>primality | pauling<br>hotelling<br>heting<br>lessing<br>hamming |

FIGURE 1.13 – Mots cibles et leurs 5 mots les plus similaires, extraits à partir de deux types de *word embeddings* : *Word2vec-deps* et *CBOW* [Levy et Goldberg, 2014].

On remarque que les *word embeddings* BoW5, BoW2 et Deps obtiennent des résultats similaires pour le mot « batman », par contre ce n'est pas le cas pour les autres mots. Par exemple, les voisins le plus proches du mot « Hogwarts » (l'école de magie de la série « Harry Potter ») sont liés à l'aspect de domaine lorsqu'ils sont induits par les variantes de *CBOW*, et correspondent aux noms d'écoles connues lorsqu'ils sont induits par *Deps*.

Ces deux relations sont connues comme étant la similarité associative (*relatedness*) et la similarité fonctionnelle dans la littérature [Gracia et Mena, 2008, Turney, 2012, Kiela *et al.*, 2015]. De plus, pour les deux variantes de *CBOW*, on remarque que plus la fenêtre de contexte est large plus le degré d'associativité entre les paires de mots est élevé.

## 1.3 Conclusion

Dans ce chapitre, nous avons abordé la première partie de l'état de l'art de ce manuscrit qui porte sur les techniques d'apprentissage profonds. Un état de l'art détaillé sur l'apprentissage profond est donné dans [Bengio, 2012, LeCun *et al.*, 2012, Ian Goodfellow et Courville, 2016].

Nous nous sommes intéressés dans un premier temps à présenter différents types d'architectures neuronales, tout en exposant les fondements théoriques des algorithmes d'apprentissage de réseau de neurones. Ensuite, nous avons exposé les différentes approches pour la construction des représentations vectorielles de mots qui sont : clustering, distributionnelles et distribuées. Enfin, nous avons montré deux techniques de visualisation de ces représentations de mots.

Nous présenterons dans le chapitre suivant, un état de l'art sur la reconnaissance automatique de la parole.

CHAPITRE 2

# LA RECONNAISSANCE AUTOMATIQUE DE LA PAROLE CONTINUE

**Sommaire**

L'évolution technologique et scientifique dans le domaine de la reconnaissance automatique de la parole permet aujourd'hui de fournir des Systèmes de Reconnaissance Automatique de la Parole (SRAP) performants, permettant dans de nombreuses situations d'obtenir une transcription de la parole de très bonne qualité. Cependant, des erreurs sont inévitables, et certaines d'entre elles demeurent très pénalisantes. Elles sont liées d'une part à la sensibilité des SRAP à la variabilité (variabilité de l'environnement acoustique, du locuteur, du style de langage, de la thématique du discours, *etc.*) et d'autre part, à la difficulté de découpage de la parole en mots. En effet, pour un SRAP, le signal de la parole est un flux continu et aucune frontière correspondant aux unités lexicales n'est clairement définie.

Les hypothèses de transcription fournies par un SRAP sont utilisées dans de nombreuses applications comme la dictée vocale, le sous-titrage, le dialogue homme-machine, la compréhension de la parole, la traduction de la parole, la détection d'entités nommées, le résumé automatique, la recherche d'information, *etc.* Les enjeux sont donc multiples pour la reconnaissance automatique de la parole, puisque le SRAP peut être vu comme un module appartenant à un système plus important.

Dans ce chapitre nous allons présenter les principes de base d'un système de reconnaissance automatique de la parole, en décrivant toutes les étapes nécessaires au système pour passer d'un signal de parole à une transcription. Dans cette optique, nous présentons dans un premier temps l'architecture générale d'un SRAP, puis nous détaillons ses composants et nous explicitons certaines techniques actuelles utilisées pour construire un SRAP à l'état de l'art.

## 2.1 Architecture d'un système de reconnaissance automatique de la parole

Les systèmes de reconnaissance automatique de la parole ont pour objectif d'analyser le signal de la parole et de produire la séquence de mots prononcée. La plupart des SRAP actuels s'appuient sur une approche statistique introduite par [Jelinek, 1976a]. À partir d'un ensemble d'observations acoustiques X = $x_1 x_2 ... x_t$, un SRAP cherche la séquence de mots $W^* = w_1 w_2 ... w_k$, telle que :

$$W^* = \underset{W}{\operatorname{argmax}}\, P(W|X) \tag{2.1}$$

où P(W|X) est la probabilité d'émission de la séquence de mots W sachant X. En appliquant le théorème de Bayes, on obtient :

$$W^* = \underset{W}{\operatorname{argmax}}\, \frac{P(X|W)P(W)}{P(X)} \tag{2.2}$$

Comme la séquence d'observations acoustiques $X$ est fixée, la probabilité $P(X)$ peut être considérée comme une valeur constante quelle que soit la séquence de mots

$W$. Il est alors possible de la retirer de l'équation 2.2. Cette équation devient :

$$W^* = \underset{W}{\operatorname{argmax}}\, P(X|W)P(W) \tag{2.3}$$

La recherche de la séquence de mots la plus probable nécessite l'utilisation de deux types de modèles probabilistes : un modèle acoustique qui permet d'estimer la valeur de P(X|W) et un modèle de langage qui fournit la valeur de P(W). P(X|W) correspond à la probabilité d'observer X lorsque W est prononcé, alors que P(W) est la probabilité que W soit généré dans un langage donné. Pour obtenir un SRAP performant, il est essentiel de définir les modèles les plus pertinents possible pour le calcul de P (W) et P (X|W). La figure 2.1 présente une schématisation générale du fonctionnement d'un SRAP ainsi que ses principaux composants.

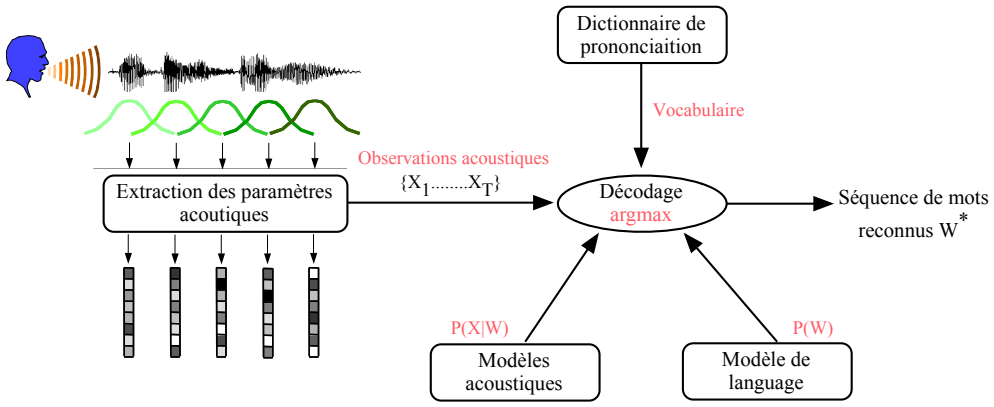



FIGURE 2.1 – Architecture d'un système de reconnaissance de la parole.

## 2.2 Paramétrisation acoustique

Le signal audio contient d'autres informations que la parole. Ces informations ne sont pas nécessaires lors du décodage de la parole. Afin de pouvoir transformer correctement le signal en hypothèses de séquences de mots, il est utile d'extraire les informations caractérisant la parole. Après avoir découpé le signal en trames de 10 ms, la paramétrisation du signal est effectuée, en calculant un vecteur de paramètres pour chaque trame. Celui-ci est généralement calculé sur des fenêtre de taille 25 ms.

Les techniques de paramétrisation les plus utilisées dans la littérature sont l'analyse cepstrale avec la méthode MFCC *"Mel-Frequency Cepstral Coefficients"* [Davis et Mermelstein, 1980], l'analyse paramétrique avec la méthode PLP *"Perceptual Linear Prediction"* [Hermansky et Cox, 1991] et l'analyse temporelle avec la méthode LPCC *"Linear Prediction Ceptral Coefficients"* [Markel et Gray, 1982]. L'ensemble de paramètres obtenu est enrichi par leurs dérivées premières (vitesses, notées $\Delta$) et secondes (accélérations, notées $\Delta\Delta$) afin de prendre en considération

certaines évolutions du signal dans le temps. Après l'extraction des paramètres de chaque trame, on obtient la séquence d'observations acoustiques $X$.

## 2.3 Modélisation acoustique

Le rôle du modèle acoustique est d'estimer la vraisemblance du signal de parole étant donnée une séquence de mots. De nombreuses approches ont été proposées pour la modélisation acoustique, notamment, les Modèles de Markov Cachés (MMC, connus en anglais sous le nom de HMM : *Hidden Markov Models*) [Jelinek, 1976b, Rabiner, 1989a], les réseaux de neurones, etc.

### 2.3.1 Modèles de Markov Cachés

Les Modèles de Markov Cachés sont des automates probabilistes à états finis, utilisés ici pour calculer la probabilité d'émission d'une séquence d'observations acoustiques. Le comportement de ces MMC correspond à l'hypothèse markovienne d'ordre 1, où l'état du modèle à l'instant $t$ dépend uniquement de l'état précédent ($t$-1). C'est à dire que le futur ne dépend que de l'état présent.

En reconnaissance de la parole, l'unité de modélisation utilisée est le phonème : chaque MMC modélise un phonème. Un phonème correspond à la plus petite unité distinctive de la chaine parlée, c'est à dire la plus petite unité de son capable de distinguer un mot d'un autre. La modélisation acoustique d'un mot consiste à considérer l'ensemble des modèles de phonèmes successifs qui le composent. Un même phonème correspond à des réalisations sonores différentes. Ces variations peuvent être dépendantes du locuteur, mais existent aussi pour les réalisations d'un phonème par un même locuteur. Ces variations intra-locuteurs peuvent par exemple dépendre de l'état émotionnel d'un individu, mais surtout du phénomène de co-articulation qui fait que la prononciation d'un phonème dépend des phonèmes qui l'entourent.

Pour prendre en compte cet phénomène de coarticulation dans la modélisation d'un phonème, un MMC est généralement construit pour un phonème donné associé à un contexte gauche et un contexte droit particuliers, qui correspondent respectivement au phonème qui précède et celui qui suit ce phonème. Le triplet composé d'un contexte gauche, d'un phonème et d'un contexte droit est appelé triphone ou phonème en contexte. La position d'un phonème dans un mot est parfois prise en compte pour affiner sa modélisation.

La figure 2.2 présente un exemple de MMC gauche-droit à trois états émetteurs, auxquels sont ajoutés un état de début et un état de fin.

De manière générale, un MMC est composé des paramètres suivants :

— Un ensemble $Q$ de N états : $E = \{E_1, E_2, E_3, ...., E_N\}$, où N est choisi empiriquement.

— d'un ensemble $A$ de probabilités de transition discrètes $a_{ij} = P(E_{t+1} = j | E_t = i)$ pour aller de l'état $i$ à l'état $j$ $(i, j \in Q)$ ;

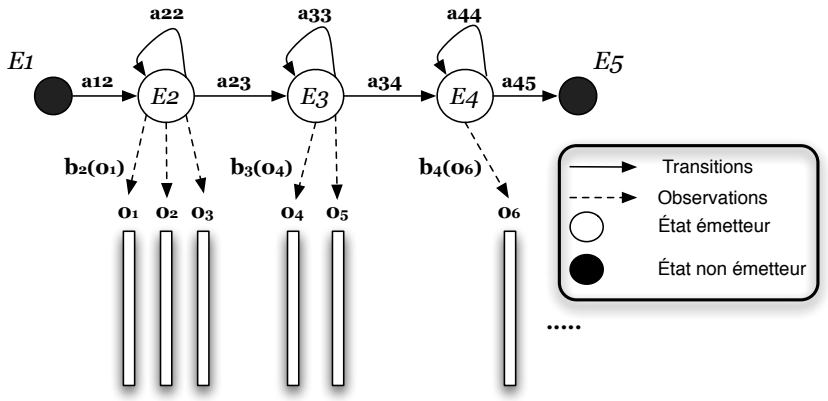



FIGURE 2.2 – Exemple d'un MMC à 5 états, dont 3 émetteurs [Bougares, 2012]

— d'un ensemble $B$ de distributions de probabilités d'émission associées aux états ;

— et d'un ensemble de probabilités initiales $\pi$ de taille $N$, tel que $\forall i, \sum_i^N \pi_i = 1$

Afin de générer la séquence d'observations $o_1 o_2 ... o_T$, un MMC transite d'un état $i$ à un état $j$ (où $j \geqslant i$, étant donné qu'un état peut boucler sur lui même) en suivant les probabilités de transitions $a_{ij}$. À chaque instant $t$, un état $j$ est atteint, générant une émission $o_t$, associée à une densité de probabilité $b_j(o_t)$.

Plusieurs méthodes ont été proposées pour calculer la valeur de $b_j(o_t)$, comme les mélanges de gaussiennes et les réseaux de neurones profonds (DNN, CNN, *etc.*). Ces approches sont détaillées dans les sections suivantes.

#### 2.3.1.1 Les mélanges de modèles gaussiens

Les mélanges de gaussiennes (GMM : *Gaussian Mixture Model*) ont jusqu'à récemment été l'une des méthodes les plus utilisées pour estimer les probabilités d'observation $b_j(o_t)$ associées à chaque état $j$ du MMC. Ce modèle estime des densités de probabilité avec une somme pondérée de $K$ fonctions de densité gaussienne $\mathcal{N}(\mu, \Theta)$, d'espérance $\mu$ et de covariance $\Theta$. La probabilité d'observation est définie comme suit :

$$b_j(o_t) = \sum_{k=1}^{K} c_{jk} \mathcal{N}(o_t, \mu_{jk}, \Theta_{jk}), \quad \sum_{k=1}^{K} c_{jk} = 1, \tag{2.4}$$

Chaque gaussienne ayant une densité de probabilité continue égale à :

$$\mathcal{N}(o_t, \mu_{jk}, \Theta_{jk}) = \frac{1}{\sqrt{(2\pi)^D |\Theta_i|}} e^{\frac{1}{2}(o_t - \mu)^\top \Theta^{-1} (o_t - \mu)} \tag{2.5}$$

où $o_t \in \mathbb{R}^D$ représente un vecteur de paramètres de l'état $j$ de $D$ dimensions, $k$

est l'indice d'une gaussienne et $c_k$ sont les coefficients du mélange.

Dans la littérature, les modèles MMC qui utilisent une distribution multi-gaussienne des probabilités d'émission sont couramment qualifiés de *HMM-GMM* [Rabiner et Juang, 1993].

#### 2.3.1.2 Réseaux de neurones profonds (DNN)

L'utilisation des réseaux de neurones artificiels (ANN : *Artificial Neural Networks*) pour la modélisation acoustique a été proposée il y a trois décennies.

Cependant, à cette époque ni les dispositifs de calcul ni les algorithmes d'apprentissage n'étaient adéquats pour l'apprentissage d'un ANN composé de nombreuses couches cachées sur de grandes quantités de données elles-mêmes très rares. L'apprentissage d'un ANN avec une seule couche cachée n'était pas suffisant pour avoir de meilleurs résultats par rapport aux GMMs.

Au cours de ces dernières années, les avancées dans le domaine de l'apprentissage automatique ainsi que la disponibilité de dispositifs de calcul puissants ont conduit à des méthodes plus efficaces pour l'apprentissage de réseaux de neurones profonds (DNN : *Deep neural networks*). Les auteurs dans [Hinton *et al.*, 2012] ont montré que les DNN obtiennent de meilleurs résultats que les GMMs dans de nombreuses tâches de reconnaissance de la parole.

La figure 2.3 illustre l'architecture d'un modèle acoustique de type HMM/DNN. La dernière couche du DNN utilise la fonction $Softmax$ pour calculer la probabilité de chaque état $j$ du HMM sachant l'observation $o_t$ à l'instant $t$ :

Sur le même principe, d'autres familles de réseaux de neurones peuvent être utilisés pour la modélisation acoustique en association avec un modèle de Markov, comme les réseaux de neurones convolutifs (CNN) présentés en détails en section 1.1.

### 2.3.2 Adaptation acoustique

Les composants d'un SRAP, modèle acoustique et modèle de langage, sont de nature probabiliste. Pour obtenir une bonne estimation des probabilités calculées par les modèles acoustiques, il est nécessaire d'entrainer ces modèles sur suffisamment de données caractéristiques des conditions d'exploitation du SRAP en termes de conditions acoustiques et de variabilité des locuteurs. Or, la construction d'un tel corpus est relativement coûteuse. Les SRAP sont alors en pratique utilisés dans un contexte relativement éloigné de celui des données d'apprentissage des modèles. C'est pourquoi des techniques d'adaptation des modèles acoustiques ont été développées pour rendre les SRAP plus robustes.

L'adaptation consiste à transformer les modèles déjà appris de façon à les rapprocher le plus possible des conditions d'utilisation, en utilisant les modèles initiaux et un ensemble réduit de données d'adaptation. Les techniques d'adaptation de modèles acoustiques HMM-GMM les plus utilisées dans les SRAP se divisent en deux grandes familles. Des méthodes de transformations linéaires comme MLLR (*Maximum Likelihood Linear Regression*) [Leggetter et Woodland, 1995], CMLLR (*Constrai-*

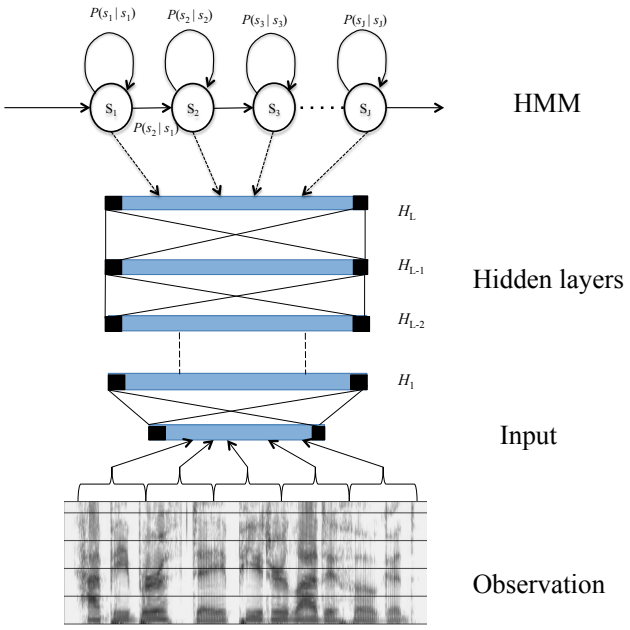



FIGURE 2.3 – Une architecture HMM/DNN pour la modélisation acoustique [Juan et Flora, 2015].

*ned Maximum Likelihood Linear Regression*) [Digalakis et Neumeyer, 1995]. Ils consistent à appliquer une seule fonction de transformation linéaire à l'ensemble des gaussiennes afin de les rapprocher de l'espace de test. Ces approches se différencient par la méthode d'estimation des matrices de transformation et par les composantes gaussiennes sur lesquelles la transformation est appliquée. Par exemple, l'adaptation CMLLR peut être appliqués directement sur les vecteurs de paramètres acoustiques. Ce type d'adaptation est appelée fMLLR (*feature-space Maximum Likelihood Linear Regression*) [Li *et al.*, 2002]. La deuxième famille est fondée sur le maximum *a posteriori* (*Maximum A Posteriori estimation* ou MAP) [Gauvain et Lee, 1994]. Elle consiste à adapter les gaussiennes de tous les états indépendamment les uns des autres.

## 2.4 Modélisation linguistique

Le modèle de langage (ML) représente un composant important dans un système de reconnaissance de la parole. Il permet de restituer des contraintes linguistiques en guidant le décodage acoustique pour choisir la suite de mots la plus probable. Afin de modéliser les contraintes linguistiques d'une langue donnée, les modèles de langages statistiques attribuent une probabilité à chaque séquence de mots $W$ de

longueur $k$, exprimée par :

$$P(W) = P(w_1) \prod_{i=2}^{k} P(w_i|w_1...w_{i-1}) \tag{2.6}$$

où $w_1...w_{i-1}$ représente l'historique du mot $w_i$.

En reconnaissance automatique de la parole, les modèles de langages sont généralement de nature probabiliste. Cependant, les modèles de langages neuronaux émergent ces dernières années.

### 2.4.1 Modèle de langage n-gramme

Le modèle de langage n-gramme permet d'estimer la probabilité d'un mot en fonction de son historique composé des $n-1$ mots que le précèdent, $n$ étant l'ordre du modèle qui est généralement égal à trois ou quatre. Dans ce cas, pour un ordre supérieur à 1, l'équation 2.6 peut s'écrire :

$$P(W) = P(w_1) \prod_{i=2}^{k} P(w_i|w_{i-n+1}...w_{i-1}) \tag{2.7}$$

où $P(w_1)$ est la probabilité d'observer le mot $w_1$ et $P(w_i|w_{i-n+1}...w_{i-1})$ est la probabilité du mot $w_i$ sachant son historique $w_{i-n+1}...w_{i-1}$.

La probabilité $P(w_i|w_{i-n+1}...w_{i-1})$ est calculée avec la méthode du *maximum de vraisemblance*, dont le nom indique que la distribution des probabilités du ML est celle qui maximise la vraisemblance du corpus d'apprentissage :

$$P(w_i|h) = \frac{C(h, w_i)}{C(h)},\ h = w_{i-n+1}...w_{i-1} \tag{2.8}$$

où C le nombre d'occurences d'une séquence de mots dans le corpus d'apprentissage.

### 2.4.2 Techniques de lissage

La performance du modèle de langage dépend de la taille du corpus d'apprentissage et de la proximité linguistique du domaine d'application du système. Plus le corpus d'apprentissage est grand, plus l'estimation des probabilités linguistiques est précise. Par contre, quelle que soit la taille du corpus d'apprentissage, il y a souvent des n-grammes non vus dans le corpus, pour lesquelles le modèle attribuera une probabilité nulle. Pour compenser ce manque, des techniques de lissage sont mis en place. Elles permettent de généraliser le modèle en attribuant une probabilité non nulle aux n-grammes non vus dans le corpus d'apprentissage. Ces techniques consistent à changer la distribution de la masse de probabilités, en retirant une certaine masse aux n-grammes observés pour l'attribuer aux n-grammes non vus. Plusieurs techniques de lissage ont été proposées dans la littérature, qui sont décrites dans [Chen et Goodman, 1996]. Parmi ces techniques, le

lissage Kneser-Ney [Kneser et Ney, 1995], *etc*, utilisées avec la technique de repli (ou *back-off*) [Katz, 1987].

### 2.4.3 Adaptation linguistique

Les modèles de langage probabilistes sont étroitement liés aux données sur lesquelles a été réalisé leur apprentissage. Cela signifie que leur capacité de généralisation est réduite en dehors du domaine sur lequel ils ont été entraînés. Par exemple, un modèle appris sur un corpus journalistique comme celui du journal Le Monde peut être utilisé pour la transcription automatique des journaux télévisés, même s'il est préférable de disposer de transcriptions manuelles de ce type de journaux. Par contre, ce modèle sera totalement inadapté pour la reconnaissance d'un dialogue oral spontané. En effet, un modèle de langage spécifique à un domaine donné est plus efficace qu'un modèle général même s'il est appris sur une vaste collection de texte. De même, il est intéressant de modifier le modèle en fonction de l'application visée. Comme la construction d'un corpus spécifique à chaque situation est très coûteuse et n'est pas toujours possible, la solution est d'adapter un modèle de langage général sur un domaine spécifique. L'adaptation d'un modèle de langage à un domaine consiste à réestimer ses probabilités n-grammes de manière à prendre en compte les spécificités linguistiques du domaine considéré. L'une des méthodes d'adaptation proposée dans la littérature est l'interpolation linéaire d'un modèle de langage général avec un second appris sur un corpus plus petit spécifique à la tâche [Woodland *et al.*, 1999]. Le schéma général du processus d'adaptation est présenté dans la figure 2.4 [Estève, 2002]. On peut également enrichir le vocabulaire du système de reconnaissance avec des mots liés au domaine afin d'éviter les mots hors vocabulaire.

### 2.4.4 Les modèles de langages neuronaux

Souvent, l'ordre retenu pour un modèle de langage n-gramme est égale à 4. Cela est dû au caractère épars des langages naturels, *i.e,* il est impossible d'observer tous les n-grammes et la plupart des n-grammes apparaissent rarement. De plus, selon [Bengio *et al.*, 2003] l'augmentation de l'ordre du modèle de langage n-grammes va augmenter de manière exponentielle le nombre de ses paramètres.

L'apparition des modèles de langages neuronaux a ouvert de nombreuses perspectives. Le principe de base de ces modèles est de projeter les mots du contexte ($n-1$ mots précédents) dans un espace continu qui permet d'exploiter la notion de similarité entre les mots. Les vecteurs résultants correspondent aux représentations continues des mots, connus aussi sous le terme de *word embeddings*. L'utilisation de ces représentations continues confère aux modèles neuronaux une meilleure capacité de généralisation et leur permet de mieux gérer le problème des n-grammes absents dans le corpus d'apprentissage. D'après [Bengio *et al.*, 2003], les modèles de langage neuronaux se différencient des modèles de langage n-grammes par leur capacité à prendre en compte un contexte large, puisque le nombre de paramètres augmente

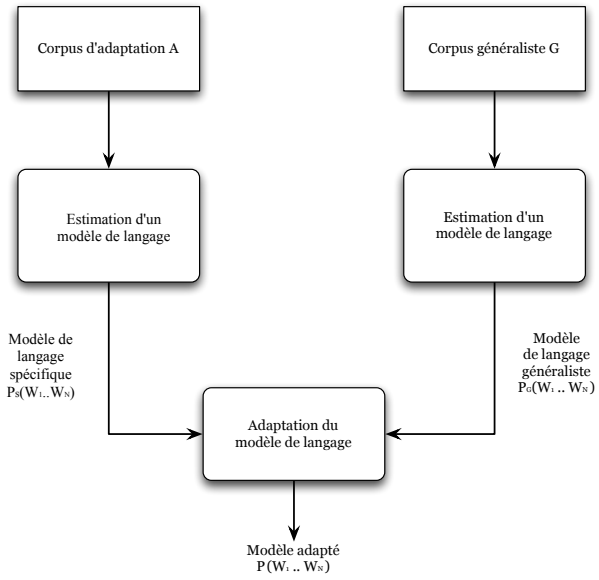



FIGURE 2.4 – Schéma général du processus d'adaptation d'un modèle de langage.

linéairement avec l'ordre du modèle.

Les modèles de langages neuronaux s'appuient sur différentes architectures : les réseaux de neurones *feedforward* [Bengio *et al.*, 2003, Schwenk, 2007] ou les réseaux de neurones récurrents [Mikolov *et al.*, 2010, Mikolov *et al.*, 2011a]. Une description du principe de base d'un modèle de langage neuronal a déjà été présentée dans la section 1.2.3.1 (chapitre 1).

### 2.4.5 Évaluation du modèle de langage

La qualité d'un modèle de langage dépend de sa capacité à désambiguiser des hypothèses de transcription proches acoustiquement à l'aide du contexte lexical. La métrique la plus couramment utilisée pour mesurer la qualité d'un modèle de langage est la perplexité [Jelinek *et al.*, 1977]. Celle-ci mesure le degré de déterminisme d'un modèle de langage. Plus la valeur de perplexité est faible, plus le modèle est pertinent. La perplexité (PPL) est définie par :

$$PPL = 2^{-\frac{1}{n}\sum_{i=1}^{n} log_2 P(w_i|h)} \tag{2.9}$$

où $P(w_i|h)$ est la probabilité proposée par le modèle de langage pour le mot $w_i$ sachant l'historique $h$ et $n$ le nombre de mots dans corpus.

## 2.5 Dictionnaire de prononciation

Le dictionnaire de prononciation (nommé également dictionnaire de phonétisation ou lexique) est nécessaire pour associer à chaque mot sa description en terme d'unités acoustiques (phonèmes), en tenant compte du fait qu'il peut exister plusieurs prononciations possibles pour un même mot.

Le jeu de phonèmes utilisé pour la prononciation varie d'une langue à une autre. Il dépend aussi des choix des concepteurs du système de reconnaissance.

La création du dictionnaire de prononciation peut se faire avec plusieurs approches. L'approche la plus efficace consiste à créer le lexique manuellement.Les prononciations ainsi produites sont considérées comme fiables, étant donné qu'elles sont vérifiées par un expert humain. Cependant, cette approche est très coûteuse en temps et en ressources, et il est difficile de couvrir la totalité des mots d'une langue et de couvrir l'ensemble des prononciations possibles pour chaque mot.

Une autre approche souvent utilisée est la phonétisation automatique des mots. Au lieu de phonétiser l'ensemble des mots manuellement, il est possible d'appliquer des règles de phonétisation [Bechet, 2001] pour transformer automatiquement les graphèmes[1] en phonèmes. Cette méthode est utilisée parfois comme complément à l'approche manuelle, en enrichissant le lexique phonétique de référence des mots manquants.

Décrire un ensemble de règles de phonétisation est également difficile, dans la mesure où elles doivent couvrir toutes les spécificités pouvant être rencontrées et ne pas entrer en conflit mutuellement. Des méthodes probabilistes, fondées sur une modélisation de la conversion graphèmes vers phonèmes d'après un lexique existant, sont aussi utilisées. Nous pouvons par exemple citer les modèles de séquences jointes [Bisani et Ney, 2008] ou plus récemment les modèles neuronaux [Yao et Zweig, 2015, Rao *et al.*, 2015].

Le dictionnaire de prononciation recense alors l'ensemble des mots reconnaissable par le SRAP. En effet, tout mot absent du lexique ne peut être reconnu par le SRAP et peut engendrer des erreurs sur son voisinage. Le même problème est également observé si la phonétisation est erronée. De plus, la taille du lexique est conditionnée par la tâche visée.

## 2.6 Évaluation d'un système de reconnaissance de la parole

Plusieurs campagnes d'évaluations ont été proposées afin de pouvoir évaluer les différents systèmes de reconnaissance de la parole (par exemple NIST). Ces évaluations se font d'une manière rigoureuse, en évaluant les SRAP sur les mêmes données de test. La figure 2.5 montre l'historique de l'évaluation des systèmes de reconnaissance automatique de la parole (fondés sur l'approche HMM-GMM) sur les tâches

1. Un graphème se définit comme l'écriture associée à un phonème donné. Ce graphème peut être constitué d'une ou plusieurs lettres.

les plus difficiles jusqu'à NIST 2009 [Fiscus et Ajot, 2009].

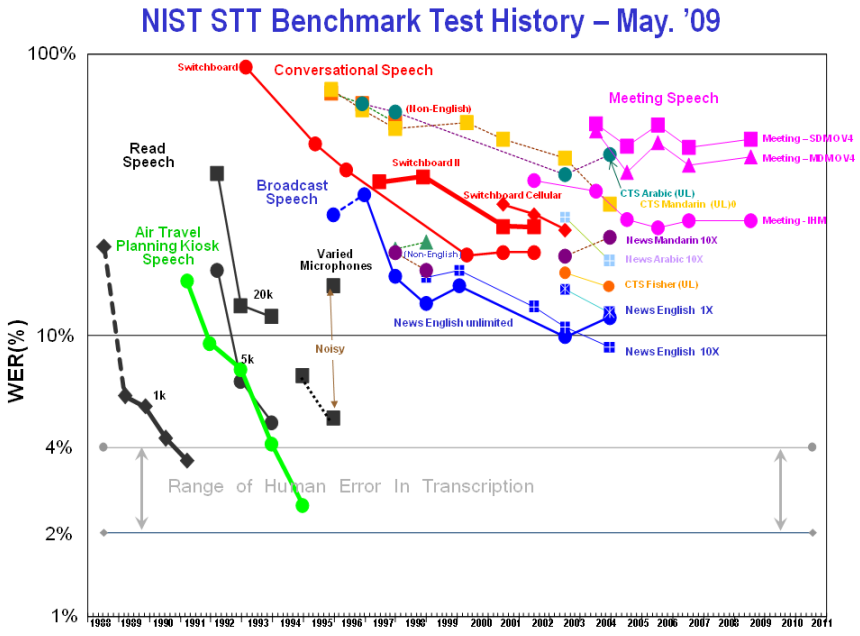



FIGURE 2.5 – Historique d'évaluation des systèmes de reconnaissance de la parole NIST 2009.

Les SRAP sont généralement évalués en alignant les transcriptions de la référence avec les transcriptions automatiques et en dénombrant les trois types d'erreurs suivantes :

— Insertion (I) : mot reconnu inséré par rapport à la référence.

— Substitution (S) : mot reconnu à la place d'un mot dans la référence.

— Supression (D) : mot de la référence oublié dans la transcription automatique.

Le taux d'erreurs sur les mots ($WER$), métrique couramment utilisée, mesure le rapport du nombre d'erreurs de reconnaissance sur le nombre total de mots prononcés, par la formule suivante :

$$WER = \frac{I + S + D}{Nombre\ de\ mots\ dans\ la\ référence} \tag{2.10}$$

## 2.7 Conclusion

Dans ce chapitre, nous avons introduit les fondements théoriques d'un système de reconnaissance automatique de la parole. Nous avons décrit ensuite ses différents composants : le modèle acoustique, le modèle de langage et le dictionnaire de prononciation.

L'introduction d'algorithmes d'apprentissage profond a amélioré les performances de SRAP, surtout par l'amélioration des modèles acoustiques. Plu-

sieurs architectures neuronales ont été utilisées pour remplacer les GMMs, notamment les DNNs [Hinton *et al.*, 2012], les CNNs [Abdel-Hamid *et al.*, 2012, Sainath *et al.*, 2013], et les réseaux de neurones récurrents de type LSTM [Sak *et al.*, 2014]. Des travaux récents [Hannun *et al.*, 2014, Graves et Jaitly, 2014, Amodei *et al.*, 2015], non présentés ici, proposent des systèmes de reconnaissance de la parole entièrement neuronaux. Ces systèmes transforment un signal de la parole en texte, sans avoir besoin d'une représentation phonétique intermédiaire.

En pratique, les systèmes de reconnaissance de la parole sont encore très sensibles à la variabilité : variabilité de l'environnement acoustique, du locuteur, du style de langage, de la thématique du discours, *etc.* Cette sensibilité, principalement liée à la nature statistique des systèmes de reconnaissance de la parole, freine le développement de nouvelles applications : les systèmes de reconnaissance génèrent des erreurs qui nuisent à certaines exploitations de leur sortie.

La détection des erreurs dans les transcriptions automatiques a pour but d'atténuer l'impact de ces erreurs sur toute application exploitant des transcriptions automatiques. Nous présentons dans le chapitre suivant un état de l'art sur les méthodes de détection d'erreurs dans les transcriptions automatiques de la parole.

CHAPITRE 3

# DÉTECTION AUTOMATIQUE DES ERREURS DANS LES TRANSCRIPTIONS AUTOMATIQUE DE LA PAROLE

**Sommaire**

Les avancés dans le domaine du traitement de la parole ainsi que la disponibilité des dispositifs de calcul puissants ont conduit à de meilleures performances dans le domaine de la reconnaissance de la parole. Cependant, malgré ces performances, les systèmes de reconnaissance automatique de la parole (SRAP) génèrent encore des erreurs. Cela s'explique par leur sensibilité à la variabilité : variabilité de l'environnement acoustique, du locuteur, du style de langage, de la thématique du discours, *etc.* Ces erreurs constituent un obstacle à l'application de certains traitements automatiques tels que l'extraction d'information, la traduction de la parole, la compréhension de la parole, *etc.*

L'exploitation des transcriptions automatiques peut devenir plus efficace si l'on est capable de détecter et/ou de corriger les erreurs contenues dans les transcriptions automatiques. La détection d'erreurs revient à estimer la fiabilité d'une hypothèse en utilisant des mesures de confiance. Cependant, la détection des erreurs n'est pas une tâche facile compte tenu du fait qu'il existe plusieurs types d'erreurs tant au niveau lexical que syntaxique. Ces erreurs peuvent aller d'une simple erreur en accord ou en genre à l'insertion d'un mot non pertinent pour la compréhension globale de la séquence de mots. Elles peuvent aussi se répercuter sur les mots voisins et créer une zone de mots erronés.

Dans ce chapitre, nous présentons d'abord la définition d'une mesure de confiance. Nous en détaillons ensuite les différentes catégories. Enfin, nous présentons un état de l'art sur les méthodes de détection d'erreurs dans les transcriptions automatiques de la parole.

## 3.1 Mesures de confiance

### 3.1.1 Intérêts

Les mesures de confiance (MC) sont utilisées pour évaluer la fiabilité des résultats de reconnaissance. Une MC donne un aspect qualitatif de la sortie du SRAP, en associant à chaque mot $w$ un score compris entre 0 et 1. Elle peut être interprétée comme étant la probabilité qu'un mot $w$ soit correct. Plus $MC(w)$ est proche de 1 plus le mot est considéré comme correct, et inversement.

Une mesure de confiance idéale doit satisfaire les propriétés suivantes :

— Elle doit être égale à 1 si le mot $w$ est correct et égale à 0 si $w$ est erroné.

— La moyenne $\mu(MC)$ des mesures de confiance de $K$ mots ($K = w_1, ..., w_k$) doit être une approximation du taux de mots émis bien reconnus, définie par la formule suivante :

$$\mu(MC) = \frac{1}{K} \sum_{i=1}^{k} MC(w_i) \tag{3.1}$$

Les mesures de confiance peuvent être calculées selon plusieurs niveaux de granularité : au niveau du phonème (généralement pour la reconnaissance de la parole),

au niveau du mot, de la phrase entière ou encore au niveau du concept (notamment utile en dialogue), cela dépend du besoin et de la tâche visée.

Les mesures de confiance ont tout d'abord été introduites pour la détection des mots hors vocabulaire par [Asadi *et al.*, 1990]. Ensuite, les auteurs dans [Young et Woodland, 1994] ont introduit l'utilisation des probabilités *a posteriori* comme mesure de confiance pour la reconnaissance automatique de la parole (RAP), en exploitant les travaux de [Asadi *et al.*, 1990]. Ces applications ont fourni la motivation initiale pour le développement d'autres mesures de confiance. Une liste des différentes mesures de confiance est présentée dans [Chase, 1997].

Les mesures de confiance sont utiles dans de nombreuses applications [Jean-Paul Haton, 2016]. Elles sont utilisées dans les systèmes de dialogue (réservation de billets de train par exemple) [Hazen *et al.*, 2002], en développant des stratégies de dialogue différentes selon la confiance qu'ils ont en l'exactitude du résultat de la reconnaissance. Elles peuvent être utilisées pour filtrer les mots peu fiables pour améliorer la qualité des modèles acoustiques, quand ils sont adaptés d'une manière non supervisée [Pitz *et al.*, 2000]. De même, elles sont utilisées pour augmenter de manière non supervisée les données d'apprentissage des modèles acoustiques [Mauclair, 2006]. Les mesures de confiance peuvent être également utilisées directement afin d'améliorer l'algorithme de décodage. Elles sont calculées en fonction des probabilités acoustiques et linguistiques des mots présents dans le graphe de mots [Wessel *et al.*, 2001].

Dans [Jiang, 2005], les auteurs présentent différentes utilisations et avancées des mesures de confiance en RAP. Ils proposent de les regrouper en trois catégories : les mesures de confiance fondées sur le rapport de vraisemblance [Rose *et al.*, 1995, Weintraub *et al.*, 1997], sur la probabilité *a posteriori* [Kemp *et al.*, 1997, Wessel *et al.*, 2001], ou sur la combinaison de descripteurs [Wessel *et al.*, 1999, Zhang et Rudnicky, 2001].

### 3.1.2 Rapport de vraisemblance

Un mot hypothèse de reconnaissance $w$ correspond à une portion de parole, c'est-à-dire une séquence $X$ d'observations acoustiques. L'estimation de mesures de confiance peut être formulée comme un test statistique dont le but est de vérifier si cette hypothèse est correcte ou non. L'hypothèse doit alors être rangée dans l'une des deux classes *correct* ou *erreur* :

— $H_{cor}(w)$ : le mot hypothèse $w$ porté par $X$ est correct,

— $H_{inc}(w)$ : le mot hypothèse $w$ est incorrect et ne correspond pas à ce qui est porté par $X$.

D'après Neyman-Pearson [Neyman et Pearson, 1992], le test du rapport de vraisemblance (*Likelihood Ratio* - LR) est le test le plus adéquat permettant de rejeter une hypothèse par rapport à une autre. Le rapport de vraisemblance LR(X|w) peut être utilisé directement comme une mesure de confiance [Lleida et Rose, 1996] at-

tribué à l'hypothèse de reconnaissance $w$, définie par l'équation 3.2 :

$$LR(X|w) = \frac{P(X|H_{cor}(w))}{P(X|H_{inc}(w))} \tag{3.2}$$

L'estimation du score de vraisemblance nécessite de trouver un moyen pour modéliser $H_{cor}$ et $H_{inc}$ afin de calculer les probabilités $P(X|H_{cor}(w))$ et $P(X|H_{inc}(w))$. Le principal défi réside dans l'estimation de $H_{inc}$ qui regroupe toutes les hypothèses alternatives erronées. La modélisation de $H_{cor}$ et $H_{inc}$ peut se faire avec des HMMs. En effet, $H_{cor}$ peut être estimé avec le score acoustique de $w$ fourni lors du décodage de $X$ par le SRAP. Alors que l'estimation de $H_{inc}$ nécessite de calculer des scores alternatifs en construisant des anti-modèles ou des modèles de rejet. Une description détaillée de ces modèles est donnée dans [Lleida et Rose, 1996, Charlet *et al.*, 2001, Falavigna *et al.*, 2002, Jiang, 2005, Mauclair, 2006, Seigel, 2013].

### 3.1.3 Probabilité *a posteriori*

En calculant la séquence de mots la plus vraisemblable pour proposer une hypothèse de reconnaissance, un SRAP collecte des informations informations qui peuvent être utiles pour calculer la probabilité *a posteriori* (PAP) d'un mot hypothèse. Celle-ci donne une estimation de fiabilité sur le fait que $w$ correspond à l'information lexicale portée par les observations acoustiques $X$ [Moreau *et al.*, 2000].

Dans le chapitre 2 présentant la reconnaissance de la parole, l'équation 2.2 montre que, lors de l'attribution du score $P(w).P(X|w)$ au mot hypothèse $w$, la normalisation par la probabilité $P(X)$ est omise. En effet, on considère que $P(X)$ est constante quelle que soit l'hypothèse de transcription. Ceci explique pourquoi le score $P(w).P(X|w)$ calculé par le SRAP est inadéquat comme mesure de confiance afin de juger de la fiabilité de la reconnaissance. Avec ce score on peut connaitre l'hypothèse la plus vraisemblable du point de vue des connaissances du SRAP, mais on ne sait pas à quel point elle est correcte. Cependant, après avoir normalisé par $P(X)$, la probabilité *a posteriori* peut servir de véritable mesure de confiance, car elle représente une mesure quantitative de l'association entre $X$ et $w$. En théorie, $P(X)$ est calculée par l'équation 3.3 :

$$P(X) = \sum_{hyp} P(hyp).P(X|hyp) \tag{3.3}$$

où toutes les hypothèses de reconnaissance *hyp* sont sommées pour l'ensemble des observations acoustiques X. Cependant, l'énumération de toutes les hypothèses possibles n'est pas réalisable, et donc il est difficile d'estimer P(X) d'une manière précise. En pratique, des techniques d'approximation sont employées pour estimer P(X) lors du calcul de la probabilité *a posteriori*. Les techniques les plus courantes sont fondées sur les graphes de mots [Kemp *et al.*, 1997], les listes de N-meilleures hypothèses [Stolcke *et al.*, 1997] ou les réseaux de confusion [Mangu *et al.*, 2000].

En effet, la probabilité *a posteriori* d'un mot est approchée par le rapport entre

la probabilité *a priori* d'un mot et la somme des probabilités *a priori* de toutes les autres hypothèses alternatives. Ces probabilités *a priori* correspondent à la combinaison des scores fournis par le modèle acoustique et le modèle de langage.

La probabilité *a posteriori* peut être calculée de trois manières différentes :

— Pour un graphe de mots, la probabilité *a posteriori* d'un arc est le ratio de la somme des probabilités de tous les chemins passant par l'arc auquel est associé le mot sur la somme des probabilités de tous les chemins composant le graphe. Cette probabilité peut être utilisée comme une mesure de confiance du mot supporté par l'arc, mais elle risque de ne pas être optimale. En effet, comme le mot peut être porté par plusieurs arcs différents, la prise en compte d'un seul arc pour le calcul de sa PAP va sous-estimer son score de confiance, car la probabilité totale pour un mot est répartie entre tous les arcs associés à ce mot.

— Pour une liste de N-meilleures hypothèses, la probabilité *a posteriori* d'un mot est le ratio entre la somme des probabilités *a priori* des occurrences de ce mot à une position donnée parmi les N hypothèses et la somme de toutes les probabilités *a priori* des mots situés à la même position.

— Enfin, dans un réseau de confusion, la probabilité *a posteriori* d'un mot est le rapport entre la somme des probabilités *a priori* des transitions dans la même cohorte portant ce mot et la somme des probabilités *a priori* des mots en concurrence avec ce mot y compris lui même. Une cohorte (ou *bin* en anglais) d'un réseau de confusion correspond à l'ensemble des mots concurrents entre deux noeuds du réseau de confusion.

Une description détaillée de ces trois techniques est donnée dans [Mauclair, 2006].

### 3.1.4 Approche par classification

La mesure de confiance peut être vue comme un classifieur statistique permettant d'estimer la probabilité qu'un mot de la transcription soit correctement transcrit [Jiang, 2005]. Le but du classifieur est d'attribuer une étiquette *correct* ou *erreur* pour chaque mot. L'estimation de la confiance d'une transcription dépend de plusieurs facteurs :

— Tout d'abord, le choix de la granularité de l'erreur à détecter [Hazen *et al.*, 2002] (mot, syntagme ou phrase entière).

— Puis, la définition de l'ensemble des descripteurs et mesures de confiance qui seront intégrés dans l'estimation de confiance, qui peuvent être issus du décodage ou collectés à partir d'autres sources d'information.

— Enfin, la combinaison des différents descripteurs et/ou mesures de confiance ainsi que le développement d'une stratégie de décision prenant en compte l'ensemble d'entre eux [Sarikaya *et al.*, 2005].

#### 3.1.4.1 Ensemble de descripteurs

Plusieurs descripteurs ont été proposés dans la littérature, dont certains sont issus du système de transcription de la parole. Certaines classes de descripteurs les plus communs, ainsi que des exemples de chaque classe, sont décrits ci-dessous.

**Descripteurs fondés sur le graphe de mots (ou treillis) :** la *densité* de l'hypothèse [Kemp *et al.*, 1997] correspond au nombre d'arcs en concurrence couvrant le segment temporel d'un mot dans la transcription la plus probable. Cette mesure peut donner une idée de l'incertitude du décodeur à choisir ce mot. La *stabilité* du mot dans un treillis [Sanchis *et al.*, 2003] est une mesure fondée sur le principe qu'un mot est plus susceptible d'être correct s'il se trouve en concurrence avec un certain nombre de mots couvrant un intervalle de temps similaire.

**Descripteurs fondés sur les informations acoustiques :** le *score de vraisemblance acoustique du mot* normalisé par le nombre de trames ou par le nombre de phonèmes est une estimation de la correspondance entre le signal et le mot. La *stabilité acoustique* est le nombre d'hypothèses alternatives qui sont générées en fonction des différents poids du modèle de langage au moment du décodage. Ainsi, la stabilité acoustique d'un mot donné est définie comme le nombre de fois où le mot apparaît à la même place dans les hypothèses alternatives divisé par le nombre total d'hypothèses. Le principe de cette mesure consiste en ce que si plusieurs hypothèses sont produites avec des poids de modèles de langage différents (*Grammar Scaling Factor*), un mot sera vraisemblablement correct s'il apparaît à la même position dans la majorité des hypothèses [Jiang, 2005].

**Descripteurs fondés sur les informations linguistiques :** *la probabilité du modèle de langage* attribuée à un mot ou à une séquence de mots ou encore le *comportement de repli* sont également utilisés comme mesure de confiance. En effet, le modèle de langage est moins confiant quand il attribue pour certains mots une probabilité issue d'un ordre inférieur [Uhrik et Ward, 1997].

**Descripteurs extraits d'une liste de N-meilleures hypothèses :** à partir de cette liste on peut calculer le *nombre d'occurrences* d'un mot dans les différentes hypothèses, la *différence entre la meilleure hypothèse et les suivantes*, le *score* des premières hypothèses [Hazen *et al.*, 2002], *etc.*

**Descripteurs prosodiques :** *l'énergie*, la *fréquence fondamentale F0 (le maximum, le minimum, la moyenne)*, le *débit de parole* [Stoyanchev *et al.*, 2012a, Hirschberg *et al.*, 2004] peuvent être utilisés. Des descripteurs fondés sur le *nombre de phonèmes* dans le mot ainsi que la *durée* de chaque phonème le sont également. Ces descripteurs reflètent la performance du modèle acoustique sur certaines régions du signal audio [Weintraub *et al.*, 1997].

**Descripteurs collectés à partir d'autres sources d'informations :** ils peuvent apporter des connaissances de plus haut niveau aux mesures de confiance telles que des connaissances sémantiques comme

l'analyse sémantique latente ou l'information mutuelle inter-mots [Cox et Dasmahapatra, 2002, Mauclair, 2006], des connaissances lexicales (*longueur* et *position* du mot dans la transcription), syntaxiques (*l'étiquette syntaxique* du mot, *regroupement en syntagmes*), contextuelles [Béchet et Favre, 2013, Dufour *et al.*, 2012, Parada *et al.*, 2010], *etc.*

#### 3.1.4.2 Classifieurs

Un descripteur idéal devrait fournir des informations utiles permettant de discriminer les mots correctement reconnus d'autres non reconnus. Toutefois, aucun des descripteurs ci-dessus n'est idéal en ce sens. Par conséquent, plusieurs travaux de recherche se sont focalisés sur la combinaison de ces différents descripteurs pour obtenir de meilleures performances. Ainsi, les auteurs de [Cox et Dasmahapatra, 2002, Thomas et Trancoso, 2010] ont montré que la combinaison des descripteurs issus du système de reconnaissance de la parole avec ceux collectés d'autres sources d'informations améliore significativement les résultats.

Supposons que chaque mot $w$ est associé à un ensemble de descripteurs et mesures de confiance. Leur combinaison pourrait permettre de tirer parti de la qualité de chacun d'entre eux et ainsi atteindre de meilleures performances. Cette combinaison peut être obtenue par des opérations mathématiques (minimum, maximum, moyenne arithmétique, moyenne géométrique, produit ou encore la moyenne quadratique [Mauclair, 2006]) dans le cas des mesures de confiance ou des descripteurs numériques, ou à l'aide de classifieurs. Les approches de combinaison s'appuient sur les différents descripteurs et mesures de confiance afin de calculer une nouvelle mesure de confiance performante, permettant de séparer efficacement les hypothèses correctes des hypothèses erronées. De nombreux classifieurs sont proposés dans la littérature pour l'estimation des mesures de confiance.

##### Interpolation linéaire

La combinaison des mesures de confiance peut se faire par une simple interpolation linéaire, afin de prendre en compte les qualités de chacune d'elle. Cette interpolation est définie par l'équation suivante :

$$MC(w) = \sum_{i=1}^{k} q_k MC_k(w), \ avec \ \sum_{k=1}^{k} q_k = 1 \tag{3.4}$$

où $k$ est le nombre de mesures de confiance associées au mot $w$ ; $q_k$ est un poids attribué à chaque $MC$ qui peut être appris de manière à minimiser le taux d'erreur sur un corpus de développement. Ici, la classification des mots en *correct* ou *erreur* se fait on appliquant un seuil sur la mesure de confiance résultante. Les auteurs de [Guo *et al.*, 2004] combinent un ensemble de mesures de confiance indépendantes, notamment l'information mutuelle inter-mots et la probabilité *a posteriori*. L'interpolation linéaire améliore le taux d'erreurs de 10% par rapport à l'utilisation de chacune de ces deux mesures séparément.

### Machines à vecteurs de support

Utilisées dans [Zhang et Rudnicky, 2001, Hillard et Ostendorf, 2006], les machines à vecteurs supports (*Support Vector Machine* - SVM) sont une méthode de classification binaire introduite par [Vapnik et Kotz, 1982]. Cette méthode repose sur l'existence d'un classifieur linéaire appelé hyperplan. Le but du SVM est de trouver un hyperplan qui va séparer deux nuages de points (représentant les deux classes *correct* ou *erreur*) et maximiser la marge entre ces deux classes. La marge est la distance du point le plus proche à l'hyperplan.

### Arbres de décision

L'arbre de décision [Olshen *et al.*, 1984, Cornuéjols et Miclet, 2011] est une méthode de classification binaire qui consiste à partitionner les données en sous-ensembles les plus purs possible, *i.e.* contenant des objets d'une seule classe. L'arbre modélise une hiérarchie de tests sur les valeurs d'un ensemble de descripteurs. Il décide une réponse (par exemple classification dans l'ensemble {*correct*, *erreur*}) en fonction des valeurs de descripteurs.

L'arbre est appris sur un corpus de mots étiquetés en *correct* ou *erreur*. Son apprentissage consiste à optimiser un critère de test sur les descripteurs représentant le degré de confiance d'un mot. À chaque nœud, l'algorithme de construction de l'arbre détermine le meilleur descripteur à tester en optimisant localement un critère de test (un critère de pureté). Il s'agit d'évaluer localement quel descripteur apporte le plus d'information. Ceci permet de progresser dans l'arbre et atteindre la meilleure répartition possible entre les classes *correct* ou *erreur*. Chaque feuille dans l'arbre correspond à une mesure de confiance calculée à la fin de l'apprentissage en fonction des valeurs des descripteurs.

La performance de l'arbre doit être évaluée avec la même métrique utilisée au moment de sa construction. En effet, les auteurs dans [Zhang et Rudnicky, 2001] montrent qu'il faut avoir une corrélation entre la métrique d'évaluation des performances d'un arbre et le critère de test optimisé au moment de sa construction. Ils ont trouvé que, la performance en terme de taux d'erreurs d'un arbre appris sur un critère de test portant sur la minimisation du taux d'erreur, est meilleure qu'un arbre appris sur un critère de test fondé sur l'entropie.

### Boosting

Les méthodes de boosting [Freund *et al.*, 1996] représentent une famille d'algorithmes d'apprentissage automatique qui consistent à combiner des règles de classification peu performantes, dites *faibles*, en une seule règle de classification (très) performante. Ces méthodes sont par exemple utilisées en reconnaissance de la parole afin d'effectuer une séparation binaire *correct* ou *erreur* des hypothèses [Moreno *et al.*, 2001] et aussi pour la détection des mots hors vocabulaire dans les hypothèses de reconnaissance [Lecouteux *et al.*, 2009].

Le boosting consiste à appliquer une procédure de classification itérative à un jeu de données pondérées, où les poids changent au moment de l'apprentissage. En effet, à chaque itération, un classifieur (*apprenant faible*) est appris et met à jour les poids : les poids des données mal classées augmentent, alors que ceux des données correctement classées diminuent. L'objectif est de focaliser l'attention sur les données mal classées dans les itérations suivantes. Finalement, l'algorithme de boosting combine (typiquement par un vote pondéré) l'ensemble des règles de classifications *faibles* pour obtenir une seule règle dite *finale* ou *combinée* .

### Discussion

Certains travaux de recherche se sont intéressés à la comparaison des méthodes de classification décrites ci-dessus. Les auteurs dans [Moreno *et al.*, 2001] ont constaté que le boosting améliore les résultats de classification par rapport à l'utilisation des SVM et des arbres de décision. Dans [Zhang et Rudnicky, 2001], les auteurs ont trouvé que l'utilisation des SVM pour la combinaison des descripteurs améliore les résultats trouvés avec un arbre de décision ou un réseau de neurones composé d'une seule couche cachée. Les auteurs dans [Zhang et Rudnicky, 2001, Stemmer *et al.*, 2002] ont trouvé que ces réseaux de neurones simples ont de meilleures performances par rapport aux méthodes de combinaison par interpolation linéaire et arbres de décision.

Au cours des dernières années, l'avancement dans le domaine de l'apprentissage automatique ainsi que la disponibilité des dispositifs de calcul puissants ont conduit à des méthodes plus efficaces pour l'apprentissage de réseaux de neurones profonds (Deep neural network - DNN). Les auteurs de [Jalalvand et Falavigna, 2015] ont montré la performance des DNN dans la tâche de détection d'erreurs par rapport à d'autres approches, notamment les SVM et les arbres de décision.

Les méthodes de classification décrites précédemment ne prennent que des décisions locales et ne sont pas capables de faire une prédiction optimale sur une séquence. Les champs conditionnels aléatoires (*Conditional Random Fields* - CRF) [Lafferty *et al.*, 2001] sont eux en mesure de mieux saisir les informations globales.

Dans ces travaux de thèse, nous nous sommes intéressés à deux méthodes de classification qui sont : d'une part les CRF qui constituent l'approche à l'état de l'art pour cette tâche ; d'autre part, les réseaux de neurones profonds, qui ont donné de très bons résultats dans le cadre de la modélisation acoustique [Hinton *et al.*, 2012] et pour le traitement automatique du langage écrit [Collobert *et al.*, 2011]. Les performances de notre approche basée sur les réseaux de neurones seront comparées à l'approche état de l'art fondée sur les CRFs.

## 3.1.5 Évaluation des mesures de confiance

Dans la littérature, plusieurs métriques d'évaluation ont été développées afin de comparer et déterminer l'efficacité des mesures de confiance [Mauclair, 2006].
L'entropie croisée normalisée (NCE : Normalized Cross Entropy)

[Stemmer *et al.*, 2002], est parmi les métriques d'évaluation les plus utilisées. Cette métrique est employée lors des évaluations NIST pour évaluer la qualité des mesures de confiance des systèmes de reconnaissance de la parole. Elle représente une estimation de l'information additionnelle que la mesure de confiance peut apporter à l'hypothèse fournie par le SRAP. Plus la $NCE$ se rapproche de 1, plus la mesure de confiance est prédictive. La $NCE$ est définie par :

$$NCE = \frac{H_{max} + \sum_{W_{correct}} \log_2(CM(W)) + \sum_{W_{incorrect}} \log_2(1 - CM(W))}{H_{max}} \quad (3.5)$$

où $H_{max} = -n \log_2(p_c) - (N - n) \log_2(1 - p_c)$,
n le nombre de mots correctement reconnus,
N le nombre total de mots reconnus,
$p_c = n/N$ la probabilité moyenne qu'un mot reconnu soit correct
et enfin $CM(W)$ la mesure de confiance associée à $W$.

Nous utiliserons cette mesure dans les chapitres suivants pour évaluer la nouvelle mesure de confiance que nous proposons.

## 3.2 Approches à l'état de l'art pour la détection automatiques des erreurs d'un SRAP

Cette section présente un survol des récents travaux sur la détection d'erreurs qui s'appuient sur deux approches différentes : les CRF et les réseaux de neurones.

### 3.2.1 Approches fondées sur les CRF

Nous avons présenté dans la section 3.1.4.2 différentes méthodes de classification qui permettent d'estimer une mesure de confiance pour un mot donné et de le classer dans l'une des deux catégories *correct* ou *erreur*. Cependant, ces méthodes ne prennent qu'une décision locale et ne sont pas conçues pour l'étiquetage de séquence.

Le problème de détection d'erreurs a été considéré ces derniers temps comme un problème d'étiquetage de séquence, du fait qu'un mot erroné peut engendrer des erreurs sur les mots voisins et créer toute une zone de mots erronés [Parada *et al.*, 2010, Béchet et Favre, 2013]. L'étiquetage de séquence consiste à estimer pour une séquence d'observation $X$ la séquence d'étiquettes $\widehat{Y}$ la plus probable :

$$\widehat{Y} = \arg\max_Y P(Y|X) = \arg\max_Y \frac{P(X,Y)}{P(X)} = \arg\max_Y P(X,Y) \quad (3.6)$$

De nombreux modèles statistiques pour l'étiquetage de séquence (modèles graphiques en particulier) ont été proposés dans la littérature. Les Modèles de Markov Cachés (*Hidden Markov Model* - HMM) sont les modèles les plus largement utilisés pour cette tâche. Ce sont des modèles génératifs qui définissent une probabilité jointe

$P(X, Y)$ des séquences d'observations et d'étiquettes. Pour estimer cette probabilité, le modèle génératif doit énumérer toutes les séquences d'observations possibles, ce qui en pratique n'est pas réalisable à cause du phénomène d'explosion combinatoire [Rabiner, 1989b].

Les modèles conditionnels discriminants comme les CRF [Lafferty *et al.*, 2001] sont proposés comme une alternative aux problèmes soulevés précédemment. Ces modèles définissent une loi de probabilité conditionnelle d'une séquence d'étiquettes sachant une séquence d'observation. L'avantage principal de ces modèles par rapport aux HMM est la possibilité d'utiliser l'ensemble des observations d'une séquence pour prédire une étiquette.

## Présentation des CRF

D'une manière générale, les CRF sont un processus stochastique qui modélise les dépendances entre une séquence d'observations discrètes $X = x_1, ..., x_T$ et un ensemble d'étiquettes $Y = y_1, ..., y_T$ associées aux observations de la séquence $X$.

D'après la théorie fondamentale des champs aléatoires [Hammersley et Clifford, 1971], la probabilité conditionnelle qu'une séquence d'étiquettes $Y$ soit associée à la séquence d'observations $X$ est définie par l'équation suivante :

$$P(Y|X) = \frac{1}{Z(X)} \exp(\sum_{t=2}^{T} \sum_{k=1}^{K} \lambda_k f_k(y_{t-1}, y_t, t)) \tag{3.7}$$

où $Z(X)$ désigne le facteur de normalisation défini par :

$$Z(X) = \sum_{y \in Y} \exp(\sum_{t=2}^{T} \sum_{k=1}^{K} \lambda_k f_k(y_{t-1}, y_t, x, t)) \tag{3.8}$$

Comme nous pouvons le voir dans l'équation 3.7, la probabilité P(Y|X) est obtenue par une combinaison linéaire de poids $\lambda_k$ associés à des fonctions $f_k$ sur la séquence d'observation. Les poids sont des paramètres à optimiser pendant l'apprentissage. Ils peuvent être interprétés comme l'importance de l'information donnée par la fonction $f_k$. $f_k(y_{t-1}, y_t, x, t)$ est la notation générale des fonctions $f_k$, appelée fonctions de descripteurs. Les CRF permettent de définir un ensemble de descripteurs contextuels à l'aide de modèles de combinaison contextuels (ou *template*) entre observation(s) et étiquette(s). Ce modèle permet de définir une combinaison contextuelle d'une ou plusieurs observations et d'une ou plusieurs étiquettes. Par exemple, le modèle $f(y_t, x_t)$ va prendre en compte tous les couples (étiquettes, observation) à chaque position $t$ dans la séquence.

La phase d'apprentissage consiste à estimer les vecteurs de poids $\theta = (\lambda_1, \lambda_2, ...)$. Le but est de maximiser la log-vraisemblance conditionnelle $\mathcal{L}(\theta)$ sur un corpus

d'apprentissage $D = (x^i, y^i)_{i=1}^n$, comme le montre l'équation suivante :

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} \log P_\theta(y_i|x_i) \tag{3.9}$$

Le décodage permet de trouver la meilleure séquence d'étiquettes $\widehat{y}$ pouvant être associée à une séquence d'observation donnée, qui maximise la probabilité $p(y|x)$ selon l'équation suivante :

$$\widehat{y} = \underset{y}{\text{argmax}}(p(y|x)) \tag{3.10}$$

Plusieurs méthodes ont été proposées pour l'apprentissage, cela dépend de la structure du CRF. Si les CRF ont une structure en chaîne, l'algorithme de *Viterbi* peut être utilisé [Do et Artières, 2006]. L'algorithme de *Belief Propagation* peut être utilisé s'ils ont une structure en arbre [Weiss et Freeman, 2001]. Dans le cas d'une structure quelconque le *Loopy Belief Propagation* est utilisé [Murphy *et al.*, 1999].

## Travaux connexes

Plusieurs travaux sur la détection d'erreurs sont fondés sur l'utilisation des CRF. Dans [Thomas et Trancoso, 2010], les auteurs ont prouvé la complémentarité des descripteurs issus du système de reconnaissance de la parole avec ceux extraits d'autres sources d'information.

Le problème de détection des mots hors vocabulaire est un cas particulier de la détection d'erreurs. Les mots hors vocabulaire ont un comportement et un impact proche des autres types d'erreurs, du fait qu'ils peuvent générer plusieurs erreurs. En effet, chaque mot mal reconnu est remplacé par une séquence de mots connus. Les auteurs dans [Parada *et al.*, 2010] se sont intéressés à la détection des régions d'erreurs générées par les mots hors vocabulaire. Ils ont proposé une approche fondée sur les CRF qui prend en compte les informations contextuelles des régions voisines au lieu de ne considérer que la région locale d'un mot hors vocabulaire.

Un autre cas particulier de la détection d'erreurs est celui qui concerne les noms propres présents dans le dictionnaire qui sont mal transcrits automatiquement. [Dufour *et al.*, 2012] s'intéresse à la détection et à la caractérisation de régions d'erreurs dans des transcriptions automatiques et plus particulièrement aux erreurs des noms de personnes. Deux approches ont été proposées. La première consiste à segmenter les transcriptions en deux régions *correct* ou *erreur*, et puis à associer une classe aux régions erronées. La deuxième traite ces deux problèmes conjointement. Pour cela, ils ont utilisé les CRF et l'algorithme Adaboost qui intègre plusieurs descripteurs notamment les bigrammes de mots, l'étiquetage grammatical, le regroupement en syntagme, les mesures de confiance d'un ASR et les durées du mot courant, précédent et suivant.

Une généralisation de la méthode de [Parada *et al.*, 2010] pour tous types d'erreurs est proposée par [Béchet et Favre, 2013]. Celle-ci est fondée sur les CRF uti-

lisant des descripteurs issus de systèmes de transcription, des descripteurs lexicaux, syntaxiques, la représentation orthographique du mot et des descripteurs contextuels.

### 3.2.2 Approches fondées sur les réseaux de neurones

Parallèlement à mes travaux de thèse, des travaux récents ont commencé à appliquer les réseaux de neurones pour la tâche de détection d'erreurs. Dans ces travaux, différentes architectures neuronales ont été exploitées (détaillées en section 1.1).

L'approche proposée par [Tam *et al.*, 2014] s'appuie sur l'utilisation d'un MLP composé de trois couches cachées qui intègre plusieurs sources d'information afin de détecter si un mot est *correct* ou *erroné*. On peut notamment citer des descripteurs extraits à partir du modèle de langage fondé sur les réseaux de neurones récurrents et des réseaux de confusion. D'autres descripteurs proviennent de la complémentarité de deux systèmes de reconnaissance de la parole.

Dans [Jalalvand et Falavigna, 2015], les auteurs ont proposé d'utiliser un MLP composé de deux couches cachées qui intègre des descripteurs issus d'un SRAP et d'autres syntaxiques. Ils proposent d'utiliser un *Stacked Denoising Auto-Encoder* (SDAE) pour faire le pré-apprentissage du MLP afin d'éviter l'initialisation aléatoire des poids. Ils ont montré que le SDAE est capable d'apprendre les erreurs faites par le SRAP beaucoup mieux que les autres classifieurs (SVM, l'entropie maximale, et *Extremely Randomized Tree*). Il permet également d'améliorer la généralisation du classifieur.

Les auteurs dans [Ogawa et Hori, 2015] utilisent un réseau de neurones bidirectionnel (Bi-RNN) pour la tâche de détection d'erreurs. Ils ont étudié trois types de tâches : la classification des mots en *correct* ou *erreur*, la détection des mots hors vocabulaire et la classification de types d'erreurs (insertion, suppression et suppression). Le Bi-RNN intègre des informations extraites du réseau de confusion, plus particulièrement des probabilités d'erreurs de substitution, d'insertion et de suppression. Ces nouvelles probabilités vont améliorer la classification de types d'erreurs et la détection des erreurs de suppression. De plus, ils ont été utilisés pour l'estimation du taux d'erreur sans faire appel à une transcription de référence. Contrairement aux autres approches neuronales citées précédemment, le mot est également utilisé comme descripteur. Il est modélisé par un vecteur *one-hot* en entrée du réseau. Les auteurs ont montré que les Bi-RNN généralisent mieux que les CRF quand ils sont entrainés sur un corpus de taille réduite.

## 3.3 Conclusion

Ce chapitre décrit un état de l'art sur la détection d'erreurs dans les transcriptions automatiques de la parole.

Dans un premier temps, nous avons présenté les différentes catégories de mesures de confiance proposées dans la littérature : le rapport de vraisemblance, la probabilité *a posteriori* et la combinaison des descripteurs.

Dans ces travaux de thèse, nous nous sommes intéressés à la dernière catégorie. C'est pourquoi, nous avons défini plusieurs types de descripteurs issus du SRAP et d'autres sources d'information. Nous avons également fait un point sur l'état de l'art de différents types de classifieurs permettant la combinaison de ces descripteurs.

Dans un second temps, nous avons présenté les approches à l'état de l'art pour la détection automatique d'erreurs qui s'appuient sur les CRF et plus récemment sur des réseaux de neurones. Dans ces travaux de thèse, nous avons utilisé les CRF comme système de base, tandis que nos contributions concernent l'exploitation des techniques d'apprentissage profond.

Deuxième partie

# Contributions

CHAPITRE 4

# SYSTÈME NEURONAL POUR LA DÉTECTION D'ERREURS : ÉTUDE PRÉLIMINAIRE

**Sommaire**

Dans ce chapitre, nous proposons une architecture neuronale pour la tâche de détection d'erreurs. Nous proposons également une étude sur l'utilisation de plusieurs types de *word embeddings* qui proviennent de différentes implémentations disponibles, notamment *w2vf-deps* fondé sur l'analyse de dépendance [Levy et Goldberg, 2014], *CBOW* et *Skip-gram* fournis par *word2vec* [Mikolov *et al.*, 2013a] et *GloVe* [Pennington *et al.*, 2014]. Ces différents *words embeddings* ont été détaillés dans le chapitre 1 (section 1.2.3).

Ce chapitre est organisé comme suit. Nous détaillons dans un premier temps le système de détection d'erreurs proposé : son architecture neuronale ainsi que les descripteurs (section 4.1). Nous présentons ensuite le protocole expérimental, le système de reconnaissance de la parole utilisé et les données expérimentales (section 4.2). Enfin, nous reportons les résultats expérimentaux dans la section 4.3, juste avant la conclusion (section 4.4).

## 4.1 Système de détection d'erreurs

Notre système de détection d'erreurs est fondé sur une architecture neuronale. Celle-ci est capable d'intégrer différents types de descripteurs, y compris par nature les *word embeddings*. Dans notre approche, la classification *Erreur* / *Correct* est réalisée en analysant chaque mot dans son contexte, comme présenté dans la figure 4.1.

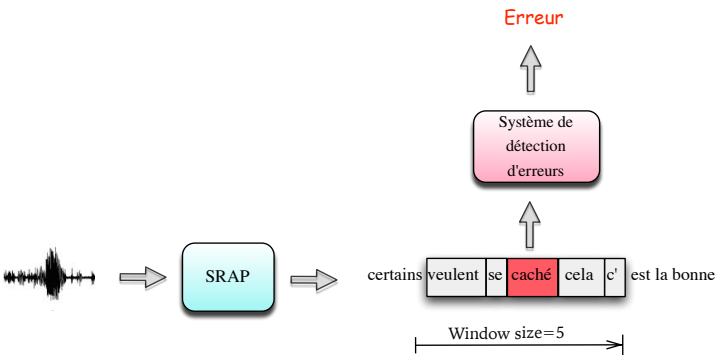



FIGURE 4.1 – Principe de fonctionnement d'un système de détection d'erreurs de mots issus d'un SRAP.

### 4.1.1 Architectures neuronales

Notre système s'appuie sur une stratégie multi-flux pour l'apprentissage du réseau de neurones, nommé Perceptron Multicouches Multi-Stream (*MLP-MS*). L'architecture *MLP-MS* est utilisée afin de mieux intégrer des informations contextuelles à partir des mots voisins. Elle est inspirée de [Estève *et al.*, 2015] où le mot et les descripteurs sémantiques sont intégrés pour l'identification de thèmes dans les conversations téléphoniques. L'apprentissage du *MLP-MS* repose sur le pré-apprentissage

de chaque couche cachée séparément, puis l'optimisation de l'ensemble du réseau (*pretraining* puis *fine-tuning*). Nous avons proposé de faire le *pretraining* de chaque couche cachée séparément afin de pallier le manque de données d'apprentissage et d'affiner les poids du réseau de neurones. Cela permet de guider le système dans l'extraction des informations importantes à partir des données en entrée.

Cette architecture, illustrée dans la figure 4.2, est détaillée comme suit : trois vecteurs de descripteurs sont présentés en entrée du réseau (le contenu d'un vecteur de descripteurs est détaillé dans la section suivante 4.1.2). Ces trois vecteurs sont respectivement un vecteur pour les deux mots gauches ($G$), un vecteur pour le mot courant ($W$) et un vecteur pour les deux mots droits ($D$).

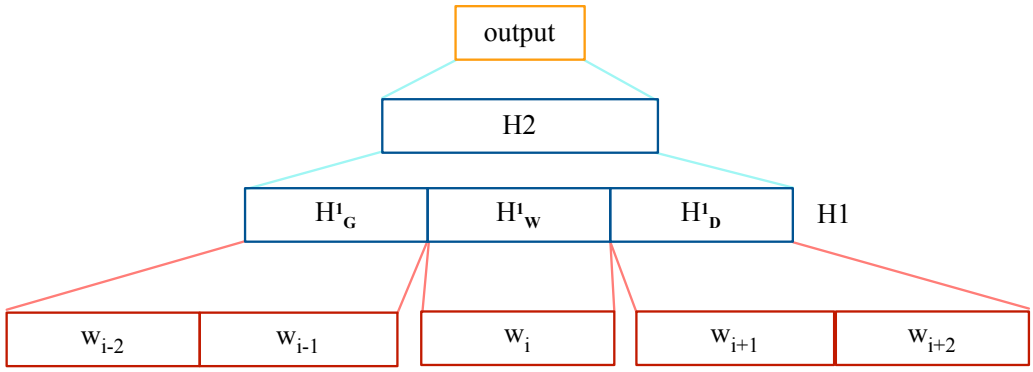



FIGURE 4.2 – L'architecture MLP-MS pour la détection d'erreurs de SRAP.

Chaque vecteur de descripteurs est utilisé séparément pour entraîner un MLP avec une seule couche cachée. En ne se reposant que sur un seul vecteur de descripteurs donné en entrée ($G$, $W$ ou $D$), chaque MLP est entrainé pour déterminer si le mot courant $w_i$ est correct ou erroné. Ce *pretraining* permet de construire des représentations spécialisées pour cette tâche, c'est à dire qui présentent une sensibilité sur le fait que le mot $w_i$ soit correct ou erroné. Formellement, l'architecture est décrite par les équations suivantes :

$$H^1_X = f(P^1_X.X + b^1_X) \tag{4.1}$$

où $X$ représente l'un des trois vecteurs de descripteurs ($G$,$W$ ou $D$), $f$ est la fonction d'activation ReLu, $P$ est la matrice de poids et $b$ est le vecteur de biais. Les vecteurs résultants $H^1_G$, $H^1_W$ et $H^1_D$ sont concaténés pour former la première couche cachée $H1$ :

$$H1 = H^1_G \oplus H^1_W \oplus H^1_D \tag{4.2}$$

Le vecteur $H1$ est utilisé ensuite comme entrée de la deuxième couche cachée $H2$, qui est définie par :

$$H2 = g(P^2.H1 + b^2) \tag{4.3}$$

où $g$ est la fonction d'activation tangente hyperbolique (tanh).

Enfin, la couche de sortie du *MLP-MS* est un vecteur $O_c$ de $c = 2$ unités qui correspondent aux deux étiquettes *correct* et *erreur*. La couche de sortie $O_c$ est calculée en fonction des sorties de la deuxième couche :

$$O_c = q(P^3.H2 + b^3) \tag{4.4}$$

où, $q$ est la fonction Softmax.

### 4.1.2 Descripteurs utilisés

Dans cette section, nous décrivons les descripteurs recueillis pour chaque mot et comment ils sont extraits. Certains de ces descripteurs sont identiques à ceux présentés dans [Béchet et Favre, 2013].
Chaque mot est représenté par un vecteur composé des descripteurs suivants :

— **Mesures de confiance du SRAP :** Ces mesures représentent les probabilités *a posteriori* (PAP) générées par le SRAP et se calculent comme défini dans la section 3.1.3.

— **Descripteurs lexicaux :** Il s'agit de la longueur du mot (nombre de lettres) et de trois indices binaires indiquant si les trois 3-grammes contenant le mot courant ont été vus dans le corpus d'apprentissage du modèle de langage du SRAP.

— **Descripteurs syntaxiques :** Ces descripteurs sont fournis par la boite à outils MACAON [1] [Nasr *et al.*, 2011] appliquée aux sorties du SRAP. Des analyseurs morphosyntaxiques et de dépendances sont utilisés pour extraire les étiquettes syntaxiques (POS), le mot gouverneur (le mot racine dans un arbre syntaxique) du mot courant et les liens de dépendance. Le lien de dépendance est une relation grammaticale tenue entre un mot et son gouverneur.

— **Mot :** La représentation orthographique du mot est utilisée dans l'étiqueteur de séquence fondé sur les CRFs comme dans [Béchet et Favre, 2013]. Dans le système neuronal, le mot est remplacé par son *word embedding* afin d'exploiter certaines généralisations extraites lors de la construction de ces *embeddings*. Comme il est indiqué dans l'introduction, nous allons tester quatre types différents de *word embeddings* : *w2vf-deps*, *CBOW* et *Skip-gram* et *GloVe*.

En entrée d'une architecture neuronale, chaque mot est représenté par le vecteur de descripteurs décrit dans la figure 4.3. Ici, le POS et le lien de dépendance sont remplacés par des vecteurs one-hot de dimensions 25 et 22 respectivement. Ces dimensions correspondent aux nombres d'étiquettes syntaxiques et de liens de dépendances proposés par MACAON. Le mot gouverneur est lui remplacé par son *word embedding*.

1. http ://macaon.lif.univ-mrs.fr

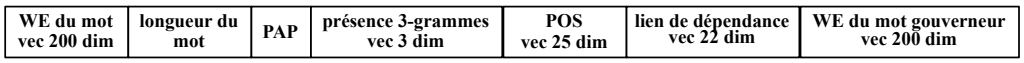



FIGURE 4.3 – Vecteur de descripteurs d'un mot en entrée de notre système de détection d'erreurs. L'acronyme WE est utilisé pour le terme *word embedding* et les expressions "vec X dim" pour "vecteur de X dimensions".

## 4.2 Protocole expérimental

### 4.2.1 Système de reconnaissance du LIUM

Dans cette section, nous allons nous intéresser au système de reconnaissance de la parole développé au sein du LIUM (SRAP LIUM) [Deléglise *et al.*, 2009, Bougares *et al.*, 2013]. C'est un système multi-passes fondé sur le décodeur CMU (*Carnegie Mellon University*) Sphinx [Lee *et al.*, 1990, Walker *et al.*, 2004], utilisant des modèles acoustiques GMM/HMM (chapitre 2). Plusieurs modifications ont été apportées au décodeur *CMU Sphinx*, afin de rendre le SRAP LIUM le plus performant possible.

L'architecture générale du SRAP LIUM est résumée dans la figure 4.4 [Estève, 2009]. Les étapes de développement du système, notamment la construction des ressources et le processus de décodage, y sont représentées. De plus, les bases de connaissance créées pendant la phase d'apprentissage et utilisées lors de la transcription y sont mises en évidence. Dans cette section nous allons détailler les composants du SRAP LIUM qui a remporté la campagne d'évaluation ETAPE en 2012 [Bougares *et al.*, 2013].

#### 4.2.1.1 Apprentissage

Le système de reconnaissance automatique de la parole utilisé pour nos expériences a été spécialement construit pour ce travail sur les erreurs, dans le cadre du projet VERA. Il n'utilise que des données d'apprentissage accessibles.

##### Données d'apprentissage

Ainsi, le corpus d'apprentissage des modèles acoustiques est constitué de l'ensemble des données du corpus ESTER [Galliano *et al.*, 2005] (environ 100 heures de parole transcrite) étendu d'une soixantaine d'heures du corpus EPAC [Estève *et al.*, 2010], d'une quinzaine d'heure du corpus ESTER 2 [Galliano *et al.*, 2009] et de toutes les données du corpus REPERE (35h).

Au total, ce corpus d'apprentissage est composé d'environ 210 heures transcrites manuellement d'émissions radiophoniques et d'émissions télévisées.

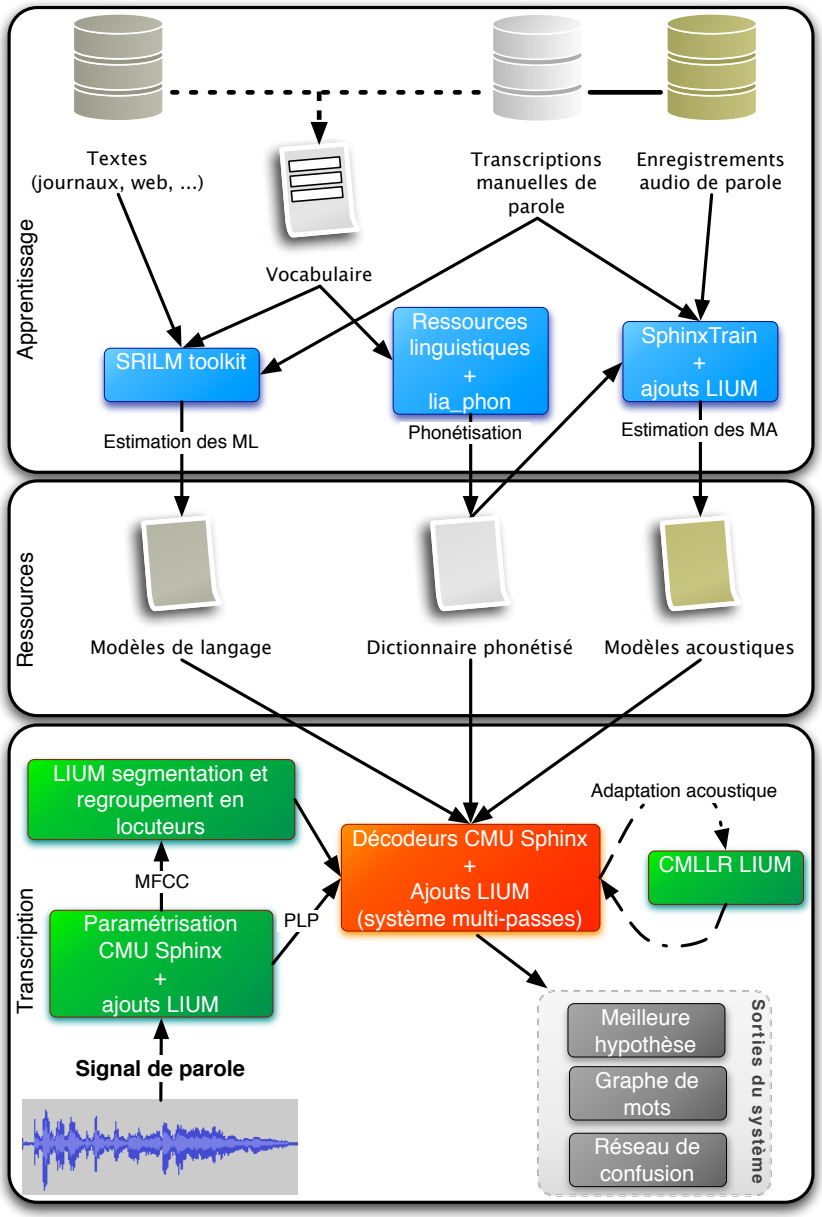



FIGURE 4.4 – Architecture générale du système de transcription automatique du LIUM [Estève, 2009].

#### Modèles acoustiques

Les modèles acoustiques utilisent des paramètres acoustiques de type PLP, sous forme de vecteurs à 39 dimensions (13 valeurs PLP avec l'énergie, complétées de leurs dérivées et de leurs dérivées secondes).

Dans l'approche multi-passes de ce système, les premiers modèles acoustiques sont utilisés afin de générer une première transcription automatique en vue d'une adaptation automatique au locuteur et aux conditions acoustiques. Ces modèles gaussiens comportent 2500 états partagés de GMM, modélisés par un mélange de 22 gaussiennes.

Pour le second et les décodages acoustiques suivants, les modèles, également de type GMM/HMM, utilisent 10000 états partagés, chaque état étant modélisé par un mélange de 48 modèles gaussiens. Un apprentissage de type *Speaker Adaptative Training* (SAT) [Gales, 2000] est réalisé sur l'ensemble du corpus d'apprentissage, suivi d'un apprentissage discriminant de type Minimum Phone Error [Povey et Woodland, 2002].

#### Vocabulaire

Le vocabulaire est composé d'environ 160 000 mots et correspond au vocabulaire qui avaient été utilisé par le LIUM pour la campagne REPERE, qui engloble le vocabulaire utilisé durant la campagne ETAPE : il s'agit d'un vocabulaire spécialisé pour le traitement de journaux d'actualités.

#### Modèles de langage

Deux modèles de langage sont utilisés : un modèle de langage 3-grammes et un modèle de langage 4-grammes, comme décrit dans la sous-section suivante. Les deux modèles de langage sont construits sans *cut-off*, en utilisant la technique de discounting dite de Kneser-Ney modifié [Chen et Goodman, 1996].

Ces modèles ont été construits par interpolation linéaire de modèles dont les sources sont des articles du journal Le Monde, des Google News, un corpus Gigaword, et des transcriptions manuelles de parole (celles utilisées pour l'apprentissage des modèles acoustiques).

Ils contiennent environ 464 millions de 4-grammes [2], 199 millions de 3-grammes et 36 millions de 2-grammes.

### 4.2.1.2 Transcription

L'obtention de bonnes performances en reconnaissance de la parole nécessite la construction de bases de connaissance : modèles acoustiques, dictionnaire de prononciation et modèles de langage. Ces connaissances sont utilisées par le SRAP pour le processus de transcription qui est réalisé en deux étapes : la segmentation et le décodage.

2. Bien sûr, les 4-grammes ne sont présents que dans le modèle 4-grammes.

### Système de segmentation et de regroupement en locuteurs

Le signal audio contient souvent, en plus de la parole, d'autres événements sonores (musiques, publicité, pauses,*etc.*). Ces événements ne doivent pas être traités par le SRAP sans quoi cela induirait certainement des insertions de mots qui nuiraient à la qualité des sorties du système. Le traitement du signal audio par un système de segmentation parole/non parole est donc nécessaire.

Un système de segmentation en locuteur vise à regrouper les segments acoustiquement homogènes en termes de locuteurs, sexe et largeur de bande. Ces informations sont utilisées pour adapter les modèles acoustiques et créer des modèles spécialisés afin d'utiliser le modèle acoustique approprié à chaque segment.

Le système de segmentation et regroupement en locuteurs utilisé a été développé au LIUM [Meignier et Merlin, 2010]. Il repose sur le Critère d'Information Bayesien (BIC) [Chen et Gopalakrishnan, 1998] et se compose de trois étapes :

— Le signal est découpé en petits segments acoustiquement homogènes,

— Les segments sont ensuite regroupés en classes de locuteurs sans changer les frontières,

— Enfin, les frontières sont ajustées.

Le système de segmentation et regroupement en locuteurs a gagné la campagne d'évaluation ESTER 2 [Galliano *et al.*, 2009]. Il est présenté en détails dans [Meignier et Merlin, 2010].

### Décodage : Système de transcription multi-passes

Le SRAP LIUM est un système multi-passes. Chaque étape utilise un algorithme de recherche manipulant des données de l'étape précédente et propose une nouvelle hypothèse de reconnaissance. Le processus de décodage se déroule en cinq étapes :

1. Dans la première, un traitement utilisant le décodeur rapide de *CMU Sphinx* 3 est appliqué sur des paramètres acoustiques PLP. Le décodeur utilise dans cette passe un modèle de langage 3-grammes et des modèles acoustiques adaptés au sexe du locuteur (homme/femme) et aux conditions acoustiques (audio/téléphone).
2. Dans la seconde étape, une matrice de transformation CMLLR est calculée de façon à adapter les paramètres acoustiques aux seconds jeux de modèles acoustiques. Ces modèles sont estimés à l'aide des méthodes SAT et MPE. Le traitement est réalisé de nouveau avec le décodeur CMU Sphinx 3, en utilisant le modèle de langage 3-grammes de la première passe. Le système produit en sortie un graphe de mots.
3. Dans la troisième étape, le graphe de mots est utilisé pour faire le décodage du graphe avec un vrai contexte droit des phonèmes en fin de mot. Le but de cette étape est d'obtenir une meilleure précision acoustique, en particulier dans les zones inter-mots. Les mêmes modèles acoustiques et de langage que lors de la deuxième étape sont utilisés, avec l'application de la même matrice

de transformation CMLLR sur les paramètres acoustiques. Un nouveau graphe de mots est généré.

4. Lors de la quatrième étape un modèle 4-grammes est utilisé pour recalculer les scores linguistiques des mots du graphe généré pendant la passe précédente.
5. Enfin, le graphe de mots issu de la quatrième passe est transformé en un réseau de confusion. La méthode de consensus [Mangu *et al.*, 2000] permet d'obtenir l'hypothèse de reconnaissance finale et de disposer pour chaque mot d'une probabilité *a posteriori* pouvant être employée comme mesure de confiance.

### 4.2.2 Données expérimentales

Les données expérimentales sont issues du corpus français ETAPE [Gravier *et al.*, 2012], composé d'enregistrements audio d'émissions de journaux télévisés et de leurs transcriptions manuelles. Ce corpus est enrichi avec des transcriptions automatiques générées par le SRAP LIUM.

Les transcriptions automatiques ont été alignées avec les transcriptions de référence en utilisant l'outil *sclite*[3]. À partir de cet alignement, chaque mot dans le corpus a été étiqueté *correct* ou *erreur*.

Partant du fait que l'apprentissage de réseaux de neurones nécessite une quantité de données conséquente, nous avons décidé de ne pas utiliser la répartition officielle d'ETAPE et d'augmenter la taille du corpus d'apprentissage de 28% en nombre de mots. De plus, nous avons réparti les fichiers audio d'une manière homogène : *i.e.* les fichiers audio sont répartis en fonction de leurs WER, sur les trois corpus Train, Dev et Test. La description de ces corpus, en termes de nombre de mots de référence et d'hypothèse ainsi que le taux d'erreurs mots, est présentée dans le tableau 4.1.

| **Nom** | **#mots ref** | **#mots hyp** | **WER%** |
|---|---|---|---|
| Train | 349K | 316K | 25,3 |
| Dev | 54K | 50K | 24,6 |
| Test | 58K | 53K | 21,9 |

TABLE 4.1 – Description des données expérimentales.

En ce qui concerne les *word embeddings*, ils ont été calculés à partir d'un vaste corpus textuel, composé d'environ 2 milliards de mots. Ce corpus a été construit à partir des articles du journal français *Le Monde*, le corpus *Gigaword*, d'articles fournis par *Google News* et de transcriptions manuelles d'environ 400 heures d'émissions françaises. Ce corpus est annoté par des étiquettes syntaxiques en utilisant la boite à outils MACAON [Nasr *et al.*, 2011]. La version annotée est utilisée pour l'apprentissage de *w2vf-deps* (section 1.2.3.4), tandis que la version non annotée est utilisée pour l'apprentissage de *CBOW* (section 1.2.3.3), *Skip-gram* (section 1.2.3.3) et *GloVe* (section 1.2.3.5).

3. http://www.icsi.berkeley.edu/Speech/docs/sctk-1.2/sclite.htm

### 4.2.3 Métriques d'évaluation

Les résultats expérimentaux présentés par la suite seront analysés à l'aide de métriques d'évaluation couramment utilisées dans la littérature pour l'analyse des performances d'un système de détection d'erreurs.
Des mesures classiques utilisées en recherche d'information sont appliquées pour étudier les performances des systèmes sur l'attribution spécifique de l'étiquette *erreur*.

— Rappel (R) : mesure la capacité du système à prédire l'ensemble des hypothèses pertinentes. Le rappel se calcule comme suit :

$$R = \frac{\textit{Nombre de mots correctement étiquetés} \text{ erreur}}{\textit{Nombre total de mots étiquetés} \text{ erreur } \textit{dans la référence}} \tag{4.5}$$

— Précision (P) : mesure la capacité du système à prédire une hypothèse valide. La précision se calcule ainsi :

$$P = \frac{\textit{Nombre de mots correctement étiquetés} \text{ erreur}}{\textit{Nombre total de mots étiquetés} \text{ erreur } \textit{par le système}} \tag{4.6}$$

— F-mesure (F) : mesure la performance du système, en combinant la précision et le rappel par la moyenne harmonique.

$$F = \frac{2 \times P \times R}{P + R} \tag{4.7}$$

Enfin, nous utiliserons la mesure d'erreur de classification (CER) pour évaluer notre système globalement. Elle se calcule ainsi :

$$CER = \frac{\textit{Nombre de mots incorrectement étiquetés}}{\textit{Nombre total de mots}} \tag{4.8}$$

## 4.3 Performance du système préliminaire

Cette section présente les résultats expérimentaux de notre système préliminaire de détection d'erreurs *MLP-MS* et les compare à un système état de l'art fondé sur les CRFs implémenté avec *Wapiti*[4] [Lavergne *et al.*, 2010] et proposé par [Béchet et Favre, 2013].

Ces deux systèmes intègrent l'ensemble des descripteurs présentés dans la section 4.1.2. Ils sont entrainés sur le corpus d'apprentissage (Train) et sont appliqués au corpus de test (Test). Le corpus de développement (Dev) est utilisé pour optimiser les hyperparamètres suivants du *MLP-MS* : le taux d'apprentissage (learning rate) et la taille du mini-lot (*mini-batch*). Pour le système CRF, l'optimisation se fait sur la construction du *template* qui sélectionne l'ensemble des descripteurs utilisés. Le *template* permet de générer toutes les observations sur lesquelles les CRFs vont appuyer leur apprentissage.

4. http://wapiti.limsi.fr

Un ensemble d'expériences est effectué afin d'évaluer la performance des différents types de *word embeddings* de 200 dimensions chacun. Nous rappelons que le système *MLP-MS*, défini dans la section 4.1.1, est composé de deux couches cachées et considère en entrée une fenêtre de cinq mots. Le choix de ces hyperparamètres est justifié par le fait que nous envisageons d'ajouter d'autres descripteurs, ce qui implique une augmentation du nombre de neurones à construire, donc notre choix doit être raisonnable et nous semble adapté à la quantité de données disponibles pour l'apprentissage du réseau.

### Performance des word embeddings

Le tableau 4.2 présente la performance du système MLP-MS fondé sur différents *word embeddings*, ainsi que la performance du système CRF. On remarque qu'en termes de taux d'erreurs de classification, les résultats obtenus par *MLP-MS* sont meilleurs que l'approche CRF, en particulier en utilisant *w2vf-deps*. Cette performance est liée à l'utilisation de contextes syntaxiques fondés sur l'analyse de dépendance qui permet de capturer plus d'informations que les contextes fondés sur l'approche sac de mots continu. Le *word embeddings w2vf-deps* améliore les résultats de 3% et 3,5% en termes de réduction du CER par rapport à l'approche CRF respectivement sur Dev et Test.

En termes de F-mesure appliquée aux mots erronés, l'approche CRF obtient de meilleurs résultats : cela est lié à la valeur élevée du rappel des CRFs tandis que les approches neuronales obtiennent des valeurs de précision plus élevées.

| **Corpus** | **Approches** | **Représentation du mot** | **Label erreur** | | | **Globale CER** |
|---|---|---|---|---|---|---|
| | | | **P** | **R** | **F** | |
| Dev | MLP-MS | w2vf-deps | 0,730 | 0,501 | 0,594 | **10,06** |
| | | Skip-gram | **0,750** | 0,455 | 0,567 | 10,24 |
| | | GloVe | 0,738 | 0,469 | 0,574 | 10,26 |
| | | Cbow | 0,722 | 0,490 | 0,584 | 10,28 |
| | CRF | discrète | 0,681 | **0,553** | **0,610** | 10,38 |
| Test | MLP-MS | w2vf-deps | 0,719 | 0,509 | 0,596 | **8,26** |
| | | Skip-gram | **0,744** | 0,467 | 0,574 | 8,30 |
| | | GloVe | 0,721 | 0,469 | 0,569 | 8,53 |
| | | Cbow | 0,711 | 0,482 | 0,575 | 8,54 |
| | CRF | discrète | 0,676 | **0,547** | **0,605** | 8,56 |

TABLE 4.2 – Comparaison des performances des différents *word embeddings* dans MLP-MS et du système CRF.

### Test de significativité

Nous avons trouvé que le *word embedding w2vf-deps* obtient les meilleurs résultats en termes de CER en comparaison avec les autres *embeddings* et l'approche

CRF. Mais, la question qui se pose ici est de savoir si la différence entre les systèmes est significative ou non. C'est pourquoi nous proposons d'utiliser l'intervalle de confiance (IC) à 95% reposant sur la loi de *Student* pour mesurer la significativité des résultats obtenus. Il s'agit d'un intervalle qui représente la fourchette de valeurs à l'intérieur de laquelle nous sommes certains à 95% de trouver la vraie valeur recherchée. Plus généralement, l'intervalle de confiance permet d'évaluer la précision de l'estimation d'un paramètre statistique sur un échantillon.

La loi de Student ne s'applique que sur une variable aléatoire qui suit le comportement de la loi normale. Nous devons donc vérifier que cela est le cas pour notre taux d'erreurs de classification. Pour cela, nous mettons en place une expérience simple. Elle consiste à construire 1000 mini-corpus contenant 50% des données de Test choisies aléatoirement. Puis, on calcule le CER pour chacun de ces mini-corpus. Enfin, on construit la courbe de l'effectif en fonction du CER. Si la courbe a la forme d'une gaussienne, nous considérons que le CER suit une loi normale.

Nous avons fait cette expérience sur les sorties des cinq systèmes du tableau 4.2 et les courbes sont illustrées dans la figure 4.5. On observe que ces courbes ont bien la forme d'une gaussienne, ce qui confirme que nous pouvons utiliser l'intervalle de confiance à 95% pour mesurer la significativité des résultats.

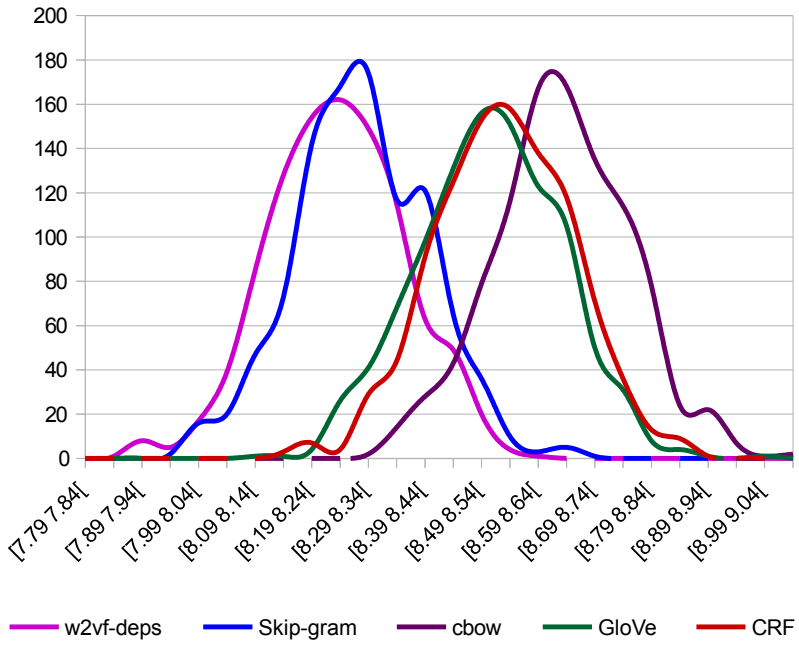



FIGURE 4.5 – Distribution de CER par intervalle pour les cinq systèmes : w2vf-deps, Skip-gram, cbow, GloVe et CRF.

Ainsi, nous obtenons un intervalle de confiance de $0,26$ et $0,23$ respectivement sur Dev et Test. Les réductions du CER observées entre les différents systèmes

MLP-MS ne sont donc pas significatives. En revanche, le meilleur système MLP-MS (*w2vf-deps*) est significativement meilleur que l'approche CRF.

### Performance de MLP-MS

Afin de nous assurer de la performance de l'architecture neuronale proposée, nous suggérons d'évaluer la performance de ses *word embeddings* en utilisant un simple MLP composé de deux couches cachées. Ici, le mot et ses contextes sont projetés dans la même position et les poids sont initialisés aléatoirement. Les résultats de ces expériences sont résumés dans le tableau 4.3 et sont comparés à ceux dans le tableau 4.2. Ces résultats montrent la performance du *MLP-MS*, surtout sa capacité de généralisation sur Test observée pour *w2vf-deps* et *Skip-gram*. Ceci corrobore notre proposition pour faire le *pretraining* des couches cachées séparément, qui a pour avantage entre autres d'aider le système à extraire des informations importantes des entrées.

| **Corpus** | **Approches** | **Représentation du mot** | **Label erreur** | | | **Globale CER** |
|---|---|---|---|---|---|---|
| | | | **P** | **R** | **F** | |
| Dev | MLP | w2vf-deps | 0,709 | 0,508 | 0,592 | 10,29 |
| | | Skip-gram | 0,751 | 0,446 | 0,560 | 10,32 |
| | | GloVe | 0,738 | 0,429 | 0,429 | 10,39 |
| | | Cbow | 0,754 | 0,434 | 0,551 | 10,40 |
| Test | MLP | w2vf-deps | 0,694 | 0,515 | 0,591 | 8,53 |
| | | Skip-gram | 0,739 | 0,444 | 0,555 | 8,53 |
| | | GloVe | 0,739 | 0,445 | 0,556 | 8,64 |
| | | Cbow | 0,742 | 0,433 | 0,547 | 8,59 |

TABLE 4.3 – Résultats comparatifs de l'utilisation des *word embeddings* du MLP-MS dans un simple MLP.

### Discussion

D'après les résultats présentés dans le tableau 4.3, on remarque que les *word embeddings* obtiennent des résultats relativement similaires en terme de CER, mais cela n'empêche pas une complémentarité potentielle. Cette complémentarité peut être suspectée par leurs résultats hétérogènes en termes de Précision/Rappel.

## 4.4 Conclusion

Dans ce chapitre, nous avons présenté un système préliminaire pour la détection d'erreurs. Le processus de détection d'erreurs est résumé dans la figure 4.6.

Le système proposé consiste à attribuer une étiquette *correct* ou *erreur* à chaque mot en s'appuyant sur les informations du mot courant et contextuelles des mots

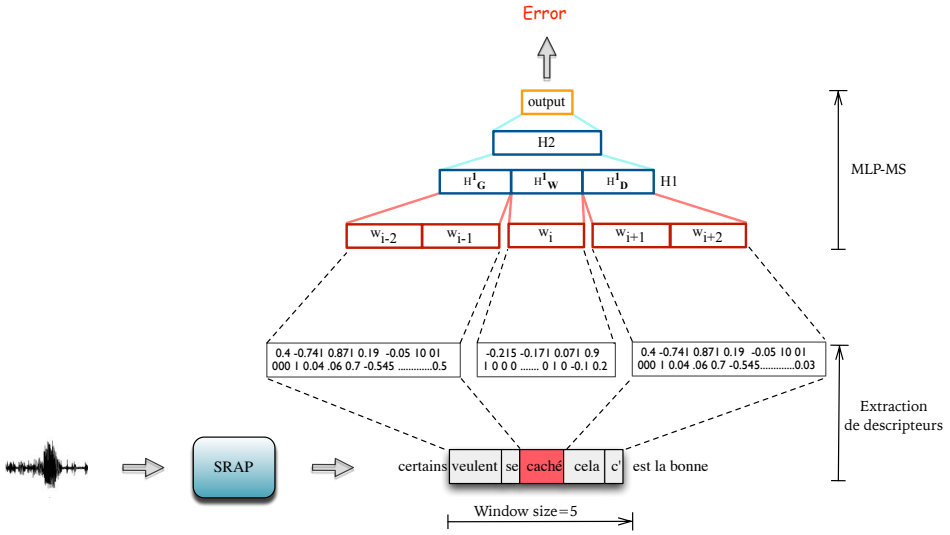



FIGURE 4.6 – Processus de détection d'erreurs dans les transcriptions automatiques de la parole

voisins par le biais d'un ensemble de descripteurs. Il s'appuie sur une architecture neuronale capable d'intégrer différents types de descripteurs en entrée, y compris les *word embeddings*. Nous avons étudié l'utilisation de plusieurs types de *word embeddings* provenant de différentes implémentations disponibles, notamment *w2vf-deps* fondé sur l'analyse de dépendances, *CBOW* et *Skip-gram* fournis par *word2vec* et *GloVe*.

La partie expérimentale effectuée sur les transcriptions automatiques du corpus ETAPE généré par le SRAP LIUM, a montré la validité de notre approche pour la détection d'erreurs. Les résultats obtenus sont meilleurs que ceux d'un système état de l'art fondé sur les champs aléatoires conditionnels (CRF). En effet, la performance de l'approche neuronale est due probablement à l'exploitation de la nature continue des *word embeddings*.

L'évaluation de l'impact des différents *word embeddings* sur la tâche de détection d'erreurs a montré une complémentarité potentielle. Dans le but d'étudier cette complémentarité, nous aborderons dans le chapitre suivant une comparaison systématique de leur impact sur plusieurs tâches d'évaluations. Si une éventuelle complémentarité apparaît, nous allons étudier et évaluer différentes approches pour combiner ces *embeddings* afin de tirer profit de leurs complémentarités pour la tâche de détection d'erreurs.

CHAPITRE 5

# ÉTUDE DES WORD EMBEDDINGS LINGUISTIQUES

**Sommaire**

Les *word embeddings* constituent une projection des mots du vocabulaire dans un espace continu de faible dimension, de manière à ce que les mots qui apparaissent dans des contextes similaires soient proches les uns des autres dans l'espace de projection. L'hypothèse sous-jacente de ce type de représentations est l'obtention de représentations génériques adaptées à la plupart des tâches de traitement automatique des langues naturelles (TALN).

De nombreuses études se sont focalisées sur l'évaluation des qualités intrinsèques des *word embeddings*, ainsi que sur leurs effets quand ils sont utilisés en entrée d'un système pour une tâche donnée. Dans [Turian *et al.*, 2010], les auteurs ont évalué différents types de *word embeddings* ainsi que leurs concaténations sur les tâches de regroupement en syntagmes et la reconnaissance d'entités nommées. D'autres évaluations se sont focalisées sur les tâches d'analogie et de similarité, comme dans [Mikolov *et al.*, 2013a, Levy et Goldberg, 2014, Pennington *et al.*, 2014, Gao *et al.*, 2014, Levy *et al.*, 2015, Ji *et al.*, 2015]. Les tâches de similarité et d'analogie permettent d'évaluer respectivement les similarités *attributionnelles* et *relationnelles* entre les mots.

Les *word embeddings* sont conçus pour capturer ce que [Turney, 2006] appelle des *similarités attributionnelles* entre les mots du vocabulaire : les mots qui apparaissent dans des contextes similaires seront proches les uns des autres dans l'espace de projection. L'idée est de regrouper les mots qui partagent les mêmes propriétés sémantiques ("chien chat vache", "manger dévorer") et/ou syntaxiques ("jours chapeaux voitures", "vidés portés dansés"). En effet, le degré de similarité attributionnelle entre deux mots A et B, dépend du degré de correspondance entre leurs propriétés. Une mesure de similarité attributionnelle est une fonction qui associe deux mots A et B, à un nombre réel $Sim_{at}(A, B) \in \mathbb{R}$. Plus il y a de correspondances entre les propriétés de A et B, plus leur similarité attributionnelle est élevée. Par exemple, le chien et le chat ont un degré élevé de similarité attributionnelle.

Les auteurs dans [Mikolov *et al.*, 2013c] ont montré que les *word embeddings* créés par un réseau de neurones récurrent captent non seulement les similarités attributionnelles entre les mots, mais aussi des similarités entre des paires de mots. Cette similarité est nommée *régularité linguistique* par [Mikolov *et al.*, 2013c] ou *similarité relationnelle* par [Turney, 2006]. Ces similarités peuvent capturer des relations de genre (comme "homme : femme", "roi : reine"), de nombre (comme "pommes : pomme", "voitures : voiture"), *etc.* Le degré de similarité relationnelle entre deux paires de mots A : B et C : D dépend du degré de correspondance entre la relation entre A et B et la relation entre C et D. Une mesure de similarité relationnelle est une fonction qui associe deux paires de mots A : B et C : D à un nombre réel $Sim_r$(A : B, C : D) $\in \mathbb{R}$. Plus il y a de correspondances entre les relations de A : B et C : D, plus leur similarité relationnelle est élevée. Par exemple, la paire de mots "chien : aboyer" et "chat : miauler" a un degré élevé de similarité relationnelle.

Dans ce chapitre, nous allons nous intéresser à l'évaluation et à la combinaison de différents types de *word embeddings* afin de tirer profit de leur complémentarité. Nous présentons dans un premier temps les différentes tâches d'évaluations, ainsi que les résultats d'évaluations des différents *word embeddings* (section 5.1). Nous nous

intéressons par la suite à leurs combinaisons selon plusieurs approches. La définition de ces approches, et les performances des *word embeddings* combinés sur les tâches d'évaluations et la tâche de détection d'erreurs sont reportées dans la section 5.2, juste avant la conclusion (section 5.3).

## 5.1 Évaluation des word embeddings

Nous avons évalué dans le chapitre précédent la performance de différents *word embeddings* sur la tâche de détection d'erreurs. Cette évaluation a mis en évidence une potentielle complémentarité. Afin d'étudier cette complémentarité, nous proposons d'évaluer d'une manière rigoureuse ces *word embeddings* pour mieux comprendre leurs points forts et points faibles. Cette évaluation est effectuée sur différentes tâches classiques du TALN, ainsi que sur des tâches d'analogie et de similarité.

Plusieurs travaux [Turian *et al.*, 2010, Mikolov *et al.*, 2013a, Pennington *et al.*, 2014, Levy *et al.*, 2015, Ji *et al.*, 2015] se sont précédemment intéressés à l'évaluation des *word embeddings*, mais certains des *word embeddings* n'ont jamais été comparés (par exemple, *w2vf-deps*, *Skip-gram* et *GloVe*). De plus, certains travaux publiés [Mikolov *et al.*, 2013a, Mikolov *et al.*, 2013b, Bansal *et al.*, 2014b] fournissent des évaluations où les embeddings ne sont pas construits sur les mêmes données, avec le même vocabulaire ou avec la même dimension.
Dans notre étude, les *word embeddings* que nous allons évaluer seront estimés sur le même corpus, avec le même vocabulaire, la même dimension, et la même taille de fenêtre contextuelle. Les tâches d'évaluations seront effectuées en langue anglaise car il n'y a actuellement pas de grands corpus d'évaluation disponibles en français pour toutes les tâches d'évaluation visées.

### 5.1.1 Tâches d'évaluation : Benchmarks

#### 5.1.1.1 Tâches classiques du traitement automatique des langues naturelles (TALN)

Dans cette section, nous présentons les différentes tâches classiques relatives au TALN sur lesquelles nous évaluons la performance des différents *word embeddings*.

— **L'étiquetage morpho-syntaxique (POS)** consiste à attribuer à chaque unité lexicale du corpus dans le contexte où elle apparaît une étiquette parmi les $n$ étiquettes morpho-syntaxiques. Cette étiquette représente la catégorie grammaticale (nom, verbe, adjectif, ...) du mot, ainsi que les informations morphologiques associées (masculin, féminin, singulier, ...). Le système est évalué sur la répartition Train, Dev et Test du corpus *Penn Treebanck Benchmark* [Marcus *et al.*, 1993].

— **Le regroupement en syntagme (CHK)** est une étape intermédiaire pour faire une analyse complète, aussi appelé analyse peu profonde. Cette tâche vise à attribuer des étiquettes syntaxiques pour les segments (ou *chunks*) d'une phrase (groupe verbal (GV), groupe nominal (GN), *etc.*). À chaque

mot dans un chunk est associée une étiquette indiquant son type accompagné du codage BIO (pour *begin*, *inside*, *outside*), qui permet d'identifier si le mot correspondant est au début, à l'intérieur ou à l'extérieur d'un segment. Par exemple B-GV sera associé à la première unité lexicale d'un groupe verbal. Ceci donne au total $n$ étiquettes. Le regroupement en syntagmes est souvent évalué en utilisant l'approche suivie dans *CoNLL 2000*[1] *Benchmark* [Tjong Kim Sang et Buchholz, 2000]. Les sections 15-18 du corpus *Wall Street Journal* (WSJ) sont utilisées pour l'apprentissage et la validation, pendant que la section 20 est utilisée pour le test.

— **La reconnaissance d'entités nommées (NER)** consiste à rechercher dans une phrase un mot, ou un groupe de mots catégorisable dans des classes telles que les noms de personnes, les noms d'organisations et les noms de lieux. Chaque mot est associé à une étiquette parmi $n$ préfixées par un indicateur BIO. Le système est évalué avec l'approche suivie dans *CoNLL 2003*[2] *Benchmark* [Tjong Kim Sang et De Meulder, 2003].

— **La détection de mention (MD)** consiste à identifier les mentions d'entités dans le texte, pour la tâche de résolution de coréférence. Celle-ci vise à déterminer quelles sont les mentions qui réfèrent à la même entité. Une mention peut être de type nom (Alex, Jean, ...), nominal (ingénieur, dentiste, ...) ou pronominal (il, je, ...), *etc.* Comme dans les autres tâches TALN, la tâche de détection de mention est formulée comme un problème de classification de séquences, en attribuant à chaque mot dans le texte une étiquette indiquant si le mot est au début, à l'intérieur ou à l'extérieur d'une mention ($n$ étiquettes au total). Cette tâche est effectuée sur le corpus *Ontonotes* [Hovy *et al.*, 2006] avec la répartition de *CoNLL 2012*[3].

Ici $n$ est le nombre d'étiquettes proposé dans les corpus qui correspondent respectivement à 48, 22, 8 et 3 pour les tâches POS, CHK, NER et MD.

#### 5.1.1.2 Tâche d'analogie

La tâche d'analogie proposée par [Mikolov *et al.*, 2013c] permet d'évaluer la similarité relationnelle entre les paires de mots. Cette tâche consiste à répondre à des questions d'analogie en s'appuyant sur les similarités cosinus calculées sur les *word embeddings*. Elle est évaluée par le taux d'exactitude des questions ayant obtenu une réponse correcte. Le taux d'exactitude (*Accuracy* - Acc.) est défini par le rapport entre le nombre de réponses correctement assignées et le nombre total de réponses.

[Mikolov *et al.*, 2013c] ont suggéré que la similarité relationnelle entre les paires de mots est présente sous forme de translations vectorielles entre les paires de mots, de sorte que dans l'espace de projection, toutes les paires de mots qui partagent une relation particulière sont liées par le même vecteur de translation ($\overrightarrow{pommes}$ -

1. http://www.cnts.ua.ac.be/conll2000/chunking
2. http://www.cnts.ua.ac.be/conll2003/ner
3. http://conll.cemantix.org/2012/introduction.html

$\overrightarrow{pomme} \approx \overrightarrow{voitures}$ - $\overrightarrow{voiture}$). Autrement dit, comme présenté dans la figure 5.1, le vecteur de translation $o_1$ entre les *word embeddings* de "roi" et "reine" doit être égal au vecteur de translation entre les *word embeddings* d'"homme" et "femme" qui ont une relation de genre. Par contre les paires de mots qui ont une relation de nombre (figure à droite) doivent avoir un vecteur de translation $o_2$ différent de $o_1$.

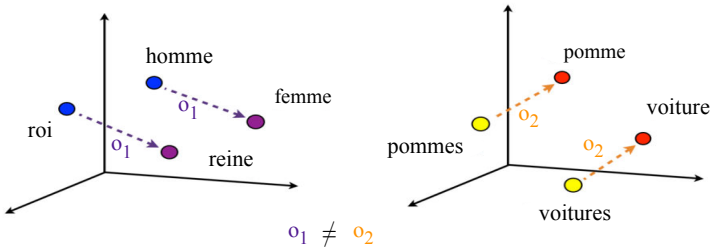



FIGURE 5.1 – Visualisation des quatre paires de mots qui ont des relations de genre (à gauche) et de nombre (à droite).

La relation de similarité entre les paires de mots du même type (*e.g.* genre) peut être transformée en une question d'analogie. Cette question peut se poser de cette manière : "quel est le mot similaire à roi de la même manière que homme est similaire à femme ?". L'utilisation des opérations arithmétiques simples sur les *embeddings* des mots (nommée aussi l'arithmétique vectorielle), permet de répondre à n'importe quelle question d'analogie de la forme "$a : a^* \rightarrow b : b^*$", où $b^*$ est le mot inconnu à trouver dans le vocabulaire $V$.

Deux méthodes ont été proposées pour répondre aux questions d'analogies : *3COSADD* et *3COSMUL*. La méthode *3COSADD* proposée par [Mikolov *et al.*, 2013c] consiste d'abord à additionner et soustraire les vecteurs des mots, puis à chercher le mot le plus proche du résultat $b^*$ en termes de distance cosinus :

$$3COSADD = \arg_{b^* \in V} \max(\cos(b^*, a^* - a + b)) \tag{5.1}$$

Les auteurs dans [Levy *et al.*, 2014] ont montré que l'équation 5.1 est équivalente à la recherche d'un mot qui maximise une combinaison linéaire de similarités de trois paires de mots. Si tous les vecteurs de mots sont normalisés, l'équation 5.1 peut s'écrire de la manière suivante :

$$3COSADD = \arg_{b^* \in V} \max(\cos(b^*, b) - \cos(b^*, a^*) + \cos(b^*, a)) \tag{5.2}$$

Cela signifie que la résolution des questions d'analogies avec l'arithmétique vectorielle est mathématiquement équivalente à chercher un mot $b^*$ qui est similaire à $b$ et $a^*$ mais différent de $a$. La similarité relationnelle est donc exprimée sous la forme d'une somme de similarités attributionnelles.

Dans l'équation 5.2, il y a deux termes à maximiser et un à minimiser. Une propriété connue de tels objectifs linéaires est qu'ils permettent à un terme suffisamment

important de dominer dans l'expression. Ce comportement est problématique dans ce test d'analogie parce que chaque terme reflète un aspect différent de similarité, et les différents aspects ont des échelles différentes. Par exemple, le "roi" est plus "royal" que "masculin", et il éclipsera donc l'aspect genre de l'analogie.

Pour atteindre un meilleur équilibre entre les différents aspects de similarité, [Levy *et al.*, 2014] ont proposé de changer l'équation $3COSADD$ et de passer d'une combinaison additive à une combinaison multiplicative, formulée par l'équation suivante :

$$3COSMUL = \arg_{b^* \in V} \frac{\cos(b^*, b) \cos(b^*, a^*)}{\cos(b^*, a) + \varepsilon} \tag{5.3}$$

où $\varepsilon = 0.001$ est utilisé pour empêcher la division par zéro.

Ceci est équivalent à prendre le logarithme de chaque terme avant de faire l'addition, ce qui amplifie les différences entre les petites quantités et réduit les différences entre les plus grandes quantités.

#### 5.1.1.3 Tâche de similarité

La dernière tâche que nous proposons, permet d'évaluer les similarités attributionnelles entre les paires de mots captées par les *word embeddings*.

Le but de cette tâche est de mesurer à la fois la similarité fonctionnelle (*genuine similarity* dans [Kiela *et al.*, 2015] ) et la similarité associative (*relatedness*) entre les paires de mots. Les auteurs dans [Gracia et Mena, 2008] ont défini ces deux relations comme suit :

— Similarité fonctionnelle : elle est généralement définie en tenant compte des relations lexicales de synonymie (par exemple "voiture : automobile") et hyponymie : la signification d'un mot est couverte par un autre terme plus général, comme dans "voiture : véhicule".

— Similarité associative : elle couvre tout type d'association lexicale ou fonctionnelle. Les mots peuvent être associés par de nombreuses autres relations, comme la méronymie [4] (partie de relation, comme dans "doigt : main"), antonymie (sens opposés, comme "chaud : froid"), ou tout autre type de relation fonctionnelle ou association fréquente (par exemple, "pingouin : antarctique", qui ne sont pas liés par une relation lexicale).

Cette tâche d'évaluation est largement utilisée dans la littérature [Levy et Goldberg, 2014, Pennington *et al.*, 2014, Ji *et al.*, 2015, Levy *et al.*, 2015]. Supposons que nous disposons d'un jeu de données composé de paires de mots et d'un score de similarité attribué par des humains (score de référence) pour chaque paire. La tâche consiste à comparer les similarités entre les paires de mots à leur score de référence. D'abord, nous calculons la similarité cosinus entre les *word embeddings* de chaque paire de mots. Puis, nous ordonnons ces paires par leur

4. La méronymie est une relation sémantique entre mots, lorsqu'un terme désigne une partie d'un second terme. Par exemple, "bras" est un méronyme de "corps", de même que "toit" est un méronyme de "maison".

degré de similarité. Enfin, nous calculons le coefficient de corrélation *Spearman's rank* $\rho$[5] [Gauthier, 2001] entre les classements produits par les *word embeddings* et ceux produits par les scores de référence.

### 5.1.2 Résultats d'évaluation

#### 5.1.2.1 Protocole expérimental

Les *words embeddings* évalués sont entrainés sur le corpus *Gigaword* composé de plus de 4 milliards de mots. Ce corpus est annoté par des analyses de dépendances en utilisant la boite à outils MACAON [Nasr *et al.*, 2011]. La version annotée est utilisée pour l'apprentissage de *w2vf-deps*, tandis que la version non annotée est utilisée pour l'apprentissage de *CBOW*, *Skip-gram* et *GloVe*. Les mots qui apparaissent moins de 100 fois sont supprimés du corpus, ce qui donne un vocabulaire de 239K mots.

Les auteurs dans [Levy *et al.*, 2015] ont montré que la performance des *word embeddings* dépend non seulement de l'approche mais aussi du jeu de paramètres utilisé pour l'apprentissage. C'est pour cette raison que notre choix de paramètres s'est appuyé sur les études précédentes [Mikolov *et al.*, 2013b, Levy et Goldberg, 2014, Levy *et al.*, 2015]. Les paramètres choisis sont résumés dans le tableau 5.1.

| Embeddings | Tail. | Dim. | Neg. |
|---|---|---|---|
| CBOW | 5 | 200 | 5 |
| Skip-gram | 5 | 200 | 5 |
| GloVe | 5 | 200 | - |
| w2vf-deps | - | 200 | 5 |

TABLE 5.1 – Les paramètres utilisés pour l'apprentissage des *word embeddings* : taille de la fenêtre, dimension et négative sampling.

En ce qui concerne les tâches d'évaluation, nous détaillons dans les sections suivantes les jeux de données ainsi que les paramètres utilisés pour chacune des tâches avant de présenter les résultats.

#### 5.1.2.2 Tâches classiques du TALN

Dans chacune des tâches TALN, une étiquette doit être prédite pour chaque mot. Cette prédiction est faite en analysant chaque mot dans son contexte. Nous avons utilisé l'architecture neuronale *MLP-MS* proposée dans le chapitre 4. Celle-ci prend en entrée un vecteur de 1000 dimensions. Ce vecteur correspond à la concaténation des *word embeddings* d'un 5-gramme centré sur le mot pour lequel la prédiction est effectuée. Si un *word embedding* n'existe pas pour chacun des mots, il est remplacé par 0. Les mots en dehors des frontières des phrases sont également remplacés par 0.

*MLP-MS* est composé de deux couches cachées. La première couche est la concaténation de trois sous-couches : $H^1_G$, $H^1_W$ et $H^1_D$ ayant respectivement 300, 100 et 300

5. http://grasland.script.univ-paris-diderot.fr/STAT98/stat98_6/stat98_6.htm

unités cachées. La deuxième couche a 300 unités cachées. Les fonctions d'activation sont ReLu pour les premières couches et tanh pour la deuxième. Les hyperparamètres : taux d'apprentissage et taille du mini-lot sont optimisés pour chaque *word embedding* sur les corpus de développement (Dev) disponibles pour chaque tâche.

La répartition des données des quatre tâches TALN est présentée dans le tableau 5.2.

| Tâche | Benchmark | Train | Dev | Test |
|---|---|---|---|---|
| POS | Penn Treebank | 958K | 34K | 58K |
| CHK | CoNLL 2000 | 191K | 21K | 47K |
| NER | CoNLL 2003 | 205K | 52K | 47K |
| MD | Ontonotes | 736K | 102K | 105K |

TABLE 5.2 – Répartition des données pour chaque tâche TALN sur les corpus Train, Dev et Test en précisant le nombre de mots dans chaque corpus.

En ce qui concerne l'évaluation, les tâches CHK, NER et MD sont évaluées en terme de F-mesure (F). Alors que, la tâche POS est évaluée en calculant le taux d'exactitude (Acc.) de mots correctement étiquetés. Pour l'évaluation, nous avons utilisé le script *conlleval*[6].

Les résultats expérimentaux sont résumés dans le tableau 5.3. On observe que l'*embedding w2vf-deps* obtient les meilleurs résultats dans toutes les tâches. Ce résultat correspond à celui du chapitre 4 (tableau 4.2) sur la tâche de détection d'erreurs.

La bonne performance de *w2vf-deps* est liée d'une part, à l'utilisation d'un corpus annoté enrichi par des indices syntaxiques, et d'autre part, à la large taille du contexte qui est extrait en s'appuyant sur des relations syntaxiques. Les autres *embeddings* sont eux appris sur un corpus non annoté avec une fenêtre de contexte de taille réduite. Malgré sa performance, l'estimation de *w2vf-deps* est contrainte par la disponibilité des données annotées et des analyseurs de dépendance. Ceci n'est pas toujours possible, par exemple pour les langues peu dotées.

En ce qui concerne les résultats des *word embeddings* entrainés sur le corpus non annoté, on remarque que *Skip-gram* obtient des résultats meilleurs que *CBOW* et *GloVe* dans les tâches POS et MD. Par contre, pour les deux autres tâches *CBOW* obtient les meilleurs résultats.

#### 5.1.2.3 Tâche d'analogie

Pour cette tâche nous avons utilisé le jeu de données ainsi que l'outil[7] fourni par [Mikolov *et al.*, 2013c]. Ce jeu de données est composé de cinq types de questions sémantiques comme : *Capital-cities* "Athens : Greece → Tehran :?" et *family* "boy : girl → brother :?", et de neuf types de questions syntaxiques comme : *adjective-to-adverb* "amazing : amazingly → calm :?" et *comparative* "bad : worse → big :?". Au total, il y a 8869 questions sémantiques

6. https://github.com/MvanErp/NER/blob/master/Scripts/conlleval.pl
7. https://code.google.com/archive/p/word2vec/source/default/source

| Embeddings | POS Acc. | CHK F | NER F | MD F |
|---|---|---|---|---|
| CBOW | 96,01 | 0,905 | 0,783 | 0,554 |
| Skip-gram | 96,43 | 0,896 | 0,776 | 0,578 |
| GloVe | 95,79 | 0,869 | 0,764 | 0,545 |
| w2vf-deps | **96,66** | **0,920** | **0,793** | **0,580** |

TABLE 5.3 – Performance des *word embeddings* sur le corpus Test, dans les quatre tâches classiques du TALN.

et 10675 questions syntaxiques. Afin de ne pas pénaliser le calcul du taux d'exactitude, nous avons nettoyé ces questions en ne gardant que les mots des questions qui existent dans le vocabulaire, ce qui donne au total 19422 questions.

Le tableau 5.4 présente en détail le taux d'exactitude obtenu par type de questions (sémantique (les cinq premiers) et syntaxiques) pour chacun des *word embeddings*.

| Sous tâches | CBOW | Skip-gram | GloVe | w2vf-deps |
|---|---|---|---|---|
| Capital cities | 89,5 | 88,3 | 93,1 | 71,5 |
| Capital-world | 81,0 | 88,2 | 92,2 | 34,4 |
| Currency | 9,5 | 17,6 | 16,6 | 8,8 |
| City-in-state | 22,9 | 27,2 | 36,2 | 6,2 |
| Family | 86,8 | 76,5 | 81,4 | 74,3 |
| Adjective-to-adverb | 13,3 | 18,2 | 22,3 | 5,3 |
| Opposite | 24,9 | 34,1 | 22,5 | 36,9 |
| Comparative | 81,4 | 79,3 | 84,8 | 87,6 |
| Superlative | 61,2 | 69,4 | 65,0 | 71,5 |
| Present-participle | 62,6 | 65,3 | 66,7 | 60,1 |
| Nationality-adjective | 81,1 | 86,7 | 91,2 | 25,8 |
| Past-tense | 55,1 | 56,7 | 59,2 | 55,9 |
| Plural | 54,0 | 55,0 | 69,8 | 59,9 |
| Plural-verbs | 37,8 | 61,8 | 48,4 | 86,8 |
| Sémantique Acc. | 58,8 | 63,7 | **68,8** | 28,5 |
| Syntaxique Acc. | 53,2 | 57,5 | **58,7** | 54,3 |
| Acc. totale | 57,2 | 62,3 | **65,5** | 42,7 |

TABLE 5.4 – Performances des *word embeddings* sur la tâche d'analogie en terme de taux d'exactitude (Acc.), détaillées par type de questions, en utilisant la méthode $3COSADD$.

Dans ces expériences, la méthode $3COSADD$ est utilisée pour trouver la bonne réponse aux questions. On peut observer dans le tableau 5.4 que les différents *word embeddings* obtiennent une grande variété de taux d'exactitude sur cette tâche. On remarque que le classement des *word embeddings* obtenu dans les évaluations précédentes n'est pas conservé. *GloVe* est le meilleur dans cette tâche, suivi de *Skip-gram*

et de *CBOW*. Ces résultats sont en adéquation avec ceux de [Pennington *et al.*, 2014, Levy *et al.*, 2015]. D'autre part, *w2vf-deps* obtient de mauvais résultats sur cette tâche, ce qui confirme les résultats de [Levy et Goldberg, 2014].

Pour avoir une idée sur la métrique $3COSMUL$ proposée par Levy *et al.*, nous résumons dans le tableau 5.5 le taux d'exactitude total obtenu avec les deux métriques, sans avoir reporté le détail par type de questions. On remarque qu'avec les deux métriques on obtient le même classement. Par contre, il y a un peu de changement au niveau du taux d'exactitude. On voit bien que cette méthode n'a amélioré que le résultat de *w2vf-deps*. Cela s'explique par le fait que *w2vf-deps* capte mieux les relations de similarité fonctionnelle comme nous l'avons vu dans la section 1.2.4. Selon [Levy et Goldberg, 2014], l'application de cette méthode va aider à trouver le bon mot par exemple pour la question de type *Capital-cities* "London : England → Baghdad : ?", où on cherche à trouver le mot *Iraq* dont *Baghdad* est la capitale. En revanche $3COSADD$ proposera *Mosul* qui est le mot le plus proche de *Baghdad* en termes de distance cosinus car il s'agit également d'une grande ville irakienne.

| **Embeddings** | **3COSADD** | **3COSMUL** |
|---|---|---|
| CBOW | 57,2 | 56,5 |
| Skip-gram | 62,3 | 61,5 |
| GloVe | **65,5** | **63,9** |
| w2vf-deps | 42,7 | 44,2 |

TABLE 5.5 – Taux d'exactitude total obtenu sur la tâche d'analogie en utilisant les méthodes $3COSADD$ et $3COSMUL$

#### 5.1.2.4 Tâche de similarité

Dans la littérature, plusieurs jeux d'évaluation ont été proposés pour mesurer la similarité entre les mots. Les auteurs dans [Faruqui et Dyer, 2014] ont détaillé les jeux d'évaluation les plus couramment utilisés. Le nombre d'échantillons dans ces jeux varie entre 30 et 3000. Pour nos expériences, nous avons sélectionné les jeux qui ont au minimum 300 échantillons afin de garantir l'efficacité de l'évaluation.

1. Le premier benchmark est **WS-353** [8] [Finkelstein *et al.*, 2001]. Il se compose de 153 paires de mots de similarité fonctionnelle (**SIM**) et 200 paires de mots de similarité associative (**REL**), soit un total de 353 paires (**ALL**). Chaque paire est associée à un score de similarité variant entre 0 et 10 attribué par des humains.
2. Le deuxième benchmark, **MTurk** [9][Halawi *et al.*, 2012], est constitué de 771 paires de mots de similarité associative. Ce jeu est différent de **WS-353**, du fait qu'il a été construit par externalisation ouverte à partir d'évaluations

8. http://www.cs.technion.ac.il/ gabr/resources/data/wordsim353/
9. http://www2.mta.ac.il/ gideon/mturk771.html

humaines de similarité obtenues via la plateforme *Amazon Mechanical Turk*[10] (AMT).

3. **MEN**[11] [Bruni *et al.*, 2014] est un autre benchmark AMT, composé de 3000 paires de mots. Ces paires sont sélectionnées aléatoirement à partir des mots qui apparaissent au moins 700 fois dans le corpus *ukWaC and Wackypedia corpora combined*[12], et au moins 50 fois dans le corpus *ESP game dataset*[13]. Dans ce jeu d'évaluation, la distribution des niveaux de similarité associative entre les paires est équilibrée.
4. Le dernier benchmark **Rare-Word (RW)**[14] [Luong *et al.*, 2013] est composé de 2034 paires de mots rares en échantillonnant des mots ayant différents niveaux de fréquence, au contraire de nos autres benchmarks qui contiennent les mots les plus fréquents. Les auteurs ont utilisé également l'AMT pour collecter dix scores de similarités attribués par des humains pour chaque paire de mots, portant une similarité associative.

La performance des *word embeddings* sur les différents benchmarks est résumée dans le tableau 5.6.

| Embeddings | WS-353 | | | MTurk | MEN | RW |
|---|---|---|---|---|---|---|
| | SIM | REL | ALL | | | |
| CBOW | 0,662 | 0,490 | **0,590** | 0,517 | 0,609 | 0,465 |
| Skip-gram | 0,627 | 0,500 | 0,558 | 0,578 | **0,662** | **0,502** |
| GloVe | 0,533 | **0,511** | 0,533 | **0,584** | 0,660 | 0,410 |
| w2vf-deps | **0,705** | 0,370 | 0,523 | 0,538 | 0,557 | 0,435 |

TABLE 5.6 – Performances des *word embeddings* sur la tâche de similarité, évaluées en terme de Spearman's $\rho$.

Les résultats dans cette tâche sont en faveur de Skip-gram, il obtient les meilleurs résultats dans la plupart des tâches. En ce qui concerne *w2vf-deps*, les résultats obtenus confirment les constatations de la section précédente. En effet, *w2vf-deps* obtient le meilleur résultat dans **SIM** qui ne contient que des relations de similarité fonctionnelle, alors qu'il obtient les résultats les plus faibles pour les autres benchmarks (similarité associative), à l'exception de **RW**.

### 5.1.3 Discussions

Les résultats d'évaluation des *word embeddings* dans les trois tâches TALN, analogie et similarité montrent que ces *embeddings* portent des informations différentes, chacun d'eux étant le meilleur dans l'une des trois tâches. Ces résultats renforcent notre intuition de complémentarité entre les différents *word embeddings*, et notre

10. https://fr.wikipedia.org/wiki/Amazon_Mechanical_Turk
11. http://clic.cimec.unitn.it/ elia.bruni/MEN.html
12. http://wacky.sslmit.unibo.it/doku.php
13. https://en.wikipedia.org/wiki/ESP_game
14. http://www-nlp.stanford.edu/ lmthang/morphoNLM/

volonté de construire un embedding plus robuste qui pourrait obtenir de bonnes performances, quelle que soit la tâche. Pour atteindre cet objectif, nous proposons dans la section suivante une étude sur la combinaison des *word embeddings*.

## 5.2 Combinaison de word embeddings

Les résultats d'évaluation présentés dans la section 5.1.1 ainsi que l'étude précédente de [Turian *et al.*, 2010], ont motivé la deuxième contribution de ce chapitre, abordant la combinaison des *word embeddings*.

Dans cette section, nous présentons les différentes approches de combinaison ainsi que les résultats d'évaluation des *words embeddings* combinés sur les tâches classiques du TALN, et les tâches d'analogie et de similarité.

### 5.2.1 Approches de combinaison

L'étude présentée par Turian *et al.* [Turian *et al.*, 2010] a montré que la combinaison des *word embeddings* améliorait les performances dans les tâches de reconnaissance d'entités nommées et de regroupement en syntagme. Les auteurs ont combiné par concaténation différents types de représentations de mots notamment : les *embeddings* de Collobert et Weston [Collobert et Weston, 2008], des *embeddings* basés sur le modèle hiérarchique log-bilinéaire (*Hierarchical Log-BiLinear* (HLBL)) [Mnih et Hinton, 2007] et les clusters de Brown [Brown *et al.*, 1992].

Dans notre étude, nous proposons plusieurs approches de combinaison : la concaténation, l'analyse en composantes principales (ACP) et l'auto-encodeur.

Dans ce qui suit, nous allons utiliser les notations suivantes : soient $d$ la dimension de l'*embedding* combiné obtenu par concaténation, et $k$ la dimension de l'*embedding* combiné obtenu par ACP ou auto-encodeur.

#### 5.2.1.1 Concatenation simple

La première approche, appelé *Concat* dans ce qui suit, consiste à concaténer les différents *word embeddings* comme dans [Turian *et al.*, 2010]. Chaque mot est représenté par un vecteur de $d$-dimensions.

#### 5.2.1.2 Analyse en Composantes Principales

La deuxième approche consiste à combiner par application d'une ACP, une méthode classique en analyse de données. Son objectif est de réduire l'espace des dimensions tout en déformant le moins possible la réalité. Il s'agit donc d'obtenir le résumé le plus pertinent des données initiales.

Mathématiquement, l'ACP est un simple changement de base (on l'appelle aussi translation). Elle consiste à passer d'une représentation dans la base des variables initiales à une représentation dans la base des facteurs définis par les vecteurs propres de la matrice de corrélation.

En pratique, on applique l'ACP sur l'*embedding* obtenu par concaténation, noté $Emb_{Concat}$, afin de réduire le nombre de dimensions et obtenir un *word embedding* de $k$-dimensions.
Soit la matrice $M$ composée de $n$ lignes et $d$ colonnes correspondant respectivement au nombre de mots et à la dimension du vecteur $Emb_{Concat}$. La première étape consiste à centrer-réduire la matrice $M$ avec la méthode *Z-scores* pour pouvoir calculer la matrice de covariance. Ensuite, la matrice de corrélation est calculée en utilisant la matrice de covariance normalisée. Puis, l'ACP est calculée pour obtenir le nouveau système vectoriel $V$ en utilisant la matrice de corrélation. Enfin, l'embedding combiné $Emb_{ACP}$ est obtenu en projetant les données sur la nouvelle base et ne considérant que les $k$ premières composantes ($k < d$), comme suit :

$$Emb_{ACP} = M \times V_k \tag{5.4}$$

Ce processus est illustré dans la figure 5.2.

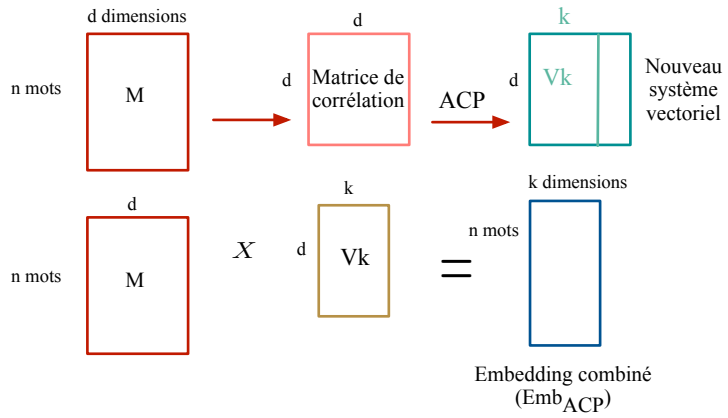



FIGURE 5.2 – Processus de combinaison des *words embeddings* au moyen d'une ACP.

#### 5.2.1.3 Auto-encodeur

La dernière approche porte sur l'utilisation d'un auto-encodeur *AutoE* pour la combinaison des words embeddings. L'auto-encodeur est une généralisation non linéaire de l'ACP. Comme nous avons vu dans la section 1.1.2.4, il utilise un encodeur pour transformer des données de grande dimension en une représentation de faible dimension et un décodeur pour reconstruire les données à partir de cette représentation.

À notre connaissance, les auto-encodeurs ne sont pas utilisés pour la combinaison, mais servent généralement à apprendre une représentation compressée et distribuée pour un ensemble de données, généralement dans le but de réduire la

dimensionalité.

Pratiquement, nous avons appliqué l'auto-encodeur sur l'*embedding* $Emb_{Concat}$ comme le montre la figure 5.3 afin d'obtenir l'*embedding* $Emb_{AutoE}$. Contrairement à l'ACP, une transformation non linéaire est appliquée à $Emb_{Concat}$. La représentation résultante de cette transformation est susceptible de préserver les informations utiles et non redondantes de $Emb_{Concat}$.

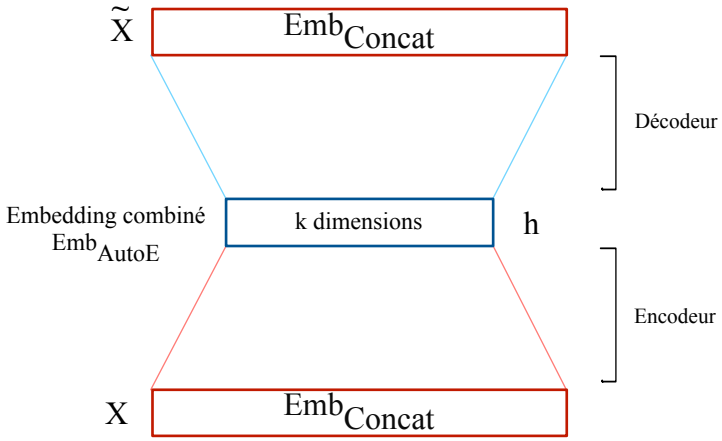



FIGURE 5.3 – Illustration de l'architecture d'auto-encodeur. $X$ représente l'entrée, $h$ la couche de projection générée par l'encodeur alors que $\tilde{X}$ définit le vecteur reconstruit par le décodeur.

### 5.2.2 Performances des word embeddings combinés

Nous proposons de combiner deux ensembles différents de *word embbedings* : Simple = {*CBOW*, *Skip-gram*, *GloVe*} contient les *embeddings* qui ne nécessitent pas de données étiquetées durant l'apprentissage, Best = {*w2vf-deps*, *Skip-gram*, *GloVe*} contient ceux qui ont obtenu les meilleurs résultats dans les tâches d'évaluation. Considérant ces deux ensembles et les trois types de combinaison, nous obtenons ainsi 6 ensembles combinés : $Simple_{Concat}$ et $Best_{Concat}$ de dimensions 600, $Simple_{ACP}$ et $Best_{ACP}$, $Simple_{AutoE}$ et $Best_{AutoE}$ de dimensions 200.

Ils sont évalués sur les trois tâches d'évaluation, en comparant leurs performances aux *embeddings* individuels qu'ils combinent. Dans les sections suivantes, les améliorations obtenues par les *embeddings* combinés en comparaison aux *embeddings* individuels sont indiquées en gras.

#### 5.2.2.1 Tâches classiques du TALN

Les résultats d'évaluation des 6 *word embeddings* combinés sur les différentes tâches classiques du TALN sont résumés dans le tableau 5.7.

On remarque que la combinaison des *word embeddings* est performante et permet d'obtenir des améliorations dans les tâches CHK, NER et MD. De plus, pour la tâche POS les *embeddings* $Best_{Concat}$ et $Best_{AutoE}$ montrent des performances similaires au meilleur embedding non combiné sur cette tâche (*w2vf-deps* 96,66%). Enfin, l'importance de la présence de *w2vf-deps* dans l'ensemble des *embeddings*combinés est mise en évidence par la performance toujours moindre de tous les *embeddings* combinés de l'ensemble Simple.

| Dim. | Embeddings | POS Acc. | CHK F | NER F | MD F |
|---|---|---|---|---|---|
| Simple : CBOW Skip-gram GloVe | | | | | |
| 600 | $Simple_{Concat}$ | 96,24 | **0,912** | **0,794** | **0,578** |
| 200 | $Simple_{ACP}$ | 96,39 | 0,902 | **0,789** | 0,577 |
| | $Simple_{AutoE}$ | 95,99 | 0,895 | **0,784** | 0,577 |
| Best : w2vf-deps Skip-gram GloVe | | | | | |
| 600 | $Best_{Concat}$ | 96,67 | 0,918 | **0,810** | **0,582** |
| 200 | $Best_{ACP}$ | 96,45 | 0,901 | **0,796** | **0,602** |
| | $Best_{AutoE}$ | 96,64 | 0,913 | **0,804** | **0,603** |

TABLE 5.7 – Performance des *embeddings* combinés sur le corpus Test, dans les quatre tâches classiques du TALN.

#### 5.2.2.2 Tâche d'analogie

Les résultats d'évaluation pour la tâche d'analogie sont présentés dans le tableau 5.8 qui résume le taux d'exactitude détaillé par type de questions, en terme d'analogie sémantique ou syntaxique et finalement le score global.

Les résultats montrent que la combinaison des *embeddings* Simple n'est ici pas avantageuse, elle n'apporte pas une amélioration par rapport au meilleur *word embedding* non combiné (*GloVe* 65,5%).

En revanche, de meilleurs résultats sont obtenus avec la combinaison de type *Concat* ou *ACP* des *embeddings Best* ($Best_{Concat}$ et $Best_{ACP}$). Ces *embeddings* obtiennent respectivement 71,4% et 70,7% de taux d'exactitude global. Néanmoins, ce n'est pas le cas pour l'*embedding* combiné avec un auto-encodeur qui obtient les résultats les plus faibles.

#### 5.2.2.3 Tâche de similarité

Les résultats de cette tâche sont à nouveau en faveur de la combinaison des *embeddings Best*, comme illustré dans le tableau 5.9. Les *embeddings* $Simple_{Concat}$ et $Simple_{ACP}$ améliorent les résultats de deux tâches : WS-353 (*ALL*) et MEN. Les *embeddings* $Best_{Concat}$ et $Best_{ACP}$ améliorent eux les résultats dans les tâches WS-353 (*ALL*), MTurk et MEN.

| **Dim.** | 600 | 200 | | 600 | 200 | |
|---|---|---|---|---|---|---|
| **Sous tâches** | $Simple_{Concat}$ | $Simple_{ACP}$ | $Simple_{AutoE}$ | $Best_{Concat}$ | $Best_{ACP}$ | $Best_{AutoE}$ |
| Capital cities | 92,9 | 92,7 | 90,1 | 94,7 | 94,7 | 90,3 |
| Capital-world | 89,0 | 89,6 | 82,4 | 93,8 | 95,3 | 85,5 |
| Currency | 14,1 | 14,4 | 9,4 | 23,5 | 23,9 | 18,3 |
| City-in-state | 30,6 | 34,9 | 33,6 | 40,0 | 39,9 | 24,6 |
| Family | 89,3 | 81,2 | 82,6 | 88,9 | 85,4 | 87,9 |
| Adj-to-adv | 17,4 | 12,3 | 11,7 | 25,2 | 20,9 | 14,5 |
| Opposite | 28,3 | 24,8 | 18,7 | 37,9 | 29,7 | 24,3 |
| Comparative | 84,6 | 81,6 | 80,6 | 91,2 | 90,0 | 83,6 |
| Superlative | 66,9 | 66,2 | 52,0 | 81,6 | 78,4 | 60,3 |
| Present-participle | 66,4 | 62,9 | 60,7 | 79,1 | 78,1 | 64,3 |
| Nationality-adj | 84,1 | 86,8 | 82,8 | 88,7 | 89,1 | 82,4 |
| Past-tense | 58,1 | 59,9 | 50,1 | 69,6 | 63,7 | 59,9 |
| Plural | 64,7 | 61,9 | 51,1 | 76,5 | 69,1 | 67,0 |
| Plural-verbs | 41,6 | 43,9 | 36,4 | 80,3 | 79,9 | 71,9 |
| Sémantique Acc. | 65,8 | 66,9 | 59,6 | **70,0** | **72,5** | 62,5 |
| Syntaxique Acc. | 57,4 | 55,4 | 50,2 | **72,6** | **69,2** | **61,7** |
| Acc. totale | 62,9 | 62,8 | 56,0 | **71,4** | **70,7** | 62,0 |

TABLE 5.8 – Performances des *embeddings* combinés sur la tâche d'analogie en termes de taux d'exactitude, en utilisant la méthode $3COSADD$.

| **Dim.** | **Embeddings** | **WS353** | | | **MTurk** | **MEN** | **RW** |
|---|---|---|---|---|---|---|---|
| | | **SIM** | **REL** | **ALL** | | | |
| Simple : Cbow-Skip-GloVe | | | | | | | |
| 600 | $Simple_{Concat}$ | **0,666** | **0,524** | **0,602** | 0,548 | 0,650 | 0,480 |
| 200 | $Simple_{ACP}$ | **0,656** | 0,489 | 0,574 | 0,534 | **0,669** | 0,496 |
| | $Simple_{AutoE}$ | **0,647** | 0,482 | 0,553 | 0,499 | 0,639 | 0,460 |
| Best : w2vf-deps-Skip-GloVe | | | | | | | |
| 600 | $Best_{Concat}$ | 0,652 | **0,533** | **0,570** | **0,613** | **0,694** | 0,486 |
| 200 | $Best_{ACP}$ | 0,666 | **0,517** | **0,579** | **0,618** | **0,713** | 0,495 |
| | $Best_{AutoE}$ | 0,611 | **0,512** | 0,558 | 0,578 | 0,649 | 0,446 |

TABLE 5.9 – Performance des *embeddings* combinés sur la tâche de similarité, évalué en terme de Spearman's $\rho$.

Malheureusement, comme dans la tâche d'analogie, les *embeddings* combinés avec auto-encodeur obtiennent les résultats les plus faibles. Ces tâches sont évaluées en utilisant la similarité cosinus : il semble que la transformation appliquée à ces *embeddings* durant la combinaison avec auto-encodeur ne préserve pas la structure linéaire des *embeddings* qui permet aux translations de représenter les propriétés linguistiques et sémantiques.

#### 5.2.2.4 Tâche de détection d'erreurs

Les résultats décrits dans les sections précédentes ont montré la performance des *embeddings* combinés, du fait qu'ils ont apporté des améliorations dans la plupart des tâches d'évaluation. C'est pourquoi nous nous sommes intéressés dans cette section à l'évaluation des *embeddings* combinés sur la tâche de détection d'erreurs. Pour faire cette évaluation, nous avons combiné les trois *embeddings* (Best = {*w2vf-deps*, *Skip-gram*, *GloVe*}), utilisés dans le chapitre 4 (en français) qui ont obtenu les meilleurs résultats dans les tâches d'évaluation. Pour ces expériences, nous avons utilisé le système de détection d'erreur *MLP-MS* qui intègre le jeu de descripteurs décrit dans la section 4.1.2. Les *embeddings* combinés sont utilisés en entrée pour représenter les mots. La significativité des résultats est mesurée en utilisant l'intervalle de confiance de 95% fondé sur la loi de Student.

Les résultats de l'évaluation sont résumés dans les tableaux 5.10 pour le corpus de développement et 5.11 pour le test.

| Approche | Représentation de mot | Label erreur | | | Global |
|---|---|---|---|---|---|
| | | P | R | F | CER |
| CRF (baseline) | discrète | 0,681 | 0,554 | 0,611 | 10,38 |
| Neuronale avec word embed. ling. non combinés | w2vf-deps | 0,730 | 0,502 | 0,595 | 10,06 |
| | Skip-gram | **0,751** | 0,456 | 0,567 | 10,24 |
| | GloVe | 0,738 | 0,469 | 0,574 | 10,26 |
| Neuronale avec word embed. ling. combinés | $Best_{Concat}$ | 0,704 | 0,554 | 0,620 | 9,99 |
| | $Best_{ACP}$ | 0,704 | 0,567 | 0,628 | 9,89 |
| | $Best_{AutoE}$ | 0,705 | **0,576** | **0,634** | **9,79** |

TABLE 5.10 – Comparaison de l'utilisation de différents types de *word embeddings* dans le système de détection d'erreurs MLP-MS sur Dev

On remarque que les *embeddings* combinés améliorent également les résultats par rapport aux *embeddings* non combinés dans cette tâche. Ces améliorations sont observées sur tous les *embeddings* combinés tant sur Dev que sur Test. L'embedding $Best_{AutoE}$ obtient les meilleurs résultats, suivi de $Best_{ACP}$ et de $Best_{Concat}$. L'*embedding* $Best_{AutoE}$ obtient respectivement $2,6\%$ et $2,3\%$ de réduction relative en CER sur Dev et Test par rapport à *w2vf-deps* qui est le meilleur *word embedding* non combiné. Une réduction significative du CER par rapport à celui des CRF est également atteinte : $5,68\%$ et $5,72\%$ respectivement sur Dev et Test.

| Approche | Représentation de mot | Label erreur | | | Global |
|---|---|---|---|---|---|
| | | P | R | F | CER |
| CRF (baseline) | discrète | 0,677 | 0,547 | 0,605 | 8,56 |
| Neuronale avec word embed. ling. non combinés | w2vf-deps | 0,719 | 0,510 | 0,596 | 8,26 |
| | Skip-gram | **0,745** | 0,468 | 0,574 | 8,30 |
| | GloVe | 0,722 | 0,470 | 0,569 | 8,53 |
| Neuronale avec word embed. ling. combinés | $Best_{Concat}$ | 0,688 | 0,558 | 0,616 | 8,34 |
| | $Best_{ACP}$ | 0,695 | 0,571 | 0,627 | 8,14 |
| | $Best_{AutoE}$ | 0,696 | **0,579** | **0,632** | **8,07** |

TABLE 5.11 – Comparaison de l'utilisation de différents types de *word embeddings* dans le système de détection d'erreurs MLP-MS sur Test

On observe ainsi que les meilleurs résultats dans cette tâche sont obtenus par *w2vf-deps* et $Best_{AutoE}$. Ces *word embeddings* ont obtenus les résultats les plus bas sur les tâches d'analogie et de similarité comme présenté dans les sections précédentes. Par conséquent, il semble que la tâche de détection d'erreurs nécessite plus d'informations syntaxiques que sémantiques, en particulier lorsque la taille du contexte est limitée à seulement deux mots de chaque côté. En effet, il est difficile de capturer des anomalies sémantiques entre les mots dans une petite fenêtre contextuelle : de telles informations devraient être capturées dans un contexte plus large.

## 5.3 Conclusion

Dans ce chapitre, nous avons effectué dans un premier temps une évaluation rigoureuse de la performance de différents types de *word embeddings* : *Skip-gram*, *CBOW*, *GloVe* et *w2vf-deps* sur les tâches classiques du TALN, et les tâches d'analogie et de similarité. Ces évaluations ont permis d'étudier la complémentarité des *word embeddings* constatée lors de leurs évaluations sur la tâche de détection d'erreurs (chapitre 4). Les résultats de l'évaluation rapportés dans ce document sont en adéquation et complètent ceux de la littérature. Ils renforcent l'intuition de complémentarité de ces *word embeddings*.

Ainsi, dans un second temps, nous avons proposé plusieurs approches de combinaisons : la concaténation, l'analyse en composante principale et l'auto-encodeur. Les *embeddings* combinés sont ensuite évalués sur les tâches classiques du TALN, et les tâches d'analogie et de similarité et comparés aux *embeddings* non combinés qui les composent. Nous avons constaté que la combinaison de *Skip-gram*, *GloVe* et *w2vf-deps* obtient les meilleurs résultats dans la plupart des tâches.

La performance des *embeddings* combinés est résumée dans la figure 5.4, ici nous avons comparé les résultats des *embeddings* combinés à ceux des meilleurs *word embeddings* non combinés dans chaque tâche. Les améliorations sont en gras et

colorées dans le tableau. On remarque que $Best_{Concat}$ a amélioré les résultats dans deux tâches TALN, trois tâches de similarité et la tâche d'analogie. Une amélioration équivalente est observée avec $Best_{ACP}$ avec des résultats encore meilleurs dans les trois tâches de similarité et la tâche TALN (MD).

| Embeddings Combinés | Tâches TALN | | | | Tâche de similarité | | | | Tâche d'analogie |
|---|---|---|---|---|---|---|---|---|---|
| | POS | CHK | NER | MD | WS353 | MTurk | RW | MEN | |
| | Acc, | F | | | Spearman's rank rho | | | | Acc, |
| $Best_{Concat}$ | 96,67 | 0,918 | **0,810** | **0,582** | **0,570** | **0,613** | 0,486 | **0,694** | **71,4** |
| $Best_{ACP}$ | 96,45 | 0,901 | **0,796** | **0,602** | **0,579** | **0,618** | 0,495 | **0,713** | **70,7** |
| $Best_{AutoE}$ | 96,64 | 0,913 | **0,804** | **0,603** | 0,558 | 0,578 | 0,446 | 0,649 | 62,0 |
| Skip-gram | **96,66** | **0,920** | **0,793** | **0,580** | **0,558** | | **0,502** | **0,662** | |
| w2v-deps | | | | | | **0,584** | | | **65,5** |
| GloVe | | | | | | | | | |

FIGURE 5.4 – Résumé des résultats d'évaluation des *embeddings* combinés (Best) comparés au meilleur *word embeddings* sur chaque tâche

En outre, l'embedding $Best_{AutoE}$ a amélioré les résultats de deux tâches TALN et obtient de meilleurs résultats que $Best_{ACP}$ dans les tâches NER et MD, et que $Best_{Concat}$ dans la tâche MD. Néanmoins, il est intéressant de noter que $Best_{AutoE}$ obtient les résultats les plus faibles dans les deux tâches d'analogie et de similarité (évaluées par la méthode de similarité cosinus). Nous supposons que cette contre-performance s'explique par le fait que la transformation non linéaire appliquée par l'auto-encodeur sur les *word embeddings* présentés en entrée ne préserve pas leur structure linéaire qui permet à l'opération mathématique de translation de représenter les propriétés linguistiques et sémantiques.

Il est intéressant de noter que les expériences faites pour évaluer les *word embeddings* et les *embeddings* combinés ont été effectuées sur la langue anglaise pour les tâches classiques du TALN, et celles d'analogie et de similarité. Les résultats obtenus sur la tâche de détection d'erreurs sont eux obtenus sur le français. Ainsi, notre étude révèle également que les résultats de l'évaluation des *embeddings* obtenus sur l'anglais semblent s'appliquer sur la langue française. Pour conclure, la combinaison des *word embeddings* s'avère donc performante dans différents types de tâches particulièrement la tâche de détection d'erreurs. Nous avons trouvé que dans cette tâche le meilleur résultat est obtenu par l'embedding combiné $Best_{AutoE}$.

Comme nous l'avons vu dans le chapitre 2 la génération de transcriptions automatiques de la parole s'appuie à la fois sur les informations linguistiques extraites des modèles de langage et sur les informations acoustiques extraites des modèles acoustiques. Pour détecter les erreurs dans ces transcriptions, nous souhaitons nous fonder sur ces deux types d'informations afin de mieux représenter le mot. Les *embeddings* présentés dans ce chapitre correspondent à la représentation linguistique d'un mot,

ils sont construits en s'appuyant sur des informations syntaxiques et contextuelles. Nous nous focalisons dans le chapitre suivant sur la représentation acoustique d'un mot et l'ajout d'informations extraites du signal de la parole.

CHAPITRE 6

# INTÉGRATION D'INFORMATIONS ACOUSTIQUES

## Sommaire

Dans ce chapitre, nous nous sommes intéressés à enrichir le système de détection d'erreurs proposé dans le chapitre 5 par des informations extraites du signal de parole, notamment des *embeddings* acoustiques *i.e. représentation continue de portion du signal de parole* et des informations prosodiques.

Nous nous intéressons dans un premier temps aux *embeddings* acoustiques (section 6.1) et commençons par présenter l'approche proposée pour leur construction. Celle-ci repose sur l'utilisation d'un réseau de neurones convolutif pour construire des *embeddings* acoustiques du signal pour chaque occurrence de mot dans le corpus d'apprentissage, et un réseau de neurones profond pour obtenir un seul *embedding* acoustique de mot. Ce dernier réseau permet également de construire des *embeddings* acoustiques pour les mots non observés dans le corpus d'apprentissage audio. Ensuite, nous présentons les tâches d'évaluations proposées pour évaluer la qualité des *embeddings* acoustiques de mots.

Dans un second temps, nous nous focalisons sur la tâche de détection d'erreurs, en exposant la performance des *embeddings* acoustiques ainsi que leur combinaison avec les descripteurs prosodiques notamment : la fréquence fondamentale, la durée de mot, le nombre et la durée moyenne de phonème, sur cette tâche (section 6.2) juste avant la conclusion (section 6.3).

## 6.1 Embeddings acoustiques

Des études récentes ont commencé à reconsidérer les unités lexicales (mots) comme unité de modélisation de base, au lieu des unités phonétiques (phonèmes), pour la reconnaissance automatique de la parole [Maas *et al.*, 2012, Bengio et Heigold, 2014]. Certains systèmes sont alors fondés sur l'utilisation de représentations vectorielles (*embeddings*) de dimension fixe de segments de parole correspondant à des mots. L'idée principale est de projeter les séquences acoustiques de longueur variable dans un espace de faible dimension de telle sorte que les mots qui se prononcent de la même manière sont projetés dans la même zone, tandis que les mots différents sont projetés dans des zones différentes. Ces *embeddings* sont utilisés dans d'autres tâches que la reconnaissance de la parole, on peut citer notamment la détection de mots clés (*spoken term detection*) [Fiscus *et al.*, 2007], la recherche par requête vocale (*spoken query-by-example search*) [Levin *et al.*, 2013, Anguera *et al.*, 2014, Kamper *et al.*, 2015], *etc.*

### 6.1.1 Construction

L'approche que nous avons utilisée pour construire des *embeddings* acoustiques s'inspire de celle proposée dans [Bengio et Heigold, 2014]. Les *embeddings* acoustiques sont construits en utilisant une architecture neuronale profonde, représentée dans la figure 6.1. Celle-ci s'appuie sur un réseau de neurones convolutif (CNN) et sur un réseau de neurones profond (DNN) entrainé en utilisant la fonction de coût *triplet ranking* [Weston *et al.*, 2011, Wang *et al.*, 2014, Bengio et Heigold, 2014]. Cette architecture a été proposée par [Bengio et Heigold, 2014] dans le but d'attribuer des

scores pour les mots non présents dans le corpus d'apprentissage, au moment du re-scoring d'un graphe de mots dans un système de reconnaissance de la parole.

Les deux architectures sont entrainées en utilisant différentes entrées : respectivement le signal de parole et la représentation orthographique du mot. Elles sont détaillées par la suite :

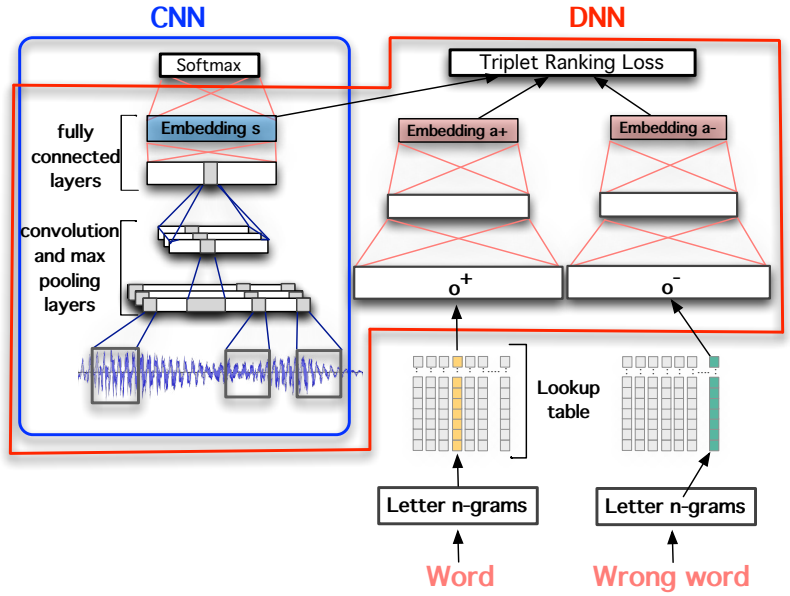



FIGURE 6.1 – Architecture profonde utilisée pour l'apprentissage des *embeddings* acoustiques.

#### 6.1.1.1 Embeddings acoustiques de signal ou d'occurrences de mots

Le CNN est entrainé pour prédire un mot sachant le signal de parole. Il est composé d'une suite de couches de convolution et de sous-échantillonnage, suivie de couches de neurones qui sont totalement connectées et qui alimentent la dernière couche *Softmax*. La couche précédant la couche *Softmax* est appelée couche de projection **s**. Elle contient une représentation compacte du signal acoustique qui a tendance à préserver la similarité acoustique entre les mots, de sorte que les mots sont proches dans cet espace s'ils ont une prononciation similaire.

Le CNN permet de construire des *embeddings* acoustiques pour chaque occurrence d'un mot dans le signal audio. Malheureusement, avec cette architecture on ne peut construire des *embeddings* acoustiques que pour les mots observés dans le corpus d'apprentissage audio. De plus, un mot peut être associé à plusieurs *embeddings* acoustiques de signal, parce qu'il peut avoir plusieurs productions sonores différentes. C'est pourquoi les auteurs dans [Bengio et Heigold, 2014] ont proposé une extension de cette architecture.

#### 6.1.1.2 Embeddings acoustiques de mots

Les *embeddings* acoustiques de mots $\mathbf{a}^+$ sont construits en utilisant un réseau de neurones profond (DNN). Ces *embeddings* sont construits en s'appuyant sur la représentation orthographique d'un mot, et peuvent être vus comme des représentations acoustiques canonique de mots. Leur construction s'appuie sur les *embeddings* de signal qui représentent des *embeddings* d'occurrences de mots observés dans le corpus d'apprentissage.

Comme dans [Bengio et Heigold, 2014], la représentation orthographique correspond à un sac de lettres de n-grammes. Ce sac est composé de 10222 tri-grammes, bi-grammes et uni-grammes extraits d'un vocabulaire de $52k$ mots, y compris les symboles spéciaux [ et ] pour spécifier le début et la fin d'un mot. Ensuite, nous avons proposé d'utiliser un auto-encodeur pour réduire la taille du vecteur de sac de n-grammes à $d$ dimensions. Il en résulte l'*embedding* orthographique $\mathbf{o}$ d'un mot. Pour vérifier la performance de cette représentation, un réseau de neurones est appris pour prédire un mot à partir de son *embedding* orthographique. Il atteint 99,99% de taux d'exactitude sur l'ensemble d'apprentissage composé de $52k$ mots du vocabulaire, cela montre la pertinence de cette représentation.

L' architecture du DNN est entrainée avec la fonction *triplet ranking* afin de projeter les *embeddings* orthographiques des mots dans l'espace des *embeddings* acoustiques $\mathbf{s}$. Nous rappelons que les *embeddings* $\mathbf{s}$ sont obtenus à partir de l'architecture CNN qui est apprise d'une manière indépendante. Pendant l'apprentissage, le DNN prend comme entrées un *embedding* acoustique de signal $\mathbf{s}$, l'*embedding* orthographique $\mathbf{o}^+$ du mot correspondant et l'*embedding* orthographique $\mathbf{o}^-$ d'un mot sélectionné aléatoirement et différent du premier mot. Ces deux représentations orthographiques sont fournies à deux instances du même réseau de neurones au cours de son apprentissage, afin d'obtenir une projection la plus discriminante possible, en utilisant la fonction de coût *triplet ranking*.

Soit le triplet $tr = (\mathbf{s}, \mathbf{a}^+, \mathbf{a}^-)$, où $\mathbf{s}$ est l'*embedding* acoustique de signal, $\mathbf{a}^+$ est l'*embedding* acoustique du mot obtenu par le DNN pour le mot correspondant, et $\mathbf{a}^-$ l'*embedding* du mot incorrect. La fonction de coût *triplet ranking* est définie comme suit :

$$Loss = \max(0, m - Sim_{dot}(s, a^+) + Sim_{dot}(s, a^-)) \tag{6.1}$$

où $Sim_{dot}(a, b)$ est la fonction produit scalaire utilisée pour calculer la similarité entre deux vecteurs $a$ et $b$, $m$ est un paramètre qui régularise la marge entre les deux paires de similarités $Sim_{dot}(s, a^+)$ et $Sim_{dot}(s, a^-)$. Cette fonction est pondérée par le rang du mot correct (prononcé) estimé par le CNN.

Le modèle résultant peut ensuite être utilisé pour construire un *embedding* acoustique $\mathbf{a}^+$ à partir de n'importe quel mot, tant qu'il est possible d'extraire une représentation orthographique de celui-ci. Plus clairement, nous pouvons dire que $\mathbf{a}^+$ est une projection des *embeddings* orthographiques $\mathbf{o}^+$ dans l'espace des *embeddings* acoustiques de signal $\mathbf{s}$.

### 6.1.2 Évaluation

#### 6.1.2.1 Tâches d'évaluation

De nombreuses études dans la littérature [Carlin *et al.*, 2011, Levin *et al.*, 2013, Kamper *et al.*, 2015] se sont intéressées à l'évaluation des *embeddings* acoustiques de signal $\mathbf{s}$. Étant donnée une paire de segments acoustiques, cette procédure d'évaluation consiste à déterminer si les segments de parole correspondent aux mêmes mots ou non. Cela peut être effectué de plusieurs façons : soit par l'utilisation de la déformation temporelle dynamique (*Dynamic time warping*) pour calculer la similarité entre deux segments lors de l'utilisation des *embeddings* [Thiolliere *et al.*, 2015] ; soit par l'utilisation de la distance euclidienne ou la similarité cosine entre les *embeddings* de mots représentant les segments. Dans [Kamper *et al.*, 2015], l'évaluation a été menée sur deux ensembles de mots (Train et Test) issus du corpus anglais *Switchboard*. Après l'apprentissage du modèle sur le corpus Train, la similarité cosine est calculée entre les *embeddings* de chaque paire de mots dans le corpus de Test. Ces paires sont ensuite classées comme similaires ou différentes en appliquant un seuil sur leur distance. Enfin, une courbe de rappel et précision est obtenue en faisant varier le seuil.

Dans ce chapitre, nous proposons deux approches pour évaluer les *embeddings* acoustiques de mots $\mathbf{a}^+$ : les tâches de similarités *orthographique* et *phonétique* et la tâche de *détection d'homophones*. Le but de ces évaluations est de mesurer la perte d'information orthographique portée par $\mathbf{a}^+$ et le gain potentiel d'information acoustique dû à cette projection, par rapport aux informations portées par $\mathbf{o}^+$.

Nous proposons de construire deux jeux d'évaluation comme suit : étant donné une liste $L$ de $n$ mots fréquents (mots candidats) du vocabulaire composé de $m$ mots, une liste de $n \times m$ mots a été créée. Puis, deux alignements ont été effectués entre chaque paire de mots en fonction de leurs représentations orthographiques (lettres) et phonétiques (phonèmes) en utilisant l'outil sclite [1]. À partir de cet alignement, deux distances d'édition sont calculées par rapport aux résultats d'alignement des représentations orthographiques et phonétiques. La distance d'édition est calculée comme suit :

$$SER = \frac{\#In + \#Sub + \#Del}{\#symbols\ in\ the\ reference\ word} \times 100 \tag{6.2}$$

où $SER$ représente le taux d'erreur de symboles (*Symbol Error Rate*). Les symboles correspondent aux lettres pour les représentations orthographiques et aux phonèmes pour celles phonétiques ; $In$, $Sub$ et $Del$ correspondent respectivement aux erreurs d'insertion, de substitution et de suppression.

Puis, nous calculons deux *scores de similarité* : les scores de similarité orthographique et phonétique $sim\_score$ attribués pour chaque paire de mots, définis comme suit :

$$sim\_score = 10 - \min(10, SER/10) \tag{6.3}$$

1. http ://www.icsi.berkeley.edu/Speech/docs/sctk-1.2/sclite.htm

où min() est une fonction utilisée pour obtenir une distance d'édition entre 0 et 10. Pour chaque mot candidat dans la liste $L$, nous extrayons ses dix mots les plus proches orthographiquement et phonétiquement. Il en résulte deux listes pour les tâches de similarité *Orthographique* et *Phonétique*. Pour chaque mot candidat dans la liste $L$ la liste *Orthographique* contient les dix mots les plus proches en termes de scores de similarité orthographique et la liste *Phonétique* contient les dix mots les plus proches en termes de scores de similarité phonétique. Enfin, la liste *Homophones*, utilisée pour la tâche de détection *d'homophones*, contient pour chaque mot candidat, les mots partageant la même représentation phonétique. Le tableau 6.1 présente un exemple du contenu de ces trois listes.

| **Liste** | **Exemples** |
|---|---|
| Orthographique | (très près) 7,5<br>(très ors) 5 |
| Phonétique | (très frais) 6,67<br>(très traînent) 6,67 |
| Homophone | (très traie)<br>(très traient) |

TABLE 6.1 – Exemple du contenu de trois listes.

Ces tâches sont évaluées de deux manières différentes. Pour les tâches de similarité orthographique et phonétique, l'évaluation est réalisée en classant les paires de mots en fonction de leurs similarités cosinus et en mesurant le coefficient de corrélation *Spearman's rank* $\rho$ [2] avec les scores de similarités $sim\_score$ (équation 6.3). Cette approche est utilisée dans [Gao *et al.*, 2014, Ji *et al.*, 2015, Levy *et al.*, 2015] et vue dans la section 5.1.1.3 pour évaluer les *word embeddings* linguistiques sur les tâches de similarité où les scores de similarité sont attribués par des annotateurs humains.

Pour la tâche de détection d'homophones, l'évaluation est réalisée en termes de précision. Pour chaque mot candidat $w$ dans la liste *Homophones*, on définit : $L_H(w)$ sa liste de $l$ homophones ; $L_{H\_neighbour}(w)$ la liste de ses $l$ plus proches voisins extraits en fonction de la similarité cosine des *embeddings* $\mathbf{a}^+$ ; $L_{H\_found}(w)$ la liste des homophones trouvés. Notons que cette dernière liste correspond à l'intersection entre $L_H(w)$ et $L_{H\_neighbour}(w)$. La précision $P_w$ du mot $w$ est définie ainsi :

$$P_w = \frac{|L_{H\_found}(w)|}{|L_H(w)|} \tag{6.4}$$

où $|.|$ correspond à la taille d'une liste. La précision globale de détection d'homophones dans la liste *Homophones* est la moyenne de $P_w$ :

$$P = \frac{\sum_{i=1}^{N_c} P_{w_i}}{N_c} \tag{6.5}$$

2. http://grasland.script.univ-paris-diderot.fr/STAT98/stat98_6/stat98_6.htm

où $N_c$ est le nombre de mots candidats qui ont une liste $L_H(w)$ non vide.

#### 6.1.2.2 Protocole expérimental

Nous présentons dans cette section les données expérimentales ainsi que la description des architectures neuronales utilisées pour construire les *embeddings* acoustiques.

**Données**

Le corpus d'apprentissage du réseau de neurones convolutif est composé de 488 heures d'émissions télévisées (*Broadcast news*) en français. Ce corpus est issu de données provenant des corpus ESTER1 [Galliano *et al.*, 2005], ESTER2 [Galliano *et al.*, 2009] et EPAC [Estève *et al.*, 2010]. Il contient $52k$ mots uniques qui sont vus au moins deux fois chacun dans le corpus pour un total de $5,75$ millions d'occurrences.

En français, plusieurs mots ont la même prononciation sans pour autant posséder la même orthographe ni la même signification : *e.g.* le son [so] correspond à quatre homophones : *sot*, *saut*, *sceau* et *seau* ; et deux fois plus en prenant en compte leurs formes plurielles : *sots*, *sauts*, *sceaux* et *seaux*. Lorsqu'un CNN est entrainé pour prédire un mot étant donné une séquence acoustique, ces homophones peuvent introduire un biais lors de l'évaluation de l'erreur de reconnaissance. Pour éviter cela, nous avons fusionné tous les homophones existant parmi les $52k$ mots uniques du corpus d'apprentissage et obtenu un nouveau dictionnaire réduit contenant $45k$ mots et classes d'homophones.

Les paramètres acoustiques fournis au CNN sont des bancs de filtre, calculés toutes les 10 ms sur une fenêtre de 25ms, correspondant à un vecteur de 23 dimensions pour chaque trame. Un alignement forcé entre les transcriptions manuelles et le signal de parole a été effectué sur le corpus d'apprentissage afin de détecter les frontières de mot. Les statistiques calculées à partir de cet alignement révèlent que 99% des mots sont plus courts que 1 seconde. C'est pourquoi nous avons décidé de représenter le signal audio de chaque mot par 100 trames, donc par un vecteur de 2300 dimensions, étant donné que le signal audio d'un mot est centré. Lorsque les mots sont plus courts, ces représentations sont complétées avec des zéros d'une manière égale sur les deux extrémités, tandis que les mots plus longs sont eux coupés sur les deux extrémités.

**Architectures**

Les architectures profondes CNN et DNN sont entrainées sur 90% du corpus d'apprentissage : les 10% restants sont utilisés pour la validation. Elles sont détaillées dans ce qui suit :

**réseau de neurones convolutifs :** cette architecture prédit un mot étant donnée une séquence de 100 trames. Elle contient deux couches de convolution et de pooling suivies de deux couches totalement connectées qui alimentent la

dernière couche Softmax. Cette dernière couche contient $45k$ unités qui correspondent aux mots et classes d'homophones. Les couches de convolution ont respectivement 15 et 10 filtres de taille 8. La couche de pooling effectue le sous-échantillonnage en prenant le maximum de 4 unités. Les deux couches totalement connectées sont composées respectivement de 500 et 100 unités. La fonction tangente hyperbolique (tanh) est utilisée comme fonction d'activation pour toutes les couches. Ce modèle obtient $61,51\%$ de taux d'exactitude (*i.e.* la précision de reconnaissance de mot) sur le corpus de validation.

**réseau de neurones profond :** cette architecture prend en entrée trois vecteurs de taille 100, 500 et 500 qui correspondent respectivement à l'*embedding* acoustique du signal, l'*embedding* orthographique de mot correspondant et l'*embedding* orthographique d'un mot choisi au hasard. De plus, elle est composée de deux couches tanh totalement connectées de 300 et 100 unités chacune. Pour tester la performance des *embeddings* résultants, nous avons remplacé l'*embedding* de signal $\mathbf{s}$ dans le CNN (*Cf.* figure 6.1) par l'*embedding* acoustique de mot $\mathbf{a}^+$. Le CNN obtient alors 50,15% de taux d'exactitude de reconnaissance de mot. Ceci montre que même si les *embeddings* acoustiques de mots sont moins précis que les *embeddings* de signal, ils sont capables de capturer des informations acoustiques pertinentes dans un espace continu proche de l'espace des *embeddings* de signal.

#### 6.1.2.3 Résultats d'évaluation

Nous rappelons que le but de ces évaluations est de mesurer la perte d'information orthographique portée par les *embeddings* acoustiques $\mathbf{a}^+$ et le gain potentiel d'information acoustique par rapport aux informations portées par les *embeddings* orthographiques $\mathbf{o}^+$.

Les *embeddings* que nous évaluons sont construits à partir de deux vocabulaires différents : celui utilisé pour entrainer les architectures neuronales (CNN et DNN), composé de $52k$ mots présents dans les transcriptions manuelles des 488 heures d'audio ; et un autre composé de $160k$ mots. Les mots présents dans le vocabulaire $52k$ sont presque tous présents dans le vocabulaire de $160k$.

Les jeux d'évaluation décrits dans la section 6.1.2 sont générés à partir de ces deux vocabulaires. Dans le vocabulaire de $52k$ mots, tous les *embeddings* acoustiques de mots $\mathbf{a}^+$ sont liés à des mots qui ont été observés pendant l'apprentissage du CNN. Cela signifie qu'au moins deux *embeddings* acoustiques de signal ont été calculés à partir de l'audio pour chacun de ces mots. Dans le vocabulaire de $160k$, environ $110k$ *embeddings* acoustiques $\mathbf{a}^+$ ont été calculés pour des mots n'ayant jamais été observés dans les données d'apprentissage audio.

##### Évaluation quantitative

Nous rappelons que l'évaluation quantitative des *embeddings* acoustiques $\mathbf{a}^+$ est effectuée sur les tâches de similarité orthographique et phonétique et la tâche de

détection d'homophones. Les résultats sont résumés dans le tableau 6.2.

| Tâches | 52k Vocab. | | 160k Vocab. | |
|---|---|---|---|---|
| | $\mathbf{o}^+$ | $\mathbf{a}^+$ | $\mathbf{o}^+$ | $\mathbf{a}^+$ |
| Orthographique | **0,542** | 0,499 | **0,569** | 0,510 |
| Phonétique | 0,404 | **0,435** | 0,414 | **0,468** |
| Homophone | 64,65 | **72,28** | 52,87 | **59,33** |

TABLE 6.2 – Résultats d'évaluation sur les tâches de similarités orthographique et phonétique ($\rho$) et la tâche de détection d'homophones (*précision* * 100).

Les résultats montrent que les *embeddings* $\mathbf{a}^+$ sont plus pertinents pour capturer des similarités phonétiques, alors que les *embeddings* $\mathbf{o}^+$ sont évidemment les meilleurs pour capturer des similarités orthographiques.

Ces résultats confirment que la projection des *embeddings* orthographiques $\mathbf{o}^+$ dans l'espace des *embeddings* acoustiques de signal $\mathbf{s}$ change leurs propriétés : ils capturent plus d'informations sur la prononciation des mots même s'ils perdent des informations sur l'orthographe. Ainsi, en plus de rendre possible une mesure de distance de similarité entre le signal acoustique (représenté par $\mathbf{s}$) et un mot (représenté par $\mathbf{a}^+$), les *embeddings* acoustiques de mots sont meilleurs que les *embeddings* orthographiques pour mesurer la proximité phonétique entre deux mots.

Pour la tâche de détection d'homophones, la liste des homophones est calculée à partir du vocabulaire $160k$ : cela donne 53869 paires d'homophones au total. Le vocabulaire $52k$ contient 13561 paires d'homophones qui sont incluses dans les couples présents dans le vocabulaire $160k$. On remarque que, les *embeddings* acoustiques obtiennent de meilleurs résultats que les *embeddings* orthographiques sur cette tâche dans les deux vocabulaires. Ceci confirme encore que les *embeddings* acoustiques de mots ont capturé des informations supplémentaires sur la prononciation des mots par rapport à celles qui sont portées par les *embeddings* orthographiques.

Pour cette tâche, nous ne pouvons pas comparer les résultats entre les deux vocabulaires puisque la mesure de précision dépend du nombre d'événements. Pour la corrélation de *Spearman*, une comparaison est approximativement possible et les résultats montrent que la façon de calculer $\mathbf{a}^+$ est efficace pour généraliser ce calcul pour les mots non observés dans les données d'apprentissage audio.

### Évaluation qualitative

Pour mieux comprendre la différence de qualité entre les *embeddings* orthographiques et acoustiques, nous proposons une comparaison empirique en montrant les plus proches voisins d'un ensemble de mots donné. Le tableau 6.3 montre des exemples de mots candidats et leurs plus proches voisins. Comme prévu, on remarque que les voisins d'un mot donné partagent la même orthographe avec lui quand ils sont induits par les *embeddings* orthographiques et sans doute ont la même prononciation que lui quand ils sont induits par les *embeddings* acoustiques.

| Mots candidats | $\mathbf{o}^+$ | $\mathbf{a}^+$ |
|---|---|---|
| grecs | i-grec, rec, marec | grec, grecque, grecques |
| ail | aile, trail, fail | aille, ailles, aile |
| arts | parts, charts, encarts | arte, art, ars |
| blocs | bloch, blocher, bloche | bloc, bloque, bloquent |

TABLE 6.3 – Exemple de mots candidats et leurs plus proches voisins

## 6.2 Évaluation sur la tâche de détection d'erreurs

Le système de détection d'erreurs que nous avons proposé n'intègre jusqu'alors que des informations linguistiques via l'utilisation des *embeddings* linguistiques ainsi que des informations syntaxiques et lexicales. Nous nous sommes intéressés à enrichir notre système par des informations acoustiques et prosodiques afin d'améliorer sa performance. Nous rappelons que jusqu'à présent, le meilleur résultat est obtenu en utilisant le système, nommé *MLP-MS-AutoE*, qui intègre un *embedding* linguistique composite obtenu par combinaison d'*embeddings* simples via un auto-encodeur.

### 6.2.1 Performance des *embeddings* acoustiques

Les informations acoustiques peuvent être obtenues en utilisant les *embeddings* acoustiques de signal $\mathbf{s}$ et de mot $\mathbf{a}^+$. Nous avons utilisé les modèles appris sur les 488 heures d'émissions télévisées et les avons appliqués sur les transcriptions automatiques du corpus ETAPE pour construire les *embeddings* acoustiques. Ces derniers sont ajoutés comme informations supplémentaires dans le vecteur de descripteurs du mot courant $w_i$ et ne sont pas ajoutés pour les mots de contexte.

Pour ces expériences, nous proposons de comparer trois configurations correspondant aux différents types de descripteurs utilisés :

— Configuration 1 (Desc-Base) : descripteurs du système de base *MLP-MS-AutoE* ;

— Configuration 2 (Desc-Base $\oplus$ $\mathbf{s}$) : ajout des *embeddings* acoustiques de signal $\mathbf{s}$ à la configuration 1 ;

— Configuration 3 (Desc-Base $\oplus$ $\mathbf{s}$ $\oplus$ $\mathbf{a}^+$) : ajout des *embeddings* acoustiques de mots $\mathbf{a}^+$ à la configuration 2.

Étant donné que, nous avons utilisé deux architectures neuronales pour la construction de l'*embedding* $\mathbf{a}^+$, *MLP-MS* est utilisé pour les configurations 1 et 2 tandis qu'une nouvelle architecture plus profonde est proposée pour la configuration 3.

#### 6.2.1.1 Architecture *DMLP-MS*

La nouvelle architecture, nommée *DMLP-MS* (*Deep MLP-MS*), est illustrée dans la figure 6.2. Elle correspond à une extension de *MLP-MS*, et est détaillée

comme suit :

— la partie gauche connectée à la couche $H2$-1 correspond à l'architecture *MLP-MS* qui intègre les descripteurs de la configuration 1 (une description détaillée de cette architecture est donnée dans la section 4.1.1).

— la partie droite traite par un flux les *embeddings* acoustiques $\mathbf{s}$ et $\mathbf{a}^+$ qui sont connectés à une seule couche cachée H2-2-AC. Comme l'*embedding* $\mathbf{s}$ porte une information sur le signal et l'*embedding* $\mathbf{a}^+$ porte une information sur l'occurrence d'un mot dans un espace comparable, la distance entre les deux permet de donner une information sur la fiabilité du mot d'un point de vue acoustique. Cette propriété a été utilisée par [Bengio et Heigold, 2014] pour *re-scorer* un graphe de mots. Dans notre cas, nous souhaitons contraindre les *embeddings* acoustiques à n'être connectés qu'à une seule couche cachée pour aider l'algorithme d'apprentissage puisque nous n'avons pas beaucoup de données. En contraignant l'architecture neuronale, on force l'algorithme à extraire ce qu'il est possible d'extraire de la comparaison de $\mathbf{s}$ et $\mathbf{a}^+$. Et c'est la représentation vectorielle de cette comparaison que l'on utilise ensuite (la couche H2-2-AC).

— la troisième couche (H3) prend en entrée la concaténation de la couche H2-1 issue de MLP-MS et la couche H2-2-AC.

— Enfin, la dernière couche prend en entrée la sortie de la couche H3.

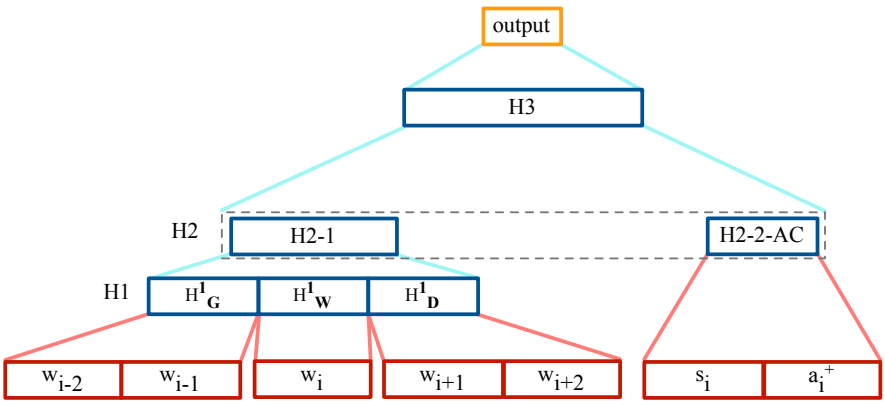



FIGURE 6.2 – L'architecture neuronale *DMLP-MS* pour la détection d'erreurs de SRAP, intégrant en plus des *embeddings* acoustiques.

#### 6.2.1.2 Résultats expérimentaux

Les résultats d'évaluation sont résumés dans les tableaux 6.4 pour le corpus de développement et 6.5 pour celui de test.

Ces tableaux montrent l'utilité des *embeddings* acoustiques de signal $\mathbf{s}$ qui semblent capables de caractériser quelques segments acoustiques suspects. Ces *em-*

| Architectures/ Approche | Représentation de mot | Label erreur | | | Global CER |
|---|---|---|---|---|---|
| | | P | R | F | |
| CRF | discrete | 0,681 | 0,553 | 0,610 | 10,38 |
| MLP-MS | Desc-Base | 0,705 | 0,575 | 0,633 | 9,79 |
| | $\oplus$ **s** | **0,719** | 0,576 | 0,640 | 9,54 |
| DMLP-MS | $\oplus$ **s** $\oplus$ $\mathbf{a}^+$ | 0,717 | **0,582** | **0,642** | **9,53** |

TABLE 6.4 – Performance des *embeddings* acoustiques sur Dev

| Architectures/ Approche | Représentation de mot | Label erreur | | | Global CER |
|---|---|---|---|---|---|
| | | P | R | F | |
| CRF | discrete | 0,676 | 0,547 | 0,605 | 8,56 |
| MLP-MS | Desc-Base | 0,696 | 0,578 | 0,632 | 8,07 |
| | $\oplus$ **s** | 0,696 | **0,591** | 0,639 | 7,99 |
| DMLP-MS | $\oplus$ **s** $\oplus$ $\mathbf{a}^+$ | **0,700** | 0,589 | **0,640** | **7,94** |

TABLE 6.5 – Performance des *embeddings* acoustiques sur Test

*beddings* ajoutés seuls apportent une amélioration en termes de réduction de CER par rapport aux résultats de système de base (*MLP-MS-AutoE*). Une légère amélioration supplémentaire est également observée en ajoutant les *embeddings* acoustiques de mots $\mathbf{a}^+$. En comparaison avec le système CRF, notre approche neuronale entière utilisant les *embeddings* **s** et $\mathbf{a}^+$ obtient des améliorations statistiquement significatives qui correspondent à 8,18% et 7,24% en termes de réduction CER sur Dev et sur Test.

### 6.2.2 Performance des informations prosodiques

#### 6.2.2.1 Descripteurs prosodiques

Nous nous sommes intéressés également à enrichir notre système de détection d'erreurs par des descripteurs prosodiques (*pros*) en plus des decsripteurs acoustiques et linguistiques. Bien que les descripteurs prosodiques aient déjà été utilisés dans le cadre de la détection d'erreurs au niveau de l'énoncé [Hirschberg *et al.*, 2004, Goldwater *et al.*, 2010], leur utilisation pour la détection d'erreurs au niveau du mot a été moins étudiée.

Notre utilisation des paramètres prosodiques est motivée par les travaux précédents [Hirschberg *et al.*, 2004, Goldwater *et al.*, 2010, Stoyanchev *et al.*, 2012a]. Stoyanchev *et al.* ont montré que la combinaison des descripteurs prosodiques et syntaxiques est utile pour localiser les mots mal reconnus dans un tour de parole. Ils ont proposé deux approches. La première est une classification simple des mots en une seule étape. La deuxième approche se fait en deux étapes. Elle consiste d'abord à déterminer les énoncés mal reconnus, puis à détecter les mots erronés dans ces énoncés. Hirschberg *et al.* ont trouvé que les paramètres prosodiques sont utiles pour la détection d'erreurs dans les énoncés. Sharon *et al.* ont découvert que les

mots mal reconnus ont des valeurs prosodiques extrêmes.

Les descripteurs prosodiques utilisés ont été fournis par nos collègues du Laboratoire de Phonétique et Phonologie (LPP). Ils sont calculés comme suit : d'abord, les outils du LIMSI [Gauvain *et al.*, 2005] ont été utilisés pour effectuer deux alignements forcés avec le signal audio : (i) en utilisant les transcriptions de référence et (ii) en utilisant les transcriptions automatiques produites par le SRAP du LIUM. À partir de ces alignements, sont obtenus, entre autres : *le nombre* et *la durée moyenne des phonèmes*, *la durée du mot* ainsi que *la durée de la pause précédente et suivante* ; La boite à outils Praat[3] [Boersma et Weenink, 2001] a été utilisée pour analyser le signal de parole, le segmenter en voyelle et mesurer *la fréquence fondamentale* ($f0$) : *i.e.* un paramètre acoustique correspondant à la fréquence vocale perçue. Cette fréquence est mesurée toutes les 5 millisecondes, selon la méthode utilisée dans [Adda-Decker *et al.*, 2008]. Les durées de phonèmes extraites lors de l'alignement forcé sont utilisées pour calculer les valeurs $f0$ à la fois pour les transcriptions de référence et d'hypothèse, sur trois points (début, milieu et fin) pour chaque segment vocalique. Comme les propriétés des segments centraux présentent une plus grande stabilité acoustique, les mesures prises au milieu des segments sont utilisées pour calculer les valeurs $f0$ au niveau du mot : *f0 moyenne du mot* (en Hertz) et *delta entre f0 du premier segment vocalique du mot* et *f0 du dernier segment vocalique du mot* (en Hertz et en demi-ton). La figure 6.3 illustre les paramètres prosodiques de deux séquences de mots de la référence (ref.) : "ficelle" (a) et "fils et l"' (b) qui sont transcrits par le même mot hypothèse "ficelle". Ici, l'exemple (b) a été mal transcrit. Cet exemple montre comment les modèles prosodiques peuvent contribuer à identifier les mots corrects parmi les différents homophones possibles des séquences de mots.

#### 6.2.2.2 Résultats expérimentaux

Nous avons intégré les descripteurs prosodiques comme information supplémentaire dans les vecteurs de descripteurs du mot courant ainsi que des mots en contexte. Ceci se justifie par le fait que les descripteurs prosodiques sont extraits au niveau de l'énoncé et l'information prosodique d'un mot dépend de son environnement.

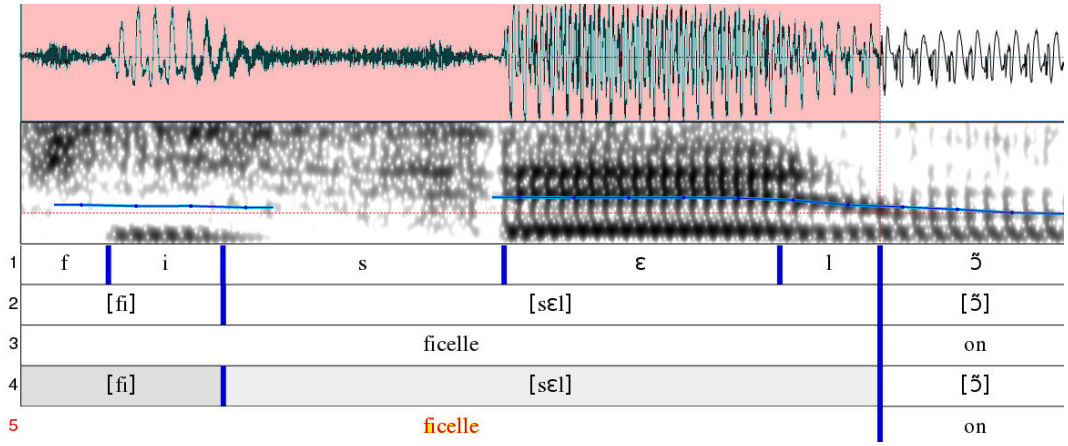
Pour ces expériences, nous avons comparé également plusieurs configurations, correspondant aux différents types de descripteurs utilisés en utilisant les deux architectures *MLP-MS* et *DMLP-MS* :

— Configuration 1 (Desc-Base) : descripteurs du système de base *MLP-MS-AutoE* ;

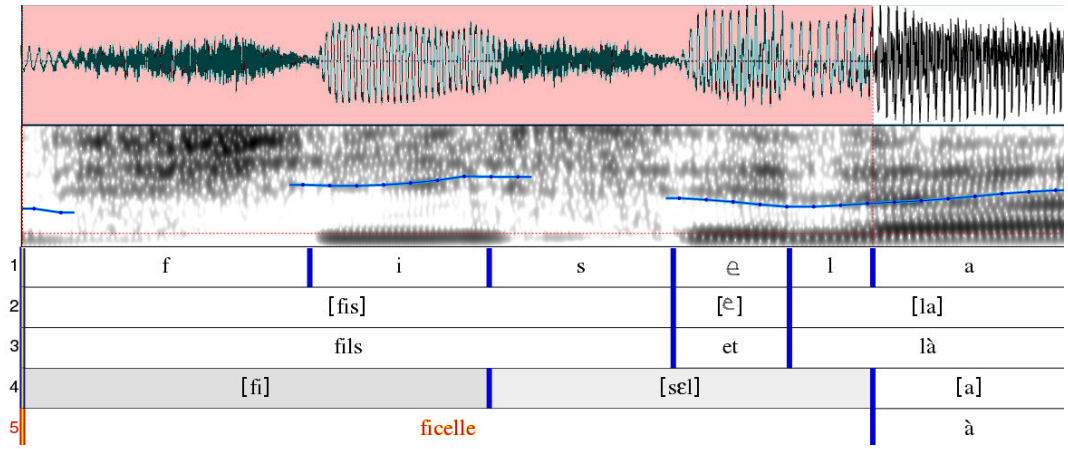
— Configuration 2 (Desc-Base $\oplus$ pros) : ajout des descripteurs prosodiques à la configuration 1 ;

— Configuration 3 (Desc-Base $\oplus$ **s**) : ajout des *embeddings* acoustiques de signal à la configuration 1 ;

3. http ://www.fon.hum.uva.nl/praat/




(a)




(b)

FIGURE 6.3 – Illustration de paramètres prosodiques pour la reconnaissance de la parole. Deux exemples [fisɛl] : (a) le mot 'ficelle (on)' ; (b) le mot 'fils et l(à)' qui a été mal transcrit 'ficelle' par le SRAP, les séquences de phonèmes des mots prononcés et reconnus étant identiques. Les lignes sous le spectrogramme sont : 1 = phonèmes ; 2 = ref.syllables ; 3 = ref.mots ; 4 = hyp.syllables ; 5 = hyp.mots.

— Configuration 4 (Desc-Base $\oplus$ **s** $\oplus$ pros) : ajout des *embeddings* acoustiques de signal **s** à la configuration 3 ;

— Configuration 5 (Desc-Base $\oplus$ **s** $\oplus$ $\mathbf{a}^+$) : ajout des *embeddings* acoustiques de mots $\mathbf{a}^+$ à la configuration 3 ;

— Configuration 6 (Desc-Base $\oplus$ **s** $\oplus$ $\mathbf{a}^+$ $\oplus$ pros) : ajout des descripteurs prosodiques à la configuration 5.

Les résultats expérimentaux sont résumés dans les tableaux 6.6 et 6.7. Les descripteurs prosodiques sont intégrés également dans le système CRF afin d'évaluer leurs impacts sur l'ensemble des configurations lorsqu'ils sont combinés avec les autres descripteurs.

| **Architecture/ Approche** | **N** | **Représentation de mot** | **Label erreur** | | | **Global** |
|---|---|---|---|---|---|---|
| | | | **P** | **R** | **F** | **CER** |
| CRF *(baseline)* | | discrète | 0,681 | 0,553 | 0,610 | 10,38 |
| *CRF* $\oplus$ *pros* | | | 0,698 | 0,550 | 0,615 | 10,12 |
| MLP-MS | 1 | Desc-Base | 0,705 | 0,575 | 0,633 | 9,79 |
| | 2 | Desc-Base $\oplus$ **s** | 0,719 | 0,576 | 0,640 | 9,54 |
| | 3 | Desc-Base $\oplus$ pros | 0,692 | **0,634** | **0,662** | 9,52 |
| | 4 | Desc-Base $\oplus$ **s** $\oplus$ pros | **0,725** | 0,586 | 0,648 | **9,35** |
| DMLP-MS | 5 | Desc-Base $\oplus$ **s** $\oplus$ $\mathbf{a}^+$ | 0,717 | 0,582 | 0,642 | 9,53 |
| | 6 | Desc-Base $\oplus$ **s** $\oplus$ $\mathbf{a}^+$ $\oplus$ pros | 0,717 | 0,599 | 0,653 | **9,38** |

TABLE 6.6 – Performance des descripteurs prosodiques quand ils sont combinés avec les *embeddings* linguistiques et acoustiques sur Dev

Généralement, on remarque que l'intégration des descripteurs prosodiques est avantageuse. En effet, ils améliorent les résultats des systèmes CRF et neuronaux. En ce qui concerne les résultats obtenus par les systèmes neuronaux, on observe que l'intégration des descripteurs prosodiques (3) ou des *embeddings* acoustiques de mot (5) obtiennent des résultats proches en termes de CER. Les améliorations obtenues par la configuration 3 par rapport à la 1 correspondent environ à $2,7\%$ et $1,3\%$ de réduction de CER sur les corpus Dev et Test. Enfin, on voit que la combinaison des *embeddings* linguistiques, acoustiques et des descripteurs prosodiques dans les configurations 4 et 6, obtient les résultats les plus intéressants. Ceci montre la complémentarité des informations acoustique et prosodique. La configuration 6 améliore significativement les résultats de $7,31\%$ et $8,39\%$ en termes de réduction de CER par rapport au CRF $\oplus$ pros respectivement sur les corpus Dev et Test.

## 6.3 Conclusion

Dans ce chapitre, nous avons présenté une approche pour construire les *embeddings* acoustiques. Celle-ci repose sur l'utilisation d'un réseau de neurones convolutif pour construire des *embeddings* acoustiques de signal pour chaque occurrence de

| Architecture/ Approche | N | Représentation de mot | Label erreur | | | Global |
|---|---|---|---|---|---|---|
| | | | P | R | F | CER |
| CRF *(baseline)* | | discrète | 0,676 | 0,547 | 0,605 | 8,56 |
| CRF ⊕ pros | | | 0,686 | 0,541 | 0,605 | 8,46 |
| MLP-MS | 1 | Desc-Base | 0,696 | 0,578 | 0,632 | 8,07 |
| | 2 | Desc-Base ⊕ **s** | 0,696 | 0,591 | 0,639 | 7,99 |
| | 3 | Desc-Base ⊕ pros | 0,681 | **0,631** | **0,655** | 7,96 |
| | 4 | Desc-Base ⊕ **s** ⊕ pros | **0,708** | 0,595 | 0,646 | 7,79 |
| DMLP-MS | 5 | Desc-Base ⊕ **s** ⊕ $\mathbf{a}^+$ | 0,700 | 0,589 | 0,640 | 7,94 |
| | 6 | Desc-Base ⊕ **s** ⊕ $\mathbf{a}^+$ ⊕ pros | 0,702 | 0,613 | 0,654 | **7,75** |

TABLE 6.7 – Performance des descripteurs prosodiques quand ils sont combinés avec les *embeddings* linguistiques et acoustiques sur Test

mot, et d'un réseau de neurones profond *feedforward* pour obtenir un seul *embedding* acoustique pour chaque mot. Le réseau de neurones prodond permet également de construire des *embeddings* acoustiques pour les mots non observés dans le corpus d'apprentissage audio.

Dans un second temps, nous avons étudié l'évaluation intrinsèque des *embeddings* acoustiques de mots. Ces derniers offrent l'opportunité de comparer en termes de similarité une représentation acoustique *a priori* de mot à une représentation continue du signal audio. Nous avons proposé deux approches pour évaluer les performances de ces *embeddings* acoustiques par rapport à leurs *embeddings* orthographiques : les tâches de similarités *orthographique* et *phonétique* et la tâche de *détection d'homophones* . Les expériences montrent que les *embeddings* acoustiques de mots sont plus performants que les *embeddings* orthographiques pour mesurer la proximité phonétique entre deux mots. De plus, ils obtiennent les meilleurs résultats sur la tâche de détection d'homophones. Ceci confirme que les *embeddings* acoustiques de mot ont capturé des informations supplémentaires sur la prononciation du mot.

Enfin, nous nous sommes focalisés sur l'évaluation des *embeddings* acoustiques ainsi que des descripteurs prosodiques pour la tâche de détection d'erreurs. Nous avons trouvé que l'ajout d'information acoustique ou prosodique permet d'obtenir des performances équivalentes en termes de CER, et améliore les résultats par rapport à l'utilisation uniquement d'informations linguistique et syntaxique. Nous avons trouvé également que les informations acoustiques et prosodiques sont complémentaires. En effet, nous avons obtenu avec cette combinaison les meilleurs résultats et ce de manière significative par rapport aux systèmes MLP-MS-AutoE et CRF.

CHAPITRE 7

# AMÉLIORATION DE LA DÉTECTION D'ERREURS APRÈS ANALYSE QUALITATIVE DES SORTIES DU SYSTÈME PROPOSÉ

**Sommaire**

Nous avons présenté dans les chapitres précédents notre système de détection d'erreurs le plus performant, le *DMLP-MS*, une architecture neuronale profonde incluant comme source d'informations : les *embeddings* linguistiques et acoustiques, les descripteurs syntaxiques, lexicaux et prosodiques ainsi que des informations contextuelles extraites des mots voisins. Nous le nommons désormais *DMLP-MS-AC* qui correspond bien à la configuration 6 présentée dans la section 6.2.2.2.

Dans ce chapitre, nous présentons tout d'abord une étude portant sur l'analyse des erreurs commises par notre système de détection d'erreurs de reconnaissance de la parole, selon plusieurs critères.

Pour compenser certains phénomènes mis en avant par cette analyse, nous proposons ensuite une étude sur la modélisation de l'erreur de reconnaissance au niveau de la phrase.

## 7.1 Analyse d'erreurs de classification

Dans cette section, nous nous intéressons à l'analyse des sorties de notre système de détection d'erreurs *DMLP-MS-AC* afin de percevoir les erreurs de reconnaissance qui sont difficiles à détecter. Ces analyses sont réalisées sur le corpus Dev selon plusieurs critères.

Nous allons utiliser par la suite les trois termes suivants : vérité terrain, prédictions et prédictions correctes. La vérité terrain correspond à l'étiquetage de référence. Cet étiquetage est extrait à partir de l'alignement des transcriptions de référence avec les transcriptions automatiques. Les prédictions sont les étiquettes *erreur* attribuées par le système *DMLP-MS-AC* sur des mots reconnus automatiquement. Les prédictions correctes sont les étiquettes *erreur* correctement attribuées (*i.e.* lorsque les mots erronés ont été correctement détectés).

### 7.1.1 Longueur du mot

On cherche dans cette analyse à étudier l'incidence de la longueur d'un mot sur les erreurs de reconnaissance de la parole et la détection de ces erreurs.

Le résultat de cette analyse est résumé dans la figure 7.1. Celle-ci présente le rappel et la précision pour la détection de mots erronés ainsi que le pourcentage d'erreurs du SRAP sur Dev. Nous remarquons que 47% des mots erronés du système sont de longueur 2 et 3 caractères. Sur cet ensemble de mots, la précision et le rappel sont parmi les plus bas.

Il peut être intéressant d'affiner notre analyse sur les mots de longueur 2 et 3. Nous remarquons que beaucoup de ces mots sont des mots outils. Dans la section suivante, notre analyse porte alors sur la détection d'erreurs en distinguant mots outils et mots non-outils.

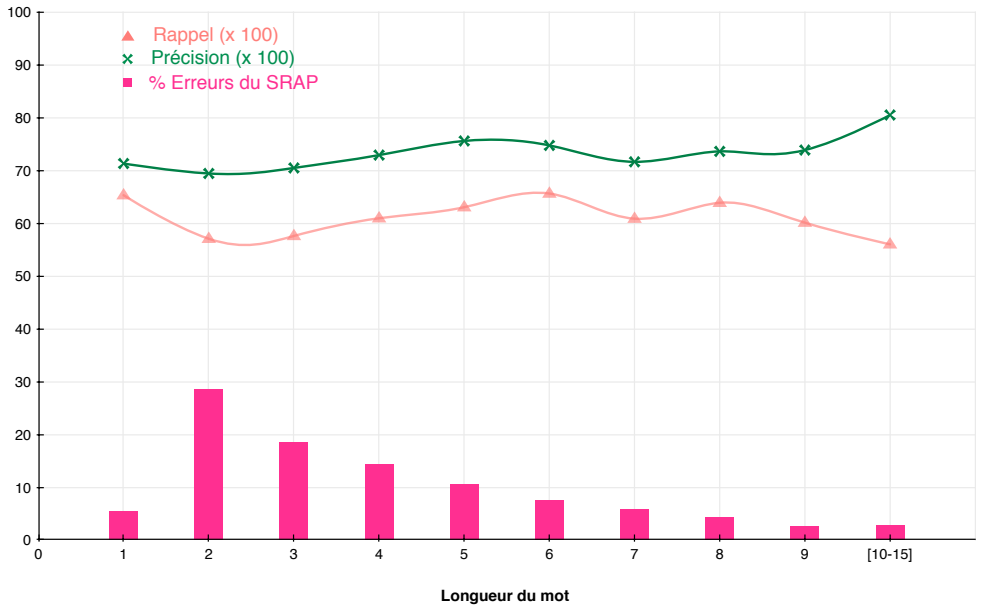



FIGURE 7.1 – Rappel et précision pour la détection de mots erronés et pourcentage de mots erronés en fonction de la longueur de mot sur Dev.

### 7.1.2 Mots outils/non outils

Les mots outils sont une catégorie de mot dont le rôle syntaxique est plus important que le rôle sémantique. Ce sont généralement les mots les plus fréquents dans le corpus. Les mots non-outils sont généralement des mots supports de la sémantique.

Dans notre analyse nous considérons une liste de 145 mots outils. Elle inclut les articles (le, la, les, ...), les prépositions (à, par, dans, ...), les pronoms (elle, il, ...), les mots de liaisons (et, plus, mais, ...), et les 70 mots les plus fréquents dans le corpus d'apprentissage d'ETAPE (que, qui, sur, plus, même, ...).

La description du corpus Dev en fonction des mots outils et non-outils est présentée dans le tableau 7.1.

| | **Mots outils** | **Mots non-outils** |
|---|---|---|
| **lexique** | 145 | 5,9K |
| **#occ. mots** | 29,2K | 20,3K |
| **#occ. err.** | 4,2K | 3,1K |

TABLE 7.1 – Description du corpus Dev en fonction de mots outils/non-outils. **#occ. mots** est le nombre d'occurence total des mots, alors que **#occ. err.** est le nombre d'occurence de mots erronés.

Le tableau 7.2 présente le rappel et la précision pour la détection de mots erronés en fonction des mots outils/non-outils.

| Mots | Label erreur | |
|---|---|---|
| | **P** | **R** |
| Non-outils | **0,756** | **0,657** |
| Outils | 0,686 | 0,558 |

TABLE 7.2 – Rappel et précision pour la détection de mots erronés en fonction des mots outils/non outils sur Dev.

Nous observons que le système *DMLP-MS-AC* détecte mieux les mots non-outils que les mots outils. Concernant la catégorie des mots outils, 71% des mots outils erronés sont de longueur 2 ou 3 caractères, comme : des, c', ce, ces, aux, mon, ne, va, que, ... Ceci corrobore l'observation précédente sur la difficulté de notre système à reconnaitre les mots de longueur 2 et 3.

Nous constatons qu'une bonne précision de 0,75 est atteinte pour la détection des mots non-outils erronés. Néanmoins, on aimerait améliorer le rappel car trop de mots porteurs de sens seront indiqués corrects alors qu'ils sont erronés. Le fait que ces mots porteurs de sens ne soient pas détectés comme erronés est peut être lié à l'information contextuelle utilisée. Il est difficile de capturer des anomalies sémantiques entre les mots dans une fenêtre de contexte de taille réduite.

### 7.1.3 L'empan moyen d'erreurs

Un autre critère qui nous intéresse dans ces analyses est l'empan moyen de l'erreurs. Un empan d'erreurs se définit par le segment constitué d'une suite contiguë de mots erronés. Nous définissons également un empan d'erreurs correctement détectés par le segment constitué d'une suite contiguë de mots erronés correctement détectés.

Le tableau 7.3 présente la taille moyenne (en nombre de mots) des empans et l'écart type pour la vérité terrain, les prédictions et les prédictions correctes de deux systèmes : *DMLP-MS-AC* et CRF⊕*pros*.

| **Corpus** | **Approche** | | **Taille moyenne de l'empan** | **Écart type** |
|---|---|---|---|---|
| Train | | vérité terrain | 3,03 | 1,72 |
| Dev | | | 3,24 | 2,15 |
| Dev | CRF⊕*pros (baseline)* | prédictions | 3,28 | 1,77 |
| | | prédictions correctes | 2,88 | 1,34 |
| | *DMLP-MS-AC* | prédictions | 2,82 | 1,28 |
| | | prédictions correctes | 2,66 | 1,05 |

TABLE 7.3 – L'empan moyen et l'écart-type pour la vérité terrain, les prédictions et les prédictions correctes de *DMLP-MS-AC* et CRF⊕*pros*.

On observe que la taille moyenne de l'empan des prédictions du CRF⊕*pros* est proche de celle de la vérité terrain. En revanche, celle du système *DMLP-MS-AC* est, elle, plus petite de $12,9\%$ par rapport à la vérité terrain avec un écart type

plus petit de 40, 5%. En ce qui concerne les prédictions correctes, l'empan moyen du CRF⊕*pros* est plus proche de celui de la vérité terrain. Néanmoins, les prédictions correctes, tant pour celles produites par le système CRF⊕*pros* que par celui proposé, présentent des empans de taille bien inférieure à la vérité terrain.

L'écart lié à la taille de l'empan d'erreur entre la vérité terrain, les prédictions et les prédictions correctes, est dû à l'architecture de *DMLP-MS-AC*. Contrairement aux CRF, dont le décodeur produit des séquences d'étiquettes à partir d'un calcul sur l'ensemble de la séquence d'entrée, notre système prend ses décisions en s'appuyant sur un contexte local restreint.

### 7.1.4 Nombre d'erreurs dans le contexte

Le dernier critère d'analyse est le nombre d'erreurs de reconnaissance dans le contexte. Nous rappelons que notre système de détection d'erreurs prend en entrée une fenêtre de taille 5. L'objectif de cette analyse est d'étudier le comportement des prédictions, en fonction du nombre d'erreurs dans le contexte : de 0 (cas d'une erreur isolée) à 4 erreurs (contexte totalement erroné).

La figure 7.2 présente le rappel et la précision pour la détection des mots erronés en fonction du nombre d'erreurs dans le contexte. Nous observons que plus il y a d'erreurs de reconnaissance dans le contexte plus le système détecte correctement si le mot central est erroné, alors qu'il a des difficultés à détecter les erreurs isolées. Nous expliquons cela par le fait que la plupart des erreurs isolées ne produisent pas une anomalie linguistique importante facilement détectable par le système de détection d'erreurs, au contraire d'une séquence de 5 mots dont la plupart seraient mal reconnus.

## Discussions

D'après les analyses réalisées sur les sorties du système *DMLP-MS-AC*, nous pouvons constater les points suivants :

— le système détecte mieux les mots non-outils, mais nous espérons améliorer le rappel ;

— la taille moyenne des empans d'erreur de notre système est plus faible que celle de la vérité terrain ;

— le système a des difficultés à détecter les erreurs isolées.

Les résultats de ces analyses nous ont fourni des informations utiles afin d'améliorer les performances de notre système de détection d'erreurs. En s'appuyant sur ces résultats, nous avons choisi d'explorer deux pistes, qui consistent à étendre au niveau de la phrase les informations prises en compte pour la détection d'erreurs. Pour cela, nous explorons deux approches : la première porte sur l'intégration des connaissances globales sur la phrase, et la deuxième repose sur l'utilisation d'un modèle contextuel probabiliste qui porte des informations sur la distribution d'erreurs.

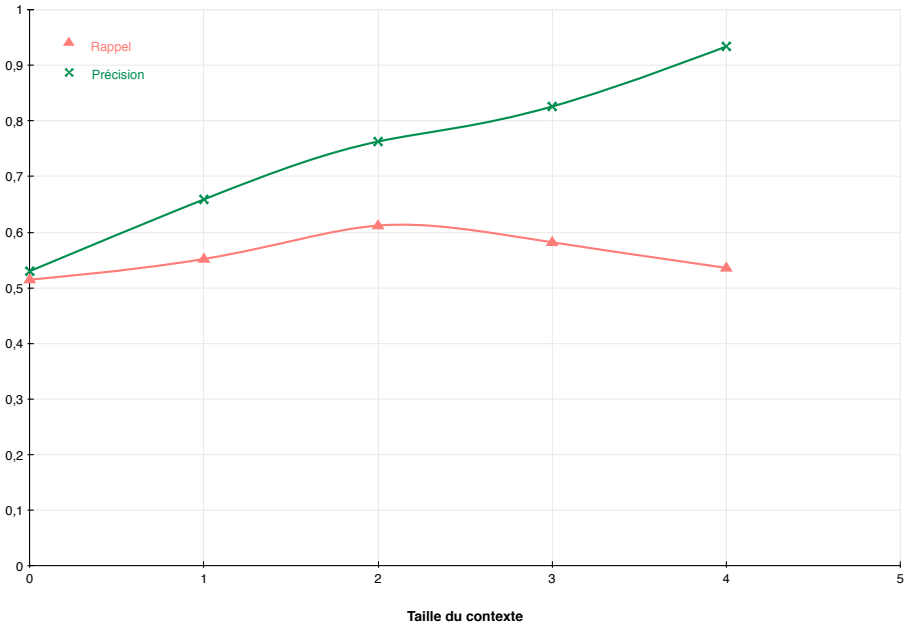



FIGURE 7.2 – Rappel et précision pour la détection des mots erronés en fonction du nombre d'erreurs dans son contexte sur Dev.

## 7.2 Prise en compte de l'ensemble de la phrase pour la détection d'erreurs

### 7.2.1 Intégration d'informations caractérisant la phrase

Dans cette section, nous nous intéressons à l'intégration d'informations globales pour enrichir notre système de détection d'erreurs. Pour cela, nous exploitons des *embeddings* de phrases (*Emb-Ph*). Ces derniers ont été utilisés avec succès dans les tâches de classification de phrases et l'analyse de sentiments [Le et Mikolov, 2014, Lin *et al.*, 2015, Tang *et al.*, 2016]. Ces *embeddings* peuvent être appris d'une manière générale en utilisant l'outil *Doc2vec* [Le et Mikolov, 2014], ou d'une manière spécifique à la tâche, comme dans [Ren *et al.*, 2016]. Dans cette étude, les auteurs ont proposé une approche pour construire des *embeddings* de phrases adaptés à la tâche de l'analyse de sentiments. Pour la tâche de détection d'erreurs, nous proposons d'exploiter deux types d'*embeddings* de phrases.

#### 7.2.1.1 *Embeddings* généralistes de phrases

Ce premier type d'*embeddings* s'appuie sur la méthode sac de mots distribués DBOW (*Distributed bag of words*) fournie par *Doc2vec* [Le et Mikolov, 2014].

L'architecture neuronale DBOW est illustrée dans la figure 7.3. Celle-ci prend en entrée un vecteur de paragraphes d'indice $i$, correspondant à la $i^{me}$ colonne dans la table de correspondance (*Lookup Table*), nommée $D$ dans la figure. Cette table

correspond à une association des indices de paragraphes à des vecteurs qui sont appris au fur et à mesure de l'apprentissage.

Lors de la construction des *embeddings*, et donc durant l'apprentissage de ce réseau de neurones, la sortie représente des mots choisis aléatoirement du paragraphe courant $i$ : cette architecture vise donc à prédire ces mots en fonction du paragraphe auquel ils appartiennent.

Cette architecture est similaire à celle de Skip-gram intégrée dans l'outil word2vec [Mikolov *et al.*, 2013b] qui permet de construire des *embeddings* de mots.

Ce modèle est appris sur le corpus ETAPE pour construire un *embedding* de 100 dimensions pour chaque transcription automatique, nommé $Emb_{DBOW}$.

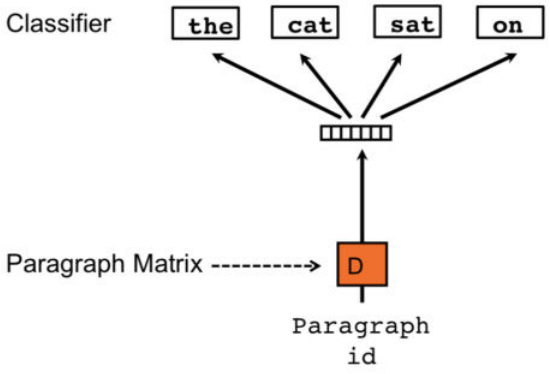



FIGURE 7.3 – L'architecture sac de mots distribué (DBOW) pour la construction des *embeddings* de phrases.

#### 7.2.1.2 *Embeddings* (de phrases) spécifiques à la tâche

Les *embeddings* de phrases $Emb_{DBOW}$ portent une information sémantique contenue dans les transcriptions, mais ne portent probablement pas une information spécifique à la tâche de détection d'erreurs. C'est pourquoi nous proposons de construire des *embeddings* de phrases spécifiques à cette tâche de détection d'erreurs.

Pour ce faire, nous proposons d'utiliser les *embeddings* extraits d'un réseau de neurones convolutif (*CNN-ph*) appris pour la classification des transcriptions automatiques de phrases complètes en phrases *peu erronées* (PE) ou *très erronées* (TE). Ces *embeddings* sont susceptibles de capter des informations sur l'erreur.

L'apprentissage du *CNN-ph* nécessite l'utilisation d'un corpus annoté. C'est pourquoi nous avons annoté les transcriptions d'ETAPE en *peu erronées* ou *très erronées*.

Dans notre étude, nous considérerons arbitrairement une phrase comme très erronée si 20% des mots qui la composent sont incorrects. En effet, comme notre objectif est de construire des *embeddings* portant une information sur l'erreur dans les phrases, le choix du 20% semble raisonnable pour dire si la phrase est perturbée

ou non. Les phrases comprenant moins de 20% de mots incorrects sont alors considérées comme peu erronées (ensemble incluant les phrases totalement correctes). Le tableau 7.4 présente la description des données utilisées pour l'apprentissage des *embeddings* de phrases $Emb_{CNN-ph}$.

| Nom | #Ph ref | #Ph hyp | #Ph PE | #Ph TE |
|---|---|---|---|---|
| Train | 22K | 21,3K | 13,3K | 8,3K |
| Dev | 3,7K | 3,5K | 2,2k | 1,3k |
| Test | 3,6K | 3,5K | 2,3K | 1,1K |

TABLE 7.4 – Description des données expérimentales pour l'apprentissage des *embeddings* de phrases extraits du réseau de neurones convolutif. Le symbole #Ph représente le nombre de phrases.

Le réseau de neurones convolutif prend en entrée une phrase représentée par un vecteur de descripteurs. Ce dernier correspond alors à la concaténation des vecteurs de descripteurs des mots qui la composent. Ainsi, chaque mot est représenté par un vecteur composé des descripteurs décrits dans le chapitre 4, des *embeddings* acoustiques et des descripteurs prosodiques décrits dans le chapitre 6. Ces descripteurs sont tous utilisés dans ce chapitre.

La taille de la phrase est fixée à 50 mots, du fait que 98,37% des phrases, dans les corpus Train, Dev, ont une taille qui varie entre 1 et 50 mots. Lorsque les phrases sont courtes (moins de 50 mots), les descripteurs des mots sont centrés et le vecteur de la phrase est rempli par des zéros d'une manière égale sur les deux extrémités, alors que les phrases longues sont, elles, coupées également sur les deux extrémités, c'est à dire on conserve l'information située au centre de la phrase.

L'architecture proposée est illustrée dans la figure 7.4, elle prend en entrée le vecteur de descripteurs de la phrase et attribue en sortie une étiquette *peu erronée* ou *très erronée* pour chaque phrase. Elle est composée de deux couches de convolution et de sous échantillonnage, suivie par deux couches de neurones qui sont totalement connectées sous la forme d'un MLP. La couche juste avant la couche *Softmax* est utilisée comme *embeddings* de phrases, nommés $Emb_{CNN-ph}$. Le *CNN-ph* obtient 13,50% de taux d'erreur de classification des transcriptions sur Test.

#### 7.2.1.3 Architectures

Afin de savoir comment intégrer l'*embedding* de phrase ($Emb_{DBOW}$ ou $Emb_{CNN-ph}$) dans l'architecture *DMLP-MS* d'une manière efficace, nous avons étudié trois niveaux d'intégration possible, résumés en images dans la figure 7.5. Les architectures proposées sont détaillées dans ce qui suit :

— *DMLP-MS Tune* : dans cette architecture, l'*embedding* de la phrase est traité par un flux, de la même manière que les vecteurs de mots et les *embeddings* acoustiques. En effet, un *pretraining* est effectué pour la couche cachée *HEmb-Ph* en utilisant un MLP qui prend en entrée l'*embedding* de phrase. Ensuite, on fait le *finetuning* de tout le réseau.

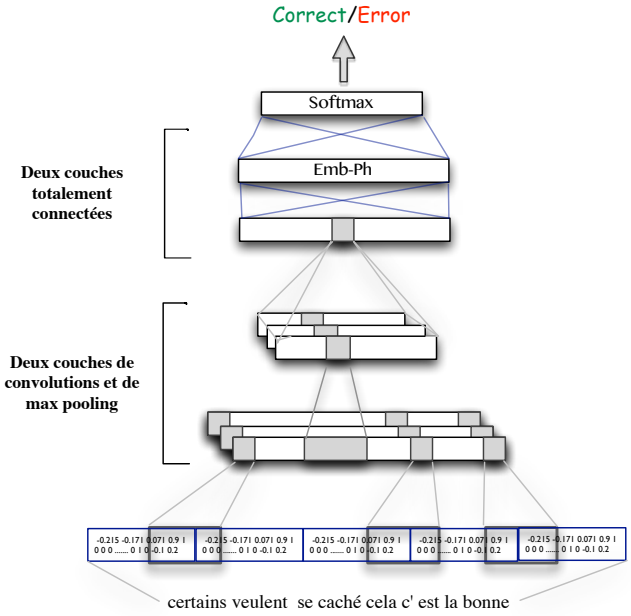



FIGURE 7.4 – Architecture d'un réseau de neurone convolutif pour la classification des phrases en *correct* ou *erreur*.

— *DMLP-MS Inject H3* : dans cette architecture l'*embedding* de la phrase, correspondant aux mots en entrée, est concaténé avec les couches $H2$-1 et $H2$-2-$AC$. Le vecteur résultant ($H2$) est utilisé ensuite comme entrée de la couche cachée $H3$.

— *DMLP-MS Inject O* : dans cette architecture l'*embedding* de la phrase est concaténé avec la couche $H3$. Le vecteur résultant ($H3'$) est utilisé ensuite comme entrée de la couche de sortie $O$.

#### 7.2.1.4 Résultats

Nous résumons dans cette section les résultats de comparaisons des *embeddings* de phrases $Emb_{DBOW}$ et $Emb_{CNN-Ph}$ en utilisant les différentes architectures.

À partir du réseau de neurones convolutif *CNN-ph*, on extrait un *embedding* de 100 dimensions pour chaque transcription dans les corpus Train, Dev et Test.

Les résultats comparatifs de ces deux types d'*embeddings* utilisant les trois architectures proposées sont résumés dans les tableaux 7.5 et 7.6. Ces résultats sont comparés à ceux obtenus par le système CRF⊕ pros et le meilleur système neuronal *DMLP-MS-AC*.

On remarque que les deux types d'*embeddings* apportent une légère amélioration par rapport aux résultats de *DMLP-MS-AC*. Dans la plupart des cas, les *embeddings* $Emb_{CNN-Ph}$ obtiennent de meilleurs résultats que les *embeddings*

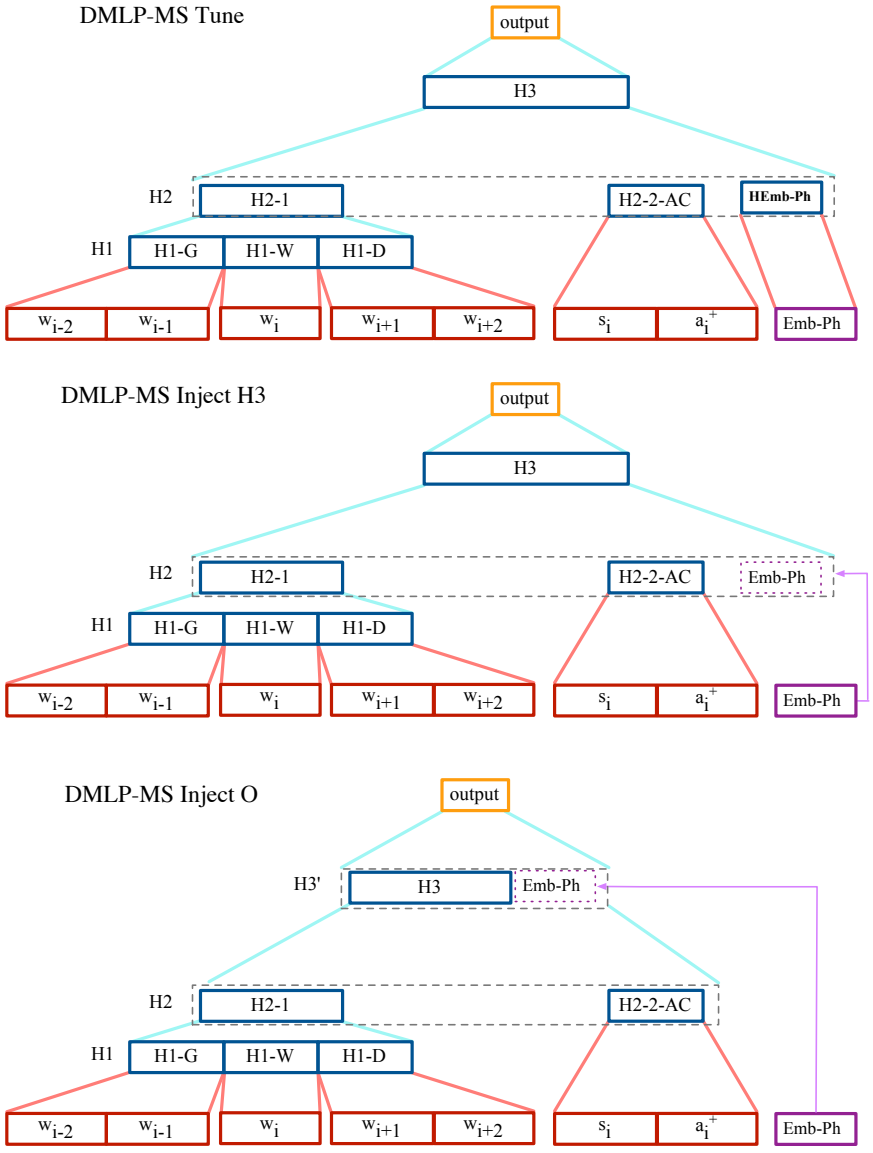



FIGURE 7.5 – Architectures neuronales proposées pour la détection d'erreurs de SRAP, intégrant les *embeddings* de phrases.

| Architecture/ Approche | Représentation de phrase | Label erreur | | | Global |
|---|---|---|---|---|---|
| | | P | R | F | CER |
| CRF ⊕ pros *(baseline)* | - | 0,691 | 0,550 | 0,615 | 10,12 |
| DMLP-MS-AC | - | 0,717 | 0,599 | 0,653 | 9,38 |
| DMLP-MS Inject H3 | $Emb_{DBOW}$ | 0,703 | 0,632 | 0,666 | 9,33 |
| | $Emb_{CNN-Ph}$ | 0,722 | 0,598 | 0,654 | 9,29 |
| DMLP-MS-AC Inject O | $Emb_{DBOW}$ | 0,701 | 0,631 | 0,667 | **9,28** |
| | $Emb_{CNN-Ph}$ | 0,699 | **0,645** | **0,670** | 9,31 |
| DMLP-MS-AC Tune | $Emb_{DBOW}$ | **0,727** | 0,582 | 0,649 | 9,36 |
| | $Emb_{CNN-Ph}$ | 0,724 | 0,598 | 0,655 | **9,26** |

TABLE 7.5 – Performance des *embeddings* de phrases $Emb_{DBOW}$ et $Emb_{CNN-Ph}$ sur Dev

$Emb_{DBOW}$. Nous pouvons émettre l'hypothèse que les *embeddings* extraits du *CNN-ph* ont capté une information sur l'erreur utile pour la détection au niveau mot. Le meilleur résultat est obtenu en utilisant l'*embedding* $Emb_{CNN-Ph}$ avec l'architecture *DMLP-MS-AC Tune*. C'est donc l'architecture *Tune* qui est choisie pour intégrer l'embedding de phrase issue du réseau convolutif dans la suite de ce document. Ce système obtient 1,27% et 0,77% de réduction de CER par rapport aux résultats de *DMLP-MS-AC*, respectivement sur Dev et Test.

Pour plus de lisibilité dans les noms de système, il sera nommé désormais *DMLP-MS-AC-*$Emb_{CNN}$,

| Architecture/ Approche | Représentation de phrase | Label erreur | | | Global |
|---|---|---|---|---|---|
| | | P | R | F | CER |
| CRF ⊕ pros *(baseline)* | - | 0,686 | 0,541 | 0,605 | 8,46 |
| DMLP-MS-AC | - | 0,702 | 0,613 | 0,654 | 7,75 |
| DMLP-MS-AC Inject H3 | $Emb_{DBOW}$ | 0,694 | 0,626 | 0,658 | 7,78 |
| | $Emb_{CNN-Ph}$ | 0,724 | 0,574 | 0,640 | 7,72 |
| DMLP-MS-AC Inject O | $Emb_{DBOW}$ | 0,699 | 0,626 | 0,661 | 7,70 |
| | $Emb_{CNN-Ph}$ | 0,697 | **0,634** | **0,664** | **7,68** |
| DMLP-MS-AC Tune | $Emb_{DBOW}$ | **0,724** | 0,573 | 0,640 | 7,72 |
| | $Emb_{CNN-Ph}$ | **0,724** | 0,578 | 0,643 | 7,69 |

TABLE 7.6 – Performance des *embeddings* de phrases $Emb_{DBOW}$ et $Emb_{CNN-Ph}$ sur Test

#### 7.2.1.5 Discussion

Nous avons montré que l'ajout de l'information extraite au niveau de la phrase a apporté une amélioration en terme de taux d'erreurs de classification. Afin de confirmer les hypothèses abordées dans la partie discussion 7.1.4, nous reportons dans cette section les résultats des analyses d'erreurs faites sur les sorties du système

*DMLP-MS-AC-Emb$_{CNN}$*. Nous nous sommes intéressés aux trois critères d'analyse suivants : les mots outils/non-outils, l'empan moyen de l'erreur et la détection d'erreurs isolées.

Les résultats des analyses en fonction des mots outils/non outils sont résumés dans le tableau 7.7.

| **Mots** | **Label erreur** | |
|---|---|---|
| | **P** | **R** |
| Non outils | **0,763** | **0,674** |
| Outils | 0,693 | 0,542 |

TABLE 7.7 – Rappel et précision pour la détection de mots erronés en fonction des mots outils/non sur Dev.

Ces résultats montrent que l'ajout de l'information globale a amélioré la détection des mots outils de 0,9% et 2,5% respectivement en termes de précision et rappel.

Le tableau 7.8 présente la taille moyenne des empans et l'écart type pour : la vérité terrain, les prédictions et les prédictions correctes, des systèmes CRF⊕*pros*, *DMLP-MS-AC* et *DMLP-MS-AC-Emb$_{CNN}$*. Les résultats montrent que les *embeddings* de phrases ont capté une information sur l'erreur en termes de propagation : en effet l'ajout de ces *embeddings* a amélioré la taille moyenne de l'erreur, en comparaison au système *DMLP-MS-AC*. Ainsi, le système *DMLP-MS-AC-Emb$_{CNN}$* obtient des résultats très proches du CRF, en termes de taille moyenne de l'erreur pour les prédictions et les prédictions correctes.

| **Approche** | | **Taille moyenne de l'empan** | **Écart type** |
|---|---|---|---|
| | vérité terrain | 3,24 | 2,15 |
| CRF⊕*pros (baseline)* | prédictions | 3,28 | 1,77 |
| | prédictions correctes | 2,88 | 1,34 |
| *DMLP-MS-AC* | prédictions | 2,82 | 1,28 |
| | prédictions correctes | 2,66 | 1,05 |
| *DMLP-MS-AC-Emb$_{CNN}$* | prédictions | 3,15 | 1,70 |
| | prédictions correctes | 2,84 | 1,22 |

TABLE 7.8 – L'empan moyen et l'écart-type des prédictions et prédictions correctes des systèmes CRF⊕*pros*, *DMLP-MS-AC* et *DMLP-MS-AC-Emb$_{CNN}$* sur Dev.

Enfin, les résultats de la détection d'erreurs isolées sont résumés dans le tableau 7.9. On observe que l'ajout de l'information globale n'a pas apporté une amélioration : la précision augmente tandis que le rappel diminue.

### 7.2.2 Modèle contextuel probabiliste pour une décision globale

Une autre approche pour compenser des lacunes remarquées lors de notre analyse d'erreurs de détection (section 7.1.3) consiste à utiliser un modèle contextuel

| **Approches** | **Label erreur** | |
|---|---|---|
| | **P** | **R** |
| *DMLP-MS-AC* | 0,530 | 0,515 |
| *DMLP-MS-AC-Emb$_{CNN}$* | **0,567** | 0,459 |

TABLE 7.9 – Détection d'erreurs isolées du système *DMLP-MS-AC*, sans et avec ajout de l'information globale sur Dev.

probabiliste qui porte des informations sur la distribution d'erreurs. Nous espérons ici corriger le problème de la taille de l'empan d'erreurs mal capturée par notre système de détection.

Cette approche est similaire à celle utilisée par [Dufour *et al.*, 2014] pour la détection automatique de segments de parole spontanée dans des émissions télévisées. Les auteurs ont proposé d'étendre un processus de classification locale à l'aide d'un modèle contextuel probabiliste d'étiquetage de séquences qui prend en compte l'étiquetage (parole préparée *vs.* parole spontanée) des segments voisins dans une fenêtre de taille 3. Grâce à cette extension, l'étiquetage, qui était issu d'une succession de décisions locales, devient un processus global.

Nous proposons d'appliquer cette idée à notre approche pour la détection d'erreur. Jusqu'à présent, l'étiquetage en *erreur vs. correct* des transcriptions automatiques par notre approche neuronale consistait en autant de classifications indépendantes que de mots à étiqueter. En tenant compte des classifications locales des mots voisins dans une fenêtre contextuelle de taille 5 identique à celle de l'entrée de notre système de détection d'erreurs, nous espérons lisser au niveau de la phrase le résultats de ces classifications. Pour cela, un modèle probabiliste d'ordre $n$ de distribution d'erreurs est utilisé : ce modèle estime la probabilité que le mot courant soit erroné en fonction de la justesse des 4 mots qui l'entourent.

#### 7.2.2.1 Formalisation

Étant donné une étiquette $e_i$ du mot $w_i$, avec $e_i \in \{correct, erreur\}$, nous définissons $\mathrm{P}(e_i|e_{i-2},e_{i-1},e_{i+1},e_{i+2})$ la probabilité d'observer un mot $w_i$ étiqueté comme $e_i$ pendant que les mots précédents et suivants sont étiquetés respectivement en $e_{i-2}$, $e_{i-1}$, $e_{i+1}$ et $e_{i+2}$ ; $\mathrm{c}(e_i)$ est la mesure de confiance fournie par le système de détection d'erreurs lors du choix de l'étiquette $e_i$ pour le mot $w_i$. $S$ est la séquence d'étiquettes affectée à la séquence de mots (une seule étiquette par mot).

Le processus de décision globale consiste à choisir la séquence d'étiquettes $\overline{S}$ qui maximise le score global obtenu en combinant $\mathrm{c}(e_i)$ et $\mathrm{P}(e_i|e_{i-2},e_{i-1},e_{i+1},e_{i+2})$, pour chaque mot dans les transcriptions automatiques. La séquence $\overline{S}$ est calculée en utilisant l'équation suivante :

$$\overline{S} = \arg\max_{e} \prod_{i=1}^{n} c(e_i)^{\lambda} \times P(e_i|h_i) \tag{7.1}$$

où $h_i = e_{i-2}, e_{i-1}, e_{i+1}, e_{i+2}$, $n$ est le nombre de mots dans la transcription automatique et $\lambda$ est un coefficient de pondération pour contrôler l'apport de $c(e_i)$ et de $P(e_i|h_i)$.

#### 7.2.2.2 Résolution de l'équation : chemin de poids optimal dans un graphe

En pratique, l'équation 7.1 peut se résoudre par la recherche du chemin de poids optimal dans un graphe. Pour cela, nous utilisons le paradigme des transducteurs à états finis (*Finite State Machine* - FSM), comme présenté dans [Mohri *et al.*, 2002].

Nous représentons le modèle portant les probabilités P($e_i$|$e_{i-2}$,$e_{i-1}$,$e_{i+1}$,$e_{i+2}$) pour tous les cinq-uplets ($e_{i-2}$,$e_{i-1}$,$e_i$,$e_{i+1}$,$e_{i+2}$) sous le format d'un transducteur nommé *Mod*. Un autre transducteur, *Hyp*, porte les valeurs $c(e_i)$ fournies par notre système de détection locale d'erreurs.

Leurs topologies ont été conçues pour capturer les contraintes contextuelles (une fenêtre de 5 mots) que nous souhaitons modéliser et pour permettre leur composition [Mohri *et al.*, 2002].

Le résultat de cette composition est un nouveau transducteur *Hyp* $\circ$ *Mod* qui représente l'espace de recherche contenant toutes les séquences d'étiquettes *erreur/correct* possibles. La séquence $\overline{S}$ est le chemin de poids optimal dans cet espace.

#### 7.2.2.3 Résultats et discussions

Nous avons utilisé la boite à outils *OpenFst*[1] pour créer les FSM *Mod* et *Hyp* respectivement sur les transcriptions automatiques du corpus ETAPE et les sorties de deux systèmes de détection : le système de base *DMLP-MS-AC* et le système *DMLP-MS-AC-Emb*$_{CNN}$ qui intègre des connaissances sur la phrase. Nous avons utilisé le corpus d'apprentissage pour construire le FSM *Mod*, et le corpus de développement pour optimiser le poids de pondération $\lambda$. Le corpus de test est ainsi utilisé pour évaluer la performance de cette approche. Les systèmes résultants sont nommés avec l'extension *-FSM* . Les résultat obtenus par l'approche globale pour la détection d'erreurs sont résumés dans le tableau 7.10.

On observe que l'application de l'approche globale aux sorties du système *DMLP-MS-AC* permet une légère diminution du CER tant sur Dev que sur Test. Le CER du *DMLP-MS-AC-FSM* est comparable à celui obtenu par le système *DMLP-MS-AC-Emb*$_{CNN}$ qui intègre des informations globales sur la phrase.

L'application de l'information globale sur les sorties de *DMLP-MS-AC-Emb*$_{CNN}$ n'a amélioré que légèrement les résultats sur Dev. Cela peut s'expliquer par le fait que ce système intègre déjà des connaissances globales sur la phrase, et l'information apportée par l'approche globale est redondante.

Nous avons analysé les sorties des deux systèmes *DMLP-MS-AC-FSM* et *DMLP-MS-AC-Emb*$_{CNN}$*-FSM* selon l'empan moyen de l'erreur, pour voir si cette approche a amélioré la modélisation de l'effet "boule de neige" de l'erreur. Comme

1. http ://www.openfst.org/twiki/bin/view/FST/WebHome

| Corpus | Approche | Label erreur | | | Globale |
|---|---|---|---|---|---|
| | | **P** | **R** | **F** | **CER** |
| Dev | CRF ⊕ pros *(baseline)* | 0,698 | 0,550 | 0,615 | 10,12 |
| | *DMLP-MS-AC* | 0,717 | **0,599** | 0,653 | 9,38 |
| | *DMLP-MS-AC-Emb$_{CNN}$* | 0,724 | 0,598 | **0,655** | 9,26 |
| | *DMLP-MS-AC-FSM* | **0,734** | 0,575 | 0,645 | 9,31 |
| | *DMLP-MS-AC-Emb$_{CNN}$-FSM* | 0,728 | 0,594 | 0,654 | **9,23** |
| Test | CRF ⊕ pros *(baseline)* | 0,686 | 0,541 | 0,605 | 8,46 |
| | *DMLP-MS-AC* | 0,702 | **0,613** | **0,654** | 7,75 |
| | *DMLP-MS-AC-Emb$_{CNN}$* | 0,724 | 0,578 | 0,643 | 7,69 |
| | *DMLP-MS-AC-FSM* | 0,718 | 0,591 | 0,649 | **7,67** |
| | *DMLP-MS-AC-Emb$_{CNN}$-FSM* | 0,726 | 0,573 | 0,641 | 7,69 |

TABLE 7.10 – Performance de l'approche globale pour la détection d'erreurs.

présenté dans le tableau 7.11, cette approche a apporté une amélioration pour les prédictions et les prédictions correctes par rapport aux résultats de *DMLP-MS-AC*.

| **Approche** | | **Taille moy. de l'empan** | **Écart type** |
|---|---|---|---|
| | vérité terrain | 3,24 | 2,15 |
| CRF⊕*pros (baseline)* | prédictions | 3,28 | 1,77 |
| | prédictions correctes | 2,88 | 1,34 |
| *DMLP-MS-AC* | prédictions | 2,82 | 1,28 |
| | prédictions correctes | 2,66 | 1,05 |
| *DMLP-MS-AC-FSM* | prédictions | 2,99 | 1,52 |
| | prédictions correctes | 2,73 | 1,17 |
| *DMLP-MS-AC-Emb$_{CNN}$* | prédictions | 3,15 | 1,70 |
| | prédictions correctes | 2,84 | 1,22 |
| *DMLP-MS-AC-Emb$_{CNN}$-FSM* | prédictions | 3,14 | 1,73 |
| | prédictions correctes | 2,85 | 1,24 |

TABLE 7.11 – L'empan moyen et l'écart-type des prédictions et prédictions correctes sur Dev, avant et après application de l'information globale sur les sorties des systèmes neuronaux.

On remarque que malgré les améliorations apportées à nos systèmes, le système CRF produit des empans dont la taille moyenne reste la plus proche de la vérité terrain pour les prédictions et les prédictions correctes.

## 7.3 Conclusion

Dans ce chapitre, nous avons tout d'abord présenté une étude sur l'analyse d'erreurs des sorties du système *DMLP-MS-AC* qui intègre plusieurs sources d'informations : les *embeddings* linguistiques et acoustiques, les descripteurs syntaxiques, lexicaux et prosodiques. Ces analyses ont été réalisées en fonction de plusieurs critères : longueur du mot, mots outils/non-outils, empan moyen de l'erreur et nombre d'erreurs dans le contexte.

En s'appuyant sur ces résultats, nous avons proposé d'utiliser des informations qui caractérisent la phrase afin de réduire les limites de notre approche de détection d'erreurs, qui ne traite chaque mot que dans un contexte local. Pour cela, nous avons exploré deux approches. La première consiste à intégrer des connaissances sur la phrase, en utilisant des *embeddings* de phrases. Deux solutions différentes ont été étudiées pour calculer ces *embeddings* de phrases. La première repose sur l'utilisation de l'architecture *DBOW* fourni par l'outil *Doc2vec*, et la deuxième consiste à extraire les *embeddings* d'un réseau de neurones convolutif appris pour la classification des phrases transcrites automatiquement en *peu erronées* ou *très erronées*, selon la proportion d'erreur que contient chaque phrase reconnue.

Nous avons montré que l'intégration de ces deux *embeddings* apportent une amélioration, les meilleurs résultats étant obtenus en utilisant le deuxième type d'*embeddings*.

Nous avons également montré que l'intégration des *embeddings* de phrases a légèrement amélioré la détection des mots non-outils d'une part, et la taille moyenne de l'empan de l'erreur d'autre part, ce qui montre que ces *embeddings* ont capturé une information sur la sémantique et sur la propagation d'erreurs.

La deuxième approche repose sur l'utilisation d'un modèle contextuel probabiliste de la distribution d'erreurs. Celle-ci a été appliquée à deux systèmes, le système de base *DMLP-MS-AC* et son extension intègrant les *embeddings* de phrase $Emb_{CNN-Ph}$. Cette approche diminue également légèrement le taux d'erreur de classification, surtout lorsqu'elle est appliquée aux sorties du système de base. La taille moyenne de l'empan d'erreurs est également plus proche de la vérité terrain.

En conclusion, les pistes d'amélioration émises à la suite de l'analyse qualitative des détections réalisées par nos premiers systèmes ont eu un impact limité, mais positif. La distribution d'erreurs, mesurée à travers la notion d'empan, est plus convaincante.

CHAPITRE 8

# CADRES APPLICATIFS

## Sommaire

Nous présentons dans ce chapitre différentes applications exploitant les *embeddings* et les sorties des systèmes de détection d'erreurs de reconnaissance de la parole présentés dans les chapitres précédents. Nous commençons d'abord par exploiter ces sorties comme des mesures de confiance et nous les évaluons dans ce cadre. Ensuite, nous proposons d'utiliser ces mesures de confiance pour améliorer les performances d'un système de compréhension de la parole. Enfin, nous nous intéressons à l'exploitation des *embeddings* linguistiques et acoustiques dans un cadre de prédictions des erreurs possibles pour un mot du vocabulaire, et dans un cadre de post-édition manuelle des transcriptions automatiques.

## 8.1 Évaluation d'une mesure de confiance

### 8.1.1 Définition

Jusqu'alors l'objectif principal de nos travaux était la détection d'erreurs. Nous proposons ici d'étendre l'usage des sorties de nos systèmes de détection d'erreurs au calcul de mesures de confiance.

Comme nous l'avons vu dans la section 3.1, une mesure de confiance a pour rôle d'évaluer la fiabilité d'un mot reconnu, et devrait idéalement fournir une valeur comprise entre 0 et 1 qui correspond à la probabilité que le mot reconnu soit correct [Mauclair, 2006].

À partir des architectures neuronales pour la détection d'erreur présentées dans les chapitres précédents, il est facile de proposer une mesure de confiance : elle correspond à la valeur numérique liée à l'étiquette *Correct* de notre système de détection d'erreurs. Celle-ci est comprise entre 0 et 1, puisqu'elle est issue de la couche $Softmax$ en sortie de réseau.

Notons que cette mesure est calculée à partir de la combinaison d'un ensemble de descripteurs adaptés à la tâche de la détection d'erreurs, dont une mesure de confiance classique comme la probabilité *a posteriori* (PAP). Il est donc aussi possible de considérer notre approche comme une approche de calibration de la probabilité *a posteriori*, qui consiste à améliorer le caractère prédictif de cette mesure de confiance [Yu *et al.*, 2011].

Nous proposons d'évaluer les performances de notre mesure de confiance et de les comparer à celles de la probabilité *a posteriori* calculée par le SRAP.

### 8.1.2 Évaluation en termes de valeurs de NCE

Nous proposons d'évaluer en terme d'entropie croisée normalisée (NCE) les valeurs proposées en sorties des systèmes neuronaux et CRF et utilisées comme mesures de confiance. La métrique NCE est parmi les métriques d'évaluation les plus utilisées pour mesurer la qualité des mesures de confiance. Sa valeur est une estimation de l'information additionnelle que la mesure de confiance apporte à l'hypothèse de reconnaissance. Plus la valeur de NCE se rapproche de 1 plus la mesure de confiance

est prédictive. Une description détaillée de cette métrique est présentée dans la section 3.1.5.

Nous souhaitons évaluer quatre de nos systèmes neuronaux :

— MLP-MS-AutoE : le système présenté dans le chapitre 5 qui intègre les descripteurs décrits dans le chapitre 4 et les *embeddings* combinés avec auto-encodeur.

— MLP-MS-AutoE ⊕ pros : le système MLP-MS-AutoE enrichi par des descripteurs prosodiques.

— DMLP-MS-AC : le système MLP-MS-AutoE enrichi par des descripteurs prosodiques et, sur un nouveau flux, enrichi par des *embeddings* acoustiques.

— DMLP-MS-AC-Emb$_{CNN}$ : le système MLP-MS-AC enrichi par des *embeddings* de phrase issus d'un réseau de neurones convolutif.

Les valeurs de NCE, obtenues par les différents systèmes sur Dev et Test (*cf.* section 4.2.2), sont présentés dans les tableaux 8.1 et 8.2.

| **Sources de mesures de confiance** | **NCE** |
|---|---|
| PAP | 0,134 |
| CRF *(baseline)* | 0,448 |
| *CRF ⊕ pros* | 0,449 |
| MLP-MS-AutoE (1) | 0,427 |
| MLP-MS-AutoE ⊕ pros (2) | 0,459 |
| DMLP-MS-AC (3) | 0,440 |
| DMLP-MS-AC-Emb$_{CNN}$ (4) | 0,478 |

Table 8.1 – Valeurs de NCE pour les mesures de confiance sur Dev

| **Sources de mesures de confiance** | **NCE** |
|---|---|
| PAP | 0,177 |
| CRF *(baseline)* | 0,463 |
| *CRF ⊕ pros* | 0,462 |
| MLP-MS-AutoE (1) | 0,442 |
| MLP-MS-AutoE ⊕ pros (2) | 0,477 |
| DMLP-MS-AC (3) | 0,455 |
| DMLP-MS-AC-Emb$_{CNN}$ (4) | 0,489 |

Table 8.2 – Valeurs de NCE pour les mesures de confiance sur Test

Ces résultats montre l'intérêt de l'approche proposée pour produire une mesure de confiance bien calibrée, alors que la probabilité *a posteriori* fournie par le SRAP est clairement mal calibrée. Les meilleures performances sont atteintes par le système (4) qui intègre tous les descripteurs proposés. On note également que la mesure de confiance fournie par le système CRF est également bien calibrée.

L'utilisation des caractéristiques prosodiques (2) permet d'obtenir une valeur de la NCE plus élevée pour le système neuronal tant sur Dev que sur Test. En revanche, l'ajout des *embeddings* acoustiques (3) a réduit la valeur de la NCE.

Dans cette expérience, les systèmes neuronaux ont été optimisés sur la tâche de détection d'erreurs. Nous pensons pouvoir améliorer ces résultats avec des systèmes optimisés pour maximiser la NCE. Les valeurs de la NCE obtenues par ces systèmes neuronaux optimisés sur le score NCE sont présentés dans le tableau 8.3.

| **Corpus** | **Sources de mesures de confiance** | **NCE** |
|---|---|---|
| Dev | MLP-MS-AutoE (5) | 0,445 |
| | MLP-MS-AutoE $\oplus$ pros (6) | 0,475 |
| | DMLP-MS-AC (7) | 0,482 |
| | DMLP-MS-AC-Emb$_{CNN}$ (8) | **0,483** |
| Test | MLP-MS-AutoE (5) | 0,460 |
| | MLP-MS-AutoE $\oplus$ pros (6) | 0,489 |
| | DMLP-MS-AC (7) | 0,491 |
| | DMLP-MS-AC-Emb$_{CNN}$ (8) | **0,492** |

TABLE 8.3 – Valeurs de NCE pour les mesures de confiances sur Dev et Test, obtenues par les systèmes neuronaux optimisés sur le score NCE.

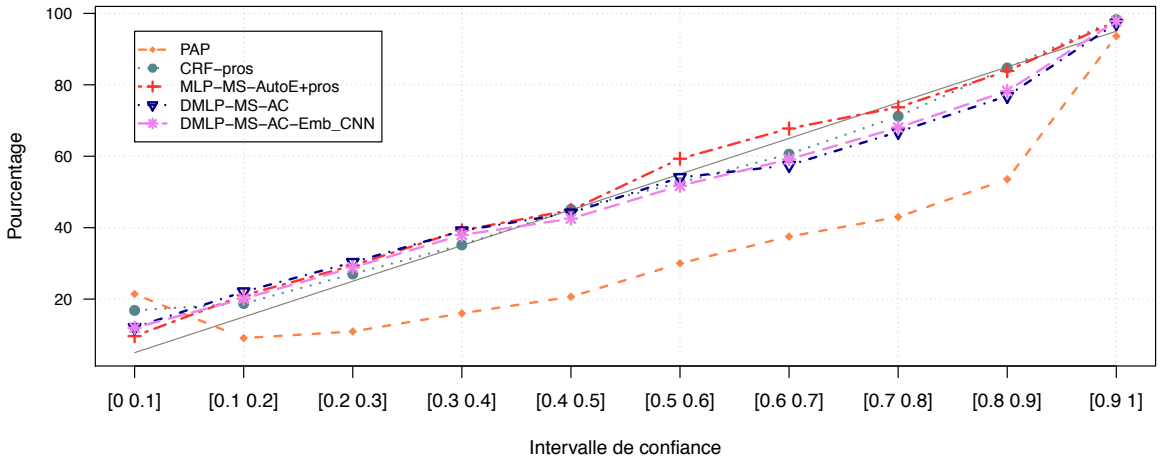
Ainsi, nous pouvons constater que cette optimisation a bien entendu amélioré la valeur de la NCE sur Dev, mais aussi sur Test. Sur Test, le gain obtenu sur le meilleur système est moins probant que pour les autres. Nous pouvons remarquer que nous perdons en termes de CER, par exemple le système (7) qui obtient 9,39% et 7,83% respectivement sur Dev et Test au lieu de 9,38% et 7,75% (le système 3).

### 8.1.3 Évaluation par intervalle de confiance

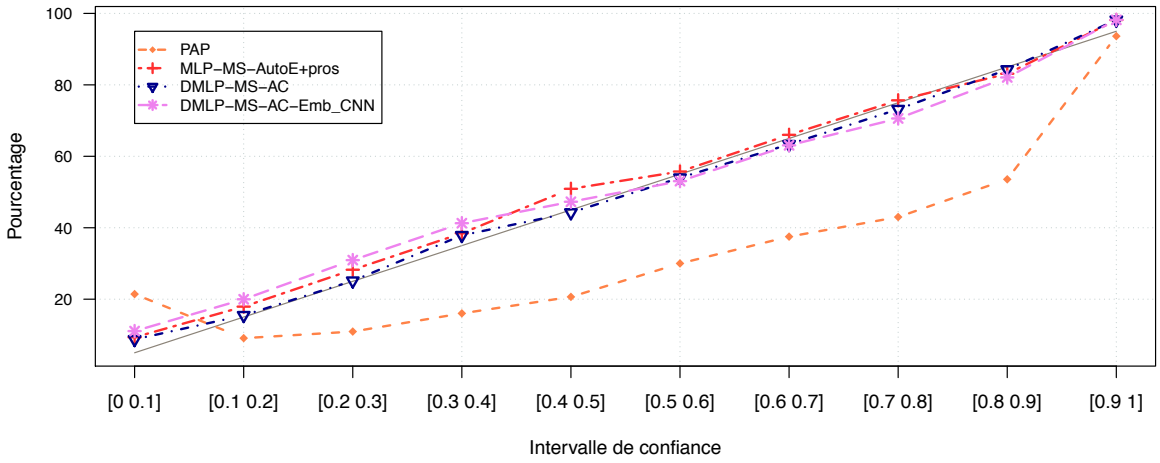
Nous présentons maintenant des résultats qui permettent d'apprécier graphiquement la bonne calibration des mesures de confiance représentant le pourcentage des mots corrects en fonction des mesures de confiance. Dans ce cadre, la bonne calibration s'apprécie par le fait qu'il faut être très proche de la diagonale. Par exemple, pour une mesure de confiance 0.65, il faudrait trouver 65% des mots corrects.

La figure 8.1 présente le pourcentage de mots corrects en fonction de différentes mesures d'évaluation : PAP, mesures issues des systèmes neuronaux, et CRF. Le graphique (a) porte sur les systèmes optimisés sur le score CER alors que le graphique (b) indique ceux optimisés sur le score NCE. On remarque que la corrélation entre les probabilités dérivées de nos systèmes neuronaux ou CRF est très forte avec le pourcentage des mots corrects. Les courbes sont bien alignées avec la diagonale, en particulier pour les systèmes optimisés sur le score NCE. La correspondance entre la valeur de pourcentage de mots corrects par rapport à la mesure de confiance est particulièrement bien observée dans l'intervalle [0,7 ; 1]. Il est intéressant de noter que cet intervalle recouvre plus de 90% de mots sur Dev.

Nous avons montré que notre système permet d'obtenir une mesure de confiance bien calibrée. Cela peut avoir un intérêt dans plusieurs cadres applicatifs, et notamment pour des applications comme la détection d'entités nommées ou la compréhension de la parole. Ce sera l'objet de la section suivante.




(a)




(b)

FIGURE 8.1 – Pourcentage de mots corrects en fonction de la PAP et des mesures de confiances issues des systèmes neuronaux et CRF. (a) les systèmes sont optimisés sur le score CER . (b) les systèmes neuronaux sont optimisés sur le score NCE.

## 8.2 Gestion d'erreurs pour la compréhension de la parole

Malgré les avancées récentes, les systèmes d'interprétation sémantique du langage écrit comme oral sont encore largement perfectibles. Parmi les problèmes communs au traitement du langage écrit et au langage oral, nous pouvons citer la difficulté

de localiser des mots supports de concepts, les ambiguïtés de leur catégorisation, ou encore la difficulté d'identifier des contraintes contextuelles suffisantes pour résoudre ses ambiguïtés.

Des difficultés supplémentaires sont introduites lorsqu'un système de compréhension de la parole (SLU) analyse des transcriptions automatiques qui comportent des erreurs de reconnaissance de la parole. Ces erreurs peuvent affecter les mots supports d'un concept, rendant difficile la détection du concept et l'extraction de la valeur d'une instance de ce concept. En outre, la prédiction des concepts et des valeurs dépend aussi du contexte dans lequel leurs supports sont localisés : des erreurs de reconnaissance de la parole présentes dans le contexte peuvent également introduire des erreurs dans la détection de ces supports de concept, ainsi que pour leur catégorisation.

Nous avons voulu mesurer l'impact de l'utilisation des sorties d'un système de détection d'erreurs de reconnaissance de la parole pour apporter des connaissances supplémentaires concernant la fiabilité des mots analysés à un système d'interprétation sémantique. Dans notre étude, l'interprétation sémantique consiste à détecter des concepts liées à une application de réservation d'hôtel par téléphone qui s'appuie sur un dialogue humain-machine. Il s'agit ici de détecter les mots supports de concept, d'associer chaque support à un des concepts prédéfinis, et d'extraire la valeur liée à l'occurrence de ce concept. Pour cela, nous appliquons la méthode qui consiste à étiqueter chaque mot avec une étiquette sémantique, ou bien une étiquette *null* lorsque ce mot n'est pas porteur d'un concept de l'application.

L'approche que nous proposons consiste à enrichir l'ensemble des étiquettes sémantiques par des étiquettes spécifiques à l'erreur, mais aussi d'exploiter la mesure de confiance issue du système MLP-MS-AutoE. Cette mesure est utilisée comme descripteur additionnel dans le système SLU : nous faisons l'hypothèse qu'elle apporte une aide pour la localisation des erreurs de reconnaissance automatique de la parole qui pourraient affecter les performances de ce système.

Il est à noter que le système SLU a été implémenté par mon collègue doctorant Edwin Simonnet, à partir du logiciel *nmtpy* développé au LIUM [Martinez *et al.*, 2015].

### 8.2.1 Module de compréhension de la parole d'un système de dialogue

Comme nous venons de le voir, le système SLU utilisé consiste à extraire automatiquement, à partir de transcriptions automatiques, des couples de concepts-valeurs qui sont spécifiques à une application, ici un système d'information touristique. Le système SLU utilisé, que nous présentons dans ce chapitre, s'appuie sur une architecture neuronale qui intègre un ensemble de descripteurs.

#### 8.2.1.1 Architecture

L'architecture du système SLU, illustrée dans la figure 8.2, s'appuie sur un réseau de neurones récurrent (RNN) de type encodeur-décodeur avec mécanisme d'attention (RNN-EDA), similaire à celle utilisée pour la traduction automatique proposé dans [Cho *et al.*, 2014b].

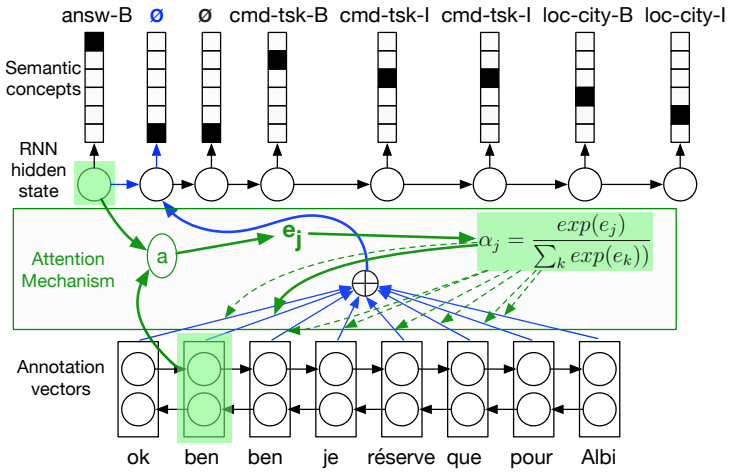



FIGURE 8.2 – Architecture du système de compréhension de la parole RNN-EDA.

Ici, le processus d'étiquetage en concepts sémantiques est considéré comme un problème de traduction depuis des mots (langage source) vers des concepts sémantiques (langage cible).

Dans cette architecture, l'encodeur est un RNN bidirectionnel qui s'appuie sur des GRU (Gated Recurrent Units), que nous avons présentés dans la section 1.1.2.3. Cet encodeur calcule une annotation $h_i$ pour chaque mot $w_i$ de la séquence d'entrée $w_1, ..., w_n$. Cette annotation est la concaténation des couches cachées avant (*forward*) et arrière (*backward*) qui se trouve à la même position. Elles sont obtenues respectivement par le RNN avant et le RNN arrière constituant le RNN bidirectionnel.

À partir de l'encodeur bidirectionnel appliqué pour chaque mot dans la séquence d'entrée, la séquence d'annotations résultante $h_1, ..., h_n$ est utilisée par le décodeur pour calculer un vecteur de contexte (représenté par un cercle avec une croix dans la figure 8.2). Ce vecteur de contexte est recalculé après chaque émission d'une étiquette de sortie. Ce calcul prend en compte une somme pondérée de toutes les annotations calculées par l'encodeur. Cette pondération dépend de la cible courante en sortie et constitue le cœur du mécanisme d'attention. En effet, une bonne estimation des poids $\alpha_{tj}$ permet au décodeur de choisir les parties de la séquence d'entrée les plus pertinentes pour émettre la prochaine étiquette sémantique. Ce vecteur de contexte est utilisé par le décodeur, un RNN qui émet les étiquettes sémantiques, conjointement avec la couché cachée précédente de ce RNN pour prendre une décision sur la

sortie de l'étiquette actuelle.

Une description plus détaillée du RNN avec mécanisme d'attention est donnée dans [Simonnet *et al.*, 2015].

#### 8.2.1.2 Descripteurs

L'architecture RNN-EDA intègre un ensemble de descripteurs représentant le mot, pour prédire l'étiquette sémantique associée à ce mot. Cet ensemble est composé des descripteurs suivants :

— le mot lui-même ;

— sa catégorie sémantique parmi un ensemble prédéfini appartenant aux :

- catégories spécifiques à l'application visée comme : noms des rues, des villes ou des hôtels, liste des équipements de la salle, type de nourriture du restaurant de l'hôtel, ...
- catégories plus générales : chiffres, jours, mois, ...

— un ensemble de descripteurs syntaxiques. L'outil MACAON [Nasr *et al.*, 2011] est appliqué sur les transcriptions du tour de parole afin d'obtenir pour chaque mot son lemme, son étiquette syntaxique (POS), son mot gouverneur et le lien de dépendance entre le mot et son gouverneur ;

— les 1-à-4 premiers et derniers n-grammes de lettres du mot ;

— un descripteur binaire qui indique si la première lettre est en majuscule ;

— les deux mesures de confiance (MC) suivantes : la probabilité *a posteriori* (PAP) et la mesure de confiance (MC-MLP-MS) fournie par le système de détection d'erreurs (MLP-MS).

Dans nos expérimentations, l'architecture RNN-EDA prend en entrée tous ces descripteurs, à l'exception des deux mesures de confiance qui peuvent toutes être prises, seulement en partie, ou pas du tout : aucune, PAP, MC-MLP-MS ou les deux.

### 8.2.2 Protocole expérimental et résultats

Nos expériences ont été réalisées sur les données MEDIA [Bonneau-Maynard *et al.*, 2005].

#### 8.2.2.1 Corpus Media

Le corpus MEDIA a été collecté durant le projet français Media/Evalda [Bonneau-Maynard *et al.*, 2005] et concerne un service touristique. Il contient trois ensembles de dialogues téléphoniques humain/machine simulés : le corpus d'apprentissage (Train), de développement (Dev) et de test (Test), composés respectivement de 17,7k, 1,3k et 3,5k tours de parole utilisateurs. Le corpus est annoté manuellement avec des concepts sémantiques caractérisés par un nom et sa valeur.

#### 8.2.2.2 SRAP LIUM

Le corpus MEDIA est enrichi avec des transcriptions automatiques générées par une variante du SRAP développé au LIUM qui a gagné la dernière campagne d'évaluation en français [Rousseau *et al.*, 2014]. Ce système est construit autour de la boîte à outils de reconnaissance de la parole Kaldi [Povey *et al.*, 2011]. Les modèles acoustiques, fondés sur l'approche DNN/HMM, ont été estimés sur 145,781 segments de parole issus de données provenant des corpus ESTER [Galliano *et al.*, 2006], ESTER2 [Galliano *et al.*, 2009], ETAPE [Gravier *et al.*, 2012], REPERE [Galibert et Kahn, 2013] et diverses sources audio disponibles au LIUM avec des transcriptions manuelles, composant un ensemble de 565 heures de parole. Ces enregistrements ont été convertis en 8 kHz afin d'apprendre des modèles acoustiques plus appropriés aux données MEDIA. Le DNN prend en entrée les MFCCs (*Mel-Frequency Cepstrum Coefficients*) concaténés avec les *i*-vecteurs, afin d'adapter les modèles acoustiques aux locuteurs [Gupta *et al.*, 2014].

Le vocabulaire du SRAP contient 2,5K mots provenant des corpus d'apprentissage et de développement MEDIA. Un premier modèle de langage (LM) bi-gramme est appliqué pendant le processus de décodage pour générer des graphes de mots. Ensuite, un modèle de langage 3-gramme est appliqué aux graphes pour recalculer les scores des différents chemins du graphe et en extraire l'hypothèse la plus probable. Pour éviter de faire face aux erreurs commises par les LMs surentrainés sur le corpus d'apprentissage MEDIA, nous avons suivi une approche de type *leave-one-out* : l'ensemble des fichiers de dialogues dans les corpus d'apprentissage et de développement a été divisé aléatoirement en 4 sous-ensembles. Chaque sous-ensemble est transcrit en utilisant des LMs (bi-gramme et tri-gramme) entrainés sur les transcriptions manuelles présentes dans les 3 autres sous-ensembles. Ces LMs sont linéairement interpolés à un modèle de langage "générique" entrainé sur un grand ensemble de journaux français, contenant 77 millions de mots. Les données de test sont transcrites avec des LMs appris sur le corpus d'apprentissage MEDIA et le même modèle de langage générique. Comme le montre le tableau 8.4, les taux d'erreurs de mots pour les corpus d'apprentissage, de développement et de test sont d'environ 23,5%.

| Train | Dev | Test |
|---|---|---|
| 23,7% | 23,4% | 23,6% |

TABLE 8.4 – Taux d'erreurs mots des transcriptions automatiques.

#### 8.2.2.3 Qualité des mesures de confiance pour la reconnaissance de la parole sur MEDIA

Nous avons utilisé le système de détection d'erreurs MLP-MS décrit dans la section 4 pour calculer des mesures de confiance sur les mots. Nous rappelons que

ce système intègre plusieurs sources d'informations : les *embeddings* linguistiques combinés avec auto-encodeur, des descripteurs syntaxiques, lexicaux ainsi que des informations contextuelles extraites des mots voisins.

Afin de pouvoir générer des mesures de confiance pour tous les corpus MEDIA, nous avons appris le système de détection d'erreurs de deux manières différentes :

— Sys1 : MLP-MS-AutoE est appris sur le corpus Train, optimisé sur Dev et appliqué à Test.

— Sys2 : pour ce système nous avons suivi une approche de type *leave-one-out* : les corpus Train et Dev sont divisés en 4 sous-ensembles de la même manière que le SRAP est appris. Le système MLP-MS-AutoE est entrainé alors sur 3 parties, et appliqué sur la partie restante. Puis les parties qui ne sont pas utilisées pour l'apprentissage sont concaténées pour former Train et Dev.

À partir de Sys1 on obtient les sorties du détecteur d'erreurs pour Test, et à partir de Sys2 celles pour Train et Dev.

Comme nous l'avons vu précédemment, ces sorties sont utilisées comme mesures de confiance, en utilisant les scores de l'étiquette *Correct* calculés par la valeur de la fonction *Softmax* du système MLP-MS-AutoE.

Afin d'évaluer dans ce contexte les qualités prédictives des mesures de confiance fournies par notre système de détection d'erreurs, nous mesurons sur le corpus MEDIA la valeur NCE des mesures de confiance qu'il fournit. Le tableau 8.5 présente les scores NCE sur le Test qui confirment la calibration de la mesure MC-MLP-MS-AutoE déjà perçue en section 8.1.

| | PAP | MC-MLP-MS-AutoE |
|---|---|---|
| NCE | 0,147 | **0,462** |

TABLE 8.5 – Comparaison entre la probabilité *a posteriori* et la mesure de confiance MC-MLP-MS-AutoE sur le corpus de Test MEDIA en termes de score NCE.

De nouveau une vue plus graphique de la calibration est proposée dans la figure 8.3 qui montre la capacité prédictive de la mesure de confiance issue de MLP-MS-AutoE en comparaison à la probabilité *a posteriori* sur le corpus de Test MEDIA. La courbe illustre le pourcentage de prédiction de mots corrects en fonction des intervalles de confiance, et confirme l'efficacité de la mesure produite par MC-MLP-MS-AutoE.

#### 8.2.2.4 Apport des mesures de confiance pour l'étiquetage sémantique

Le système de compréhension de la parole présenté dans la section 8.2.1.1 est entrainé sur les transcriptions automatiques afin d'utiliser un corpus d'apprentissage bruité par les erreurs de reconnaissance de manière à ce qu'il soit proche du corpus de test. Les évaluations sont effectuées sur les ensembles Dev et Test en termes de taux d'erreur de concept (CER) pour les étiquettes de concept uniquement et le taux d'erreur de concept/valeur (CVER) pour les paires concept/valeur. Comme nous

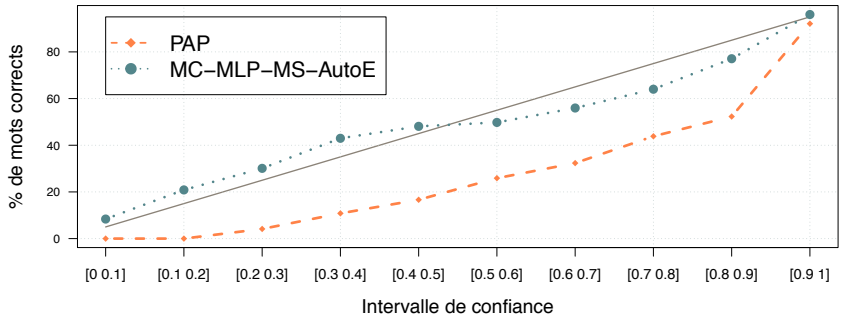



FIGURE 8.3 – Comparaison en prédiction d'erreur du SRAP de la probabilité *a posteriori* et la mesure MC-MLP-MS-AutoE sur le Test MEDIA.

l'avons brièvement évoqué plus haut, nous avons ajouté aux étiquettes de concept MEDIA habituelles deux étiquettes. Au cours de l'apprentissage, ces étiquettes remplacent celles de référence lorsque le mot est erroné. Si le mot hypothèse erroné fait partie du support de mot d'un concept, il est associé à l'étiquette *ERROR-C*, sinon *ERROR-N*. Au cours de l'évaluation, les étiquettes *ERROR-C* et *ERROR-N* sont remplacées par *null* (étiquette indiquant que le mot ne transmet aucune information MEDIA) pour appliquer le protocole d'évaluation MEDIA habituel.

Afin d'évaluer l'impact de l'utilisation des mesures de confiance, nous avons réalisé quelques expériences résumées dans le tableau 8.6.

| Sans MC | | +PAP | | +MC-MLP-MS-AutoE | | +PAP +MC-MLP-MS-AutoE | |
|---|---|---|---|---|---|---|---|
| CER | CVER | CER | CVER | CER | CVER | CER | CVER |
| 23,6 | 29,3 | 22,7 | 28,2 | 21,6 | 27,4 | **21,5** | **26,9** |

TABLE 8.6 – Impact d'utilisation des mesures de confiance (PAP et MC-MLP-MS-AutoE) sur la détection de concept sur le TEST MEDIA

Ce tableau présente les résultats sur TEST, obtenus avec la meilleure configuration observée sur DEV, en ajoutant une par une les mesures de confiance. D'une manière générale, l'ajout d'une mesure de confiance dans le système SLU est très utile et apporte une amélioration en terme de réductions des scores CER et CVER. On remarque également que, la mesure de confiance fournie par le système MLP-MS-AutoE apporte des informations pertinentes et complémentaires à celles fournies par la PAP pour améliorer la performance du système SLU en réduisant en plus les scores CER et CVER.

Le tableau 8.7 présente les résultats obtenus avec les systèmes SLU sans et avec ajout d'étiquettes erreurs en utilisant les deux mesures de confiance PAP et MC-MLP-MS-AutoE.

On observe que l'émission des étiquettes d'erreur, tout en utilisant en entrée des mesures de confiance, permet d'améliorer les performances du système SLU.

| Systèmes SLU | CER | CVER |
|---|---|---|
| sans étiquettes erreurs | 22 | 28,1 |
| avec étiquettes erreurs | **21,5** | **26,9** |

TABLE 8.7 – Impact d'utilisation des mesures de confiance (PAP et MC-MLP-MS-AutoE) sur la détection de concept sur le TEST MEDIA sans et avec ajout d'étiquettes erreurs.

## 8.3 Prédiction d'erreurs et enrichissement de réseaux de confusion

Comme nous l'avons vu dans le chapitre 2, la génération de transcriptions automatiques de la parole s'appuie à la fois sur les informations linguistiques capturées par des modèles de langage et sur les informations acoustiques portées par des modèles acoustiques. Nous pouvons donc supposer que les confusions qu'un SRAP peut générer entre plusieurs mots viennent soit de la proximité acoustique de ces mots, soit de leur proximité linguistique, et très probablement des deux.

Nous proposons une mesure de similarité qui s'appuie sur l'utilisation des *word embeddings* linguistiques et acoustiques pour prédire une liste de mots qui pourraient être substitués par un système de reconnaissance de la parole à un mot effectivement prononcé. Nous nommons cette liste une *liste de confusion*, composée des mots les plus proches du mot analysé. La mesure de similarité proposée s'appuie sur la combinaison des similarités cosinus des word embeddings de type linguistique et acoustique.

Par la suite, nous évaluons l'intérêt de la mesure de similarité proposée dans deux tâches distinctes. La première consiste à prédire les erreurs potentielles que pourrait commettre un SRAP pour des mots rares, c'est à dire des mots pour lesquels un SRAP n'a jamais commis d'erreurs connues, soit parce qu'il n'a jamais eu à reconnaître ce mot, soit parce qu'il n'existe aucune transcription manuelle ou aucune validation humaine qui aurait permis de détecter cette erreur.

La deuxième tâche consiste à enrichir les réseaux de confusion produits par un SRAP à partir de ces listes de mots potentiellement substituables par erreur. Cet enrichissement peut alors être utilisé pour l'aide à la post-édition des transcriptions automatiques, ou pour enrichir les hypothèses de reconnaissance fournies à un système de compréhension de la parole.

### 8.3.1 Une mesure de similarité combinant *word embeddings* linguistiques et acoustiques

Comme nous venons de le voir, nous proposons d'utiliser des *word embeddings* linguistiques et acoustiques de mots pour prédire des listes de confusion.

Dans les travaux présentés par la suite, les *embeddings* linguistiques, nommés $Best_{ACP}$, résultent de la combinaison de *skip-gram*, *word2vec*, et *GloVe* en utilisant l'analyse en composante principale. Nous avons montré dans le chapitre 5 que les

*embeddings* $Best_{ACP}$ obtiennent les meilleurs résultats dans la plupart des tâches d'analogie et de similarité qui s'appuient sur les similarités cosinus. Ceci justifie notre choix, puisque l'approche que nous proposons s'appuie également sur ce type de similarités.

Nous avons également utilisé les *embeddings* acoustiques de mots **a** défini dans le chapitre 6. Nous avons montré que ces *embeddings* captent plus d'informations sur la prononciation de mots que les *embeddings* d'origine purement orthographique.

Pour construire une mesure de similarité combinant des *word embeddings* de natures différentes, nous proposons d'utiliser l'interpolation linéaire des similarités cosinus linguistique et acoustique. La similarité résultante est appelée $LA_{SimInter}$, et est définie comme suit :

$$LA_{SimInter}(\lambda, w_1, w_2) = (1 - \lambda) \times L_{Sim}(w_1, w_2) + \lambda \times A_{Sim}(w_1, w_2) \tag{8.1}$$

où $w_1$ et $w_2$ sont les deux mots à comparer et $\lambda$ est le coefficient d'interpolation. Les similarités $L_{Sim}$ et $A_{Sim}$ sont calculées avec la similarité cosinus appliquée respectivement aux *embeddings* linguistiques et acoustiques de $w_1$ et $w_2$.

Comme notre objectif est de prédire ou corriger les erreurs du SRAP, nous voulons optimiser la valeur $\lambda$ à cette fin.

Pour estimer $\lambda$, une liste connue d'erreurs de substitution générées par un système de reconnaissance de la parole est utilisée. Dans cette liste, nous définissons $h$ comme étant l'hypothèse de mot erronée et $\overline{r}$ le mot de référence qui a été substitué par $h$. Pour chaque paire de mots $(h, \overline{r})$ dans la liste, nous calculons la probabilité que le mot $h$ soit reconnu lorsque le mot de référence $\overline{r}$ est erroné, *i.e.* la probabilité de substituer le mot de référence par le mot $h$ en cas d'erreur, qui est définie comme suit :

$$P(h|\overline{r}) = \frac{\#(h, \overline{r})}{\#\overline{r}}$$

où $\#(\mathrm{h}, \overline{r})$ correspond au nombre de substitutions de $\overline{r}$ par $h$ et $\#\overline{r}$ est le nombre d'erreurs sur le mot de référence $\overline{r}$.

Nous proposons alors de retenir le coefficient d'interpolation $\hat{\lambda}$ qui minimise l'erreur quadratique moyenne (MSE) entre la valeur proposée par $LA_{SimInter}(\lambda, h, \overline{r})$ et la valeur de $P(h|\ \overline{r})$. Nous définissons alors $\hat{\lambda}$ tel que :

$$\hat{\lambda} = \underset{\lambda}{\operatorname{argmin}}\, MSE(\forall(h, \overline{r}) : P(h|\overline{r}), LA_{SimInter}(\lambda, h, \overline{r})) \tag{8.2}$$

où $LA_{SimInter}(\lambda, h, \overline{r})$ et $P(h|\ \overline{r})$ sont calculés sur tous les couples $(h, \overline{r})$ possibles.

Le tableau 8.8 montre un exemple des voisins les plus proches d'un mot donné, en s'appuyant soit sur les similarités linguistiques, soit sur les similarités acoustiques, soit sur les deux. Enfin, à titre de comparaison, nous introduisons également la similarité cosinus calculée sur des représentations continues de mots obtenues par concaténation des *word embeddings* linguistique et acoustique de ces mots. Nous appellons cette dernière mesure $LA_{SimCat}$.

Dans cet exemple, les voisins d'un mot donné sont linguistiquement proches lors-

qu'ils sont induits par les *embeddings* linguistiques, et sont phonétiquement proches quand ils sont induits par les *embeddings* acoustiques de mots. En combinant les similarités linguistiques et acoustiques de mots, il est également possible de restreindre le voisinage à des mots proches d'un mot donné à la fois linguistiquement et acoustiquement.

Nous notons que la concaténation ($LA_{SimCat}$) obtient des résultats proches de ceux obtenus par la similarité linguistique.

| **Similarité** | **Liste de confusion pour le mot "portables"** |
|---|---|
| $L_{Sim}$ | téléphones, ordinateurs, portable, portatif |
| $A_{Sim}$ | portable, portant, portants, portait |
| $LA_{SimCat}$ | téléphones, ordinateurs, portable, baladeurs |
| $LA_{SimInter}$ | portable, portant, portatif, portait |

TABLE 8.8 – Exemple des voisins les plus proches (liste de confusion) du mot "portables", en s'appuyant respectivement sur les similarités linguistiques, acoustiques ou les deux.

### 8.3.2 Protocole expérimental

#### 8.3.2.1 Données

Les données expérimentales utilisées pour ces expériences sont les mêmes que celles présentées en section 4.2.2, mais le découpage utilisé est différent car l'approche retenue ne nécessite pas de corpus de développement : seules les probabilités $P(h|\overline{r})$ et le coefficient $\lambda$ doivent être calculés sur un corpus d'apprentissage et ne nécessitent pas d'autres réglages à optimiser.

La description des données expérimentales, en termes de taille, de taux d'erreurs sur les mots (WER) ainsi que le pourcentage de substitution (Sub), de suppression (Del) et d'insertion (Ins) sont présentés dans le tableau 8.9.

| **Corpus** | **#mots ref** | **#mots hyp** | **WER%** | **Sub%** | **Del%** | **Ins%** |
|---|---|---|---|---|---|---|
| Train | 399K | 364K | 25,2 | 10,3 | 11,8 | 3,1 |
| Test | 58K | 53K | 21,9 | 8,3 | 10,9 | 2,7 |

TABLE 8.9 – Description du corpus expérimental.

La liste des erreurs de substitution de Train est utilisée pour calculer le coefficient d'interpolation $\hat{\lambda}$, celle de Test est utilisée pour évaluer la performance de notre mesure de similarité dans les tâches proposées. Cette dernière liste est composée de 4678 occurrences de paires d'erreurs de substitution, nommées $Sub_{Test}$.

Les *embeddings* linguistiques sont de dimension 100 et résultent de la combinaison de *w2vf-deps*, *skip-gram*, et *GloVe*, en utilisant l'ACP comme évoqué dans

la section 8.3.1. Ces *word embeddings* ont été calculés à partir d'un grand corpus textuel composé d'environ 2 milliards de mots comme indiqué dans la section 4.2.2 du chapitre 4.

Les *embeddings* acoustiques sont également de dimension 100. Ils ont été construits sur le même vocabulaire que les *embeddings* linguistiques. Une description détaillée des données d'apprentissage et des architectures neuronales utilisées pour construire ces *embeddings* est présentée dans le chapitre 6.

#### 8.3.2.2 Tâches et métrique d'évaluation

Nous proposons deux tâches pour évaluer la performance de la mesure de similarité qui sont la prédiction d'erreurs pour les mots rares (tâche 1) et l'enrichissement de réseaux de confusion (tâche 2).

Étant donné une paire de mots $(a, b)$ dans la liste $L$ composée de $m$ erreurs de substitution, les tâches expérimentées consistent toutes les deux à chercher le mot $b$ dans la liste $N(a, \Gamma, n)$ des $n$ plus proches voisins de $a$, calculée à partir de la similarité $\Gamma$. Dans nos expériences, la similarité peut être $A_{Sim}$, $L_{Sim}$, $LA_{SimCat}$ ou $LA_{SimInter}$. La similarité $\Gamma$ est utilisée pour la tâche 1 pour chercher le mot hypothèse $(h)$ dans la liste des $n$ plus proches voisins du mot référence $\overline{r}$, et pour la tâche 2 pour chercher le mot référence $(r)$ dans la liste des $n$ plus proches voisins du mot hypothèse $h$.

La métrique d'évaluation correspond à la précision à $n$ de trouver le mot $b$, calculée en variant la taille $n$. La précision à $n$, appelée $S(\Gamma, n)$, est calculée pour toutes les paires de mots dans la liste $L$, en tenant compte du nombre de leurs occurrences dans le corpus d'évaluation. Elle se calcule ainsi :

$$S(\Gamma, n) = \frac{\sum_{i=1}^{m} f(i, \Gamma, n) \times \#(a_i, b_i)}{\sum_{i=1}^{m} \#(a_i, b_i)} \tag{8.3}$$

où $f$ est défini comme suit :

$$f(i, \Gamma, n) = \begin{cases} 1 & si\ b_i \subset N(a_i, \Gamma, n) \\ 0 & sinon \end{cases}$$

où $i$ correspond à la $i^{me}$ paire de mots $(a_i, b_i)$ de $L$, $a_i$ et $b_i$ sont définis en fonction des tâches évaluées :

— tâche 1 : $b_i$ est le mot hypothèse $h$ et $a_i$ est le mot de référence $\overline{r}$,

— tâche 2 : $b_i$ est le mot référence $\overline{r}$ et $a_i$ est le mot d'hypothèse h.

### 8.3.3 Résultats expérimentaux

Nous présentons dans cette section les performances de la mesure de similarité $LA_{SimInter}(\lambda, h, \overline{r})$ sur deux tâches : la prédiction d'erreurs pour les mots rares et l'enrichissement de réseaux de confusion.

#### 8.3.3.1 Prédiction d'erreurs pour les mots rares

Comme nous l'avons vu, nous supposons que les ambiguïtés entre les mots dans un SRAP viennent de leur proximité acoustique, ou linguistique, ou très probablement des deux. Nous proposons d'utiliser les similarités cosinus acoustiques et/ou linguistiques pour dériver les voisins les plus proches d'un mot donné. Ces voisins sont alors considérés comme des erreurs potentielles du SRAP pour un mot donné.

La capacité de pouvoir produire ce type de liste est par exemple utile pour accélérer la correction de transcriptions automatiques en fournissant une liste courte de mots parmi lesquels le mot correct peut être choisi, via une interface de correction [Liang *et al.*, 2014].

Pour les mots fréquents, il est possible de calculer une matrice de confusion en comparant les transcriptions automatiques aux transcriptions manuelles sur les données audio disponibles avec des annotations manuelles. L'analyse de cette matrice de confusion fournit des informations pour prédire les erreurs les plus courantes pour les mots fréquents.

En revanche, il est très difficile de prédire les confusions qui peuvent gêner un SRAP lors du traitement de mots rares. Ces derniers correspondent aux mots non observés précédemment dans les transcriptions automatiques de la parole *i.e.* mots pour lesquels aucune information préalable sur les erreurs n'est disponible. C'est pourquoi nous proposons d'utiliser les mesures de similarité présentées ci-dessus pour prédire les erreurs potentielles pour ces mots rares.

Pour ces expériences, la liste $Sub_{Test}$ a été filtrée pour ne garder que les erreurs (mots de référence $\overline{r}$) non présentes dans Train. Elle est composée de 538 paires d'erreurs de substitution, nommées $Sub_{TestMotsrares}$ dans ce qui suit.

En s'appuyant sur les similarités linguistiques, acoustiques ou combinées, des listes contenant les 30 plus proches voisins, extraites du vocabulaire du SRAP, sont calculées pour chaque mot de référence $\overline{r}$ dans $Sub_{TestMotsrares}$. Les quatre listes de similarité résultantes sont nommées respectivement $List_{SimL}$, $List_{SimA}$, $List_{SimCat}$ et $List_{SimInter}$.

Pour l'évaluation, nous proposons de comparer nos listes de similarité à deux listes naïves qui sont :

— La liste $Mots_{Freq}$ contient les 30 mots les plus fréquents dans Train,

— La liste $Erreurs_{Freq}$ contient les 30 erreurs les erreurs les plus fréquentes de mots hypothèses ($h$) dans Train.

Il est à noter que ces deux listes sont communes pour tous les mots de référence dans $Sub_{TestMotsrares}$, alors que ce n'est pas le cas pour les autres listes.

La figure 8.4 illustre les résultats de prédiction de confusions potentielles pour les mots non observés précédemment dans les transcriptions automatiques, en utilisant les listes décrites ci-dessus et en faisant varier leurs tailles de 1 à 30.

Nous observons que les meilleurs résultats sont obtenus par $List_{SimInter}$ suivi de $List_{SimA}$. Cela montre que notre proposition d'optimisation du coefficient d'interpolation pour combiner les similarités $L_{Sim}$ et $A_{Sim}$ est pertinente. La zone intéressante

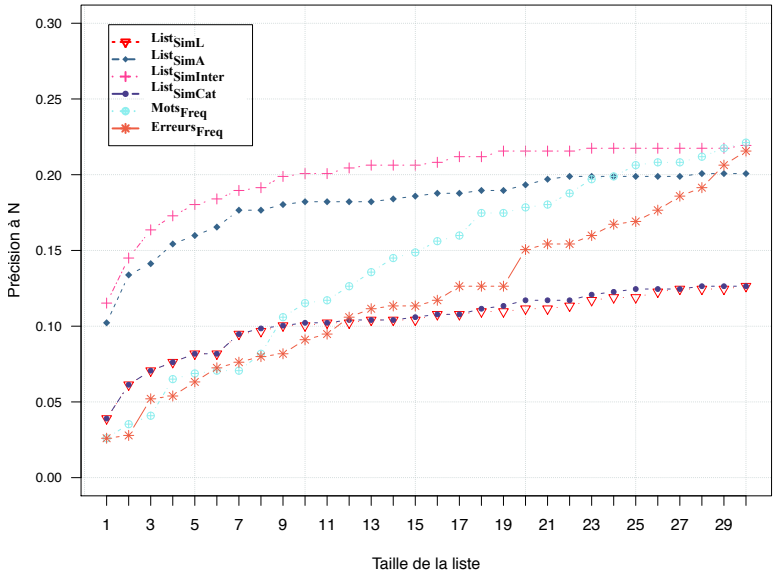



FIGURE 8.4 – Performance de prédiction d'erreurs pour les mots rares, en comparaison aux deux listes : $Mots_{freq}$ et $Erreurs_{Freq}$ et en variant la taille des listes.

dans cette figure est la partie gauche, qui montre les résultats de la prédiction lorsque la liste de confusion est courte. Lorsque cette liste est composée d'un seul mot, la prédiction est correcte dans 11% des cas. Ceci doit être analysé au regard de la taille du vocabulaire du SRAP, qui contient 160k mots : chaque mot du vocabulaire est susceptible d'être sélectionné dans une liste de confusion. La prédiction est correcte dans 20% des cas lorsque la taille de la liste $List_{SimInter}$ est égale à 12. Il semble que cette liste atteigne alors un plateau.

Ces résultats montrent que l'utilisation conjointe des *embeddings* linguistiques et acoustiques est utile pour prédire les erreurs peuvant être générées par un SRAP. Mais cette combinaison ne peut pas être réalisée par une simple concaténation de ces vecteurs : l'approche $LA_{SimCat}$ est loin d'être aussi performante que $LA_{SimInter}$.

En analysant les erreurs correctement prédites à l'aide de ces listes, nous observons dans le tableau 8.10 qu'avec la liste $List_{SimInter}$, les erreurs les plus probables pour un mot donné sont effectivement proches en terme de prononciation et d'orthographe. Cependant, avec $Mots_{Freq}$ les erreurs correctement prédites sont les mots outils et d'autres mots très fréquents.

#### 8.3.3.2 Enrichissement des réseaux de confusion

**Aide à la post-édition pour la correction des erreurs d'un SRAP** De nombreuses études se sont focalisées sur la détection et la correction d'erreurs du SRAP. Certaines d'entre elles [Stoyanchev *et al.*, 2012b, Pincus *et al.*, 2013, Soto *et al.*, 2014] ont proposé d'améliorer la qualité des transcriptions pour de nom-

| Erreurs correctement prédites | |
|---|---|
| $LA_{SimInter}$ | $Freq_{Words}$ |
| altitude → attitude | dos → de |
| baies → baie | mont-blanc → l' |
| déplorable → déplorables | citoyens → d' |
| quentel → quintel | piaffe → il |

TABLE 8.10 – Les paires de mots ($\overline{r}$, h) des confusions correctement prédites.

breuses tâches telles que la recherche de mots clés, la compréhension de la parole et d'autres tâches nécessitant la post-édition des sorties du SRAP.

D'autres études se sont intéressées à l'utilisation des réseaux de confusion (CN) issus du SRAP pour réduire le taux d'erreurs mots et calculer une mesure de confiance. Les réseaux de confusions ont été introduits dans [Mangu *et al.*, 2000]. Ils sont utilisés pour représenter un ensemble de phrases alternatives et s'appuient sur les probabilités *a posteriori*. Les auteurs dans [Fusayasu *et al.*, 2015] proposent une approche pour corriger d'une manière automatique des mots erronés dans les réseaux de confusions. Celle-ci est fondée sur l'utilisation de descripteurs contextuels et de la distance "*Normalized Relevance Distance*" comme une mesure de similarité sémantique entre les mots situés loin les uns des autres.

Les réseaux de confusion peuvent également être utilisés pour améliorer le post-traitement des sorties du SRAP. Par exemple, ils peuvent être utilisés pour proposer d'autres hypothèses quand les transcriptions automatiques sont corrigées par un humain [Laurent *et al.*, 2011]. Cependant, les cohortes ou *bins* d'un réseau de confusion, *i.e.* ensembles des mots concurrents entre deux noeuds d'un CN, n'ont pas une taille fixe et ne contiennent parfois qu'un ou deux mots. Ceci a pour effet de réduire la possibilité d'aider un annotateur humain à corriger un mot mal reconnu puisque le nombre de mots-hypothèses alternatifs est très faible.

Nous proposons d'utiliser à la fois des *embeddings* linguistiques et acoustiques pour enrichir *a posteriori* les réseaux de confusion, afin d'améliorer le post-traitement des sorties du SRAP. Pour cela, nous proposons d'enrichir les réseaux de confusion en ajoutant pour chaque mot reconnu les mots de sa liste de confusion jusqu'à obtenir pour chaque cohorte une même taille fixe.

Pour ces expériences nous avons utilisé la liste des substitutions $Sub_{Test}$ ainsi que les réseaux de confusion (cohortes) correspondants produit par le SRAP LIUM. La figure 8.5 illustre le pourcentage de cohortes dans ces CNs en fonction du nombre de mots alternatifs (*i.e* mots en concurrence avec la meilleure hypothèse (*1-best*)). Les cohortes qui ont une taille entre 6 et 12 sont regroupées en une seule classe [6-12]. Les cohortes ne sont effectivement pas de taille fixe et 55% d'entre elles contiennent aucun ou un seul mot alternatif. Ceci souligne l'intérêt de l'enrichissement des CN.

Pour chaque mot hypothèse ($h$) dans $Sub_{Test}$, ses 6 plus proches voisins dans sa liste de confusion sont extraits en s'appuyant sur la similarité $LA_{SimInter}$. La liste résultante est nommée $List^{h}_{SimInter}$. Ensuite, pour chaque paire de mots ($h$, $\overline{r}$) dans

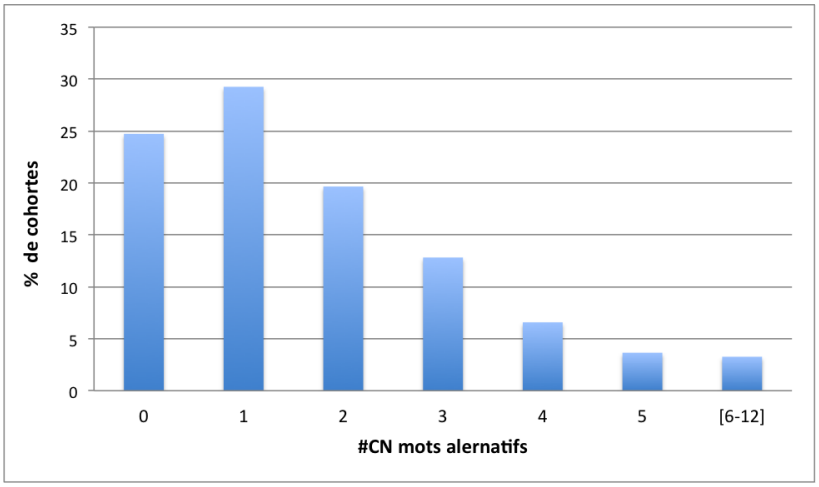



FIGURE 8.5 – Pourcentage des cohortes dans les réseaux de confusion en fonction du nombre de mots en concurrences avec les 1-meilleures hypothèses.

$Sub_{Test}$, nous enrichissons la cohorte associée à $h$ avec ses voisins les plus proches dans la liste $List^{h}_{SimInter}$ jusqu'à obtenir une taille de cohorte égale à 6 (si la cohorte contenait déjà au moins 6 mots alternatifs à $h$, elle n'est pas enrichie). Cette taille semble pertinente pour visualiser des mots alternatifs dans une interface graphique d'un logiciel de transcription assisté par ordinateur [Cardinal *et al.*, 2007]. La liste des mots alternatifs de la cohorte est nommée $List_{CN}$, et celle de la cohorte enrichie est nommée $List_{EnrichCN}$.

Nous avons évalué la performance de ces listes sur la tâche d'aide à la correction manuelle d'erreurs. Cette tâche consiste à proposer le bon mot ($\overline{r}$), lorsqu'il y a une erreur de reconnaissance, parmi les mots de la liste de mots alternatifs du mot reconnu ($h$). Les résultats expérimentaux, présentés dans le tableau 8.11, montrent que notre approche est pertinente : les réseaux de confusion enrichis permettent d'augmenter la précision à 6 ($P$@6) de plus de 23% par rapport au CN produit par notre SRAP. La valeur $P$@6 pour les mots alternatifs proposés par la $LA_{\textbf{SimInter}}$ seule est égal à $0,11$.

| | **List$_{\textbf{CN}}$** | **List$_{EnrichCN}$** |
|---|---|---|
| $P$@6 | 0,17 | **0,21 (+23,5%)** |

TABLE 8.11 – Performance de l'enrichissement de réseaux de confusion : évaluation des listes $List_{CN}$ et $List_{EnrichCN}$ sur la tâche de correction d'erreurs en termes de précision à 6.

**Aide pour la tâche de compréhension de la parole** L'approche d'enrichissement de réseaux de confusion que nous avons proposée peut être utile pour une tâche de compréhension de la parole, afin d'apporter une information supplémentaire pour aider le module d'interprétation sémantique à devenir plus robuste aux

erreurs de reconnaissance de la parole.

En particulier, dans le cas où seule la meilleure hypothèse est fournie au gestionnaire de dialogue, notre approche permet de construire des réseaux de confusion. Ce cas où seule la meilleure hypothèse de reconnaissance est disponible est très courant dans l'industrie, en particulier pour les entreprises qui font appel à des services de reconnaissance de la parole en ligne (par exemple dans le *cloud*) pour traiter le signal de parole et récupérer une transcription automatique.

Pour cette expérience, nous avons utilisé les transcriptions automatiques du corpus MEDIA décrites dans la section 8.2.

Souvent, la tâche SLU dérivée du corpus MEDIA consiste à étiqueter par des concepts sémantiques les mots reconnus par un SRAP [Raymond et Riccardi, 2007]. Pour une telle tâche, un mot mal reconnu implique habituellement une erreur d'étiquetage, dont le nombre peut être réduit en utilisant des réseaux de confusion ou des treillis de mots [Servan *et al.*, 2006], lorsqu'ils sont disponibles. Dans le cas où seule la meilleure hypothèse est disponible, nous proposons de fournir des mots alternatifs en nous appuyant sur la similarité $LA_{SimInter}$.

Comme précédemment mais sur les données MEDIA, les transcriptions automatiques sont alignées avec celles de référence afin d'extraire la liste d'erreurs de substitution produites par le SRAP. Cette liste est divisée en deux ensembles : $Train_{MEDIA}$ pour calculer le coefficient d'interpolation $\hat{\lambda}$, qui est enrichi avec le corpus Train utilisé précédemment (issu des données ETAPE). $Test_{MEDIA}$ est utilisé pour l'évaluation, et a été filtré pour ne conserver que les 1204 occurrences de mots erronés sémantiquement pertinents, c'est-à-dire associés à une étiquette porteuse de sens dans le contexte de l'application informatique des données MEDIA.

La taille du vocabulaire du SRAP dédié à MEDIA est limité à 2,5K mots. Pour le calcul des mots les plus proches, nous y ajoutons également les mots contenus dans le vocabulaire composé de 160K mots utilisé précédemment.

Pour chaque mot hypothèse ($h$) de la liste $Test_{MEDIA}$, nous dérivons leurs 6 voisins les plus proches du vocabulaire, en nous appuyant sur la similarité linguistique et acoustique $LA^{h}_{SimInter}$.

En utilisant la liste résultante, nous pouvons enrichir par des mots alternatifs l'hypothèse produite par le SRAP, puis calculer le nombre de fois où nous proposons le mot correct à reconnaître comme une alternative dans cette liste.

Les résultats de nos expériences montrent que grâce à la similarité $LA^{h}_{SimInter}$ que nous avons proposée, il est possible de construire des listes de 6 mots alternatifs parmi lesquelles se trouvent dans 20,6 % des cas le mot sémantiquement pertinent qui était initialement mal reconnu.

## 8.4 Conclusion

Nous avons présenté dans ce chapitre quelques pistes d'exploitation des sorties de notre système de détection d'erreurs, ainsi que des *embeddings* acoustiques et linguistiques dans certains cadres applicatifs en lien avec le post-traitement de sorties

de SRAP.

D'abord, nous nous sommes intéressés à l'évaluation des mesures de confiance dérivées des sorties de nos systèmes de détection d'erreurs en termes de NCE : les résultats de cette évaluation montrent l'apport très significatif de ces mesures de confiance.

Ensuite, nous avons montré que l'intégration de ces mesures de confiance comme information additionnelle dans un système de compréhension de la parole améliore ses performances en termes de taux d'erreurs de concept et taux d'erreurs de couples concept/valeur.

Enfin, nous avons exploité les *embeddings* linguistiques et acoustiques pour estimer une mesure de similarité permettant de prédire des listes de confusion qui énumèrent les mots qui peuvent être confondus par un SRAP pour un mot donné. Ces listes représentent les voisins les plus proches linguistiquement et acoustiquement de ce mot. Cette proximité est fournie par une mesure de similarité calculée à partir de l'interpolation linéaire des similarités linguistique et acoustique. Les performances de cette métrique sont évaluées sur la tâche de prédiction d'erreurs potentielles pour les mots rares et sur la tâche d'enrichissement d'hypothèses de SRAP pour l'aide au post-traitement. Les résultats expérimentaux montrent que cette mesure est efficace pour prédire les erreurs générées par un SRAP pour les mots rares, en comparaison à d'autres solutions s'appuyant uniquement sur des *embeddings* linguistiques, acoustiques ou sur des statistiques générales.

Les résultats montrent également les performances de cette mesure de similarité pour la tâche d'aide à la correction d'erreurs dans deux scénarios différents. Le premier consiste à enrichir des réseaux de confusion avec les liste de confusion. Ceci permet d'augmenter la précision à 6 pour la recherche du mot correct pour un mot erroné de 23% par rapport aux résultats obtenus par les réseaux de confusion produits par notre SRAP. Le deuxième s'intéresse à l'aide à la robustesse aux erreurs de reconnaissance de la parole pour une tâche d'extraction d'information sémantique en fournissant des alternatives aux mots présents dans la $1-best$. Dans une tâche d'extraction de concepts sémantiques de type MEDIA, l'utilisation de ces listes de confusion permet de fournir le mot correct parmi 6 alternatives pour 20,6% des mots sémantiquement pertinents qui étaient initialement mal reconnus.

# CHAPITRE 9

# CONCLUSION ET PERSPECTIVES

## 9.1 Conclusion

Le travail de thèse présenté dans ce manuscrit met en avant notre étude sur les *word embeddings* appliqués à la tâche de détection d'erreurs dans des transcriptions automatiques de la parole. L'utilisation de ces représentations de mots était initialement justifiée par leurs performances prometteuses pour différentes tâches du traitement des langues naturelles et du traitement de la parole.

La génération de transcriptions automatiques de la parole s'appuie à la fois sur des informations linguistiques capturées par des modèles de langage et sur des informations acoustiques capturées par des modèles acoustiques. La détection d'erreurs consiste alors à détecter les incongruités linguistiques et/ou acoustiques dans les transcriptions automatiques. L'objectif est donc de trouver la représentation la plus appropriée du mot qui permet de capturer des informations pertinentes pour pouvoir détecter ces anomalies.

Ce travail de thèse porte sur plusieurs axes. Dans un premier temps, une étude préliminaire sur l'utilisation de différents types de *word embeddings* linguistiques a été effectuée sur la tâche de détection d'erreurs. L'évaluation de l'impact de ces différents *word embeddings* sur cette tâche a montré une complémentarité potentielle. C'est pour cette raison que nous avons par la suite effectué une comparaison systématique de leur utilisation sur plusieurs tâches connues en traitement automatique du langage naturel. Nous avons également étudié et évalué différentes approches pour combiner ces *word embeddings* afin de tirer profit de leurs complémentarités pour la tâche de détection d'erreurs. Nous avons trouvé que le *word embedding* combiné avec un auto-encodeur obtient les meilleurs résultats sur cette tâche : l'information portée par ce *word embedding* linguistique combiné permet d'améliorer les performances de notre système de détection d'erreurs. Les résultats obtenus montrent une amélioration par rapport à l'état de l'art, qui s'appuie sur les champs aléatoires conditionnels (CRF).

Dans un second temps, nous nous sommes intéressés à enrichir le système de détection d'erreurs par des informations extraites du signal de parole, notamment des *embeddings* acoustiques et des informations prosodiques. Nous avons implémenté une approche pour construire des *embeddings* acoustiques qui repose sur l'utilisation d'un réseau de neurones convolutif qui calcule des *embeddings* acoustiques de signal pour chaque occurrence d'un mot dans un corpus d'apprentissage, et d'un réseau de neurones profond pour généraliser le calcul d'un *embedding* acoustique pour n'importe quel mot à partir de son orthographe. Ce dernier réseau permet ainsi de construire aussi des *embeddings* acoustiques pour les mots non observés dans le corpus d'apprentissage audio. La performance de ces *embeddings* acoustiques de mot a été évaluée en utilisant deux approches : les tâches de similarités orthographique et phonétique et la tâche de détection d'homophones. L'ajout de l'information acoustique ou celui de l'information prosodique permet d'obtenir des performances équivalentes en termes de CER, et améliore les résultats par rapport à l'utilisation uniquement d'informations linguistique et syntaxique (ce système est nommé MLP-MS-AutoE). Nous avons constaté que ces informations acoustiques et prosodiques sont complémentaires. En effet, nous avons obtenu avec cette combinaison les meilleurs résultats, et ce de manière significative par rapport aux systèmes MLP-MS-AutoE et CRF. La combinaison des informations syntaxiques, linguistiques et acoustiques (ce système est nommé DMLP-MS-AC) permet d'obtenir des améliorations significatives de $9,63\%$ et $9,46\%$ en termes de réduction de CER par rapport au CRF respectivement sur les corpus Dev et Test.

Ensuite, nous avons proposé une étude sur la modélisation de l'erreur au niveau de la phrase pour compenser certains phénomènes mis en avant par une analyse qualitative des sorties du système proposé. Afin de compenser les limites de notre approche de détection d'erreurs, qui ne traite chaque mot que dans un contexte local, nous avons exploré deux approches. La première consiste à intégrer des connaissances sur la phrase, en utilisant des *embeddings* de phrases. Nous avons montré que l'intégration des *embeddings* de phrases (ce système est nommé DMLP-MS-AC-Emb$_{CNN}$), apporte une amélioration par rapport aux résultats de DMLP-MS-AC. Ces *embeddings* sont extraits d'un réseau de neurones convolutif appris pour la classification de phrases transcrites automatiquement en *peu erronées* ou *très erronées*, selon la proportion d'erreurs que contient chaque phrase reconnue. La deuxième approche repose sur l'utilisation d'un modèle contextuel probabiliste de la distribution d'erreurs. Celle-ci est appliquée à deux systèmes : DMLP-MS-AC et DMLP-MS-AC-Emb$_{CNN}$. Cette approche permet également une légère amélioration des taux d'erreur de classification, surtout lorsqu'elle est appliquée aux sorties du système DMLP-MS-AC.

Nous résumons dans le tableau 9.1 les résultats obtenus par le système CRF sans et avec descripteurs prosodiques et par les systèmes neuronaux proposés qui sont :

— MLP-MS w2vf-deps : le système qui intègre l'*embedding* linguistique non combiné w2vf-deps en plus des descripteurs syntaxiques.

— MLP-MS-AutoE : le système qui intègre l'*embedding* linguistique combiné avec

un auto-encodeur à la place de w2vf-deps.

— DMLP-MS-AC : le système MLP-MS-AutoE enrichi par des *embeddings* acoustiques et des descripteurs prosodiques.

— DMLP-MS-AC-FSM : l'application du modèle contextuel probabiliste au système DMLP-MS-AC

— DMLP-MS-AC-Emb$_{CNN}$ : le système DMLP-MS-AC enrichi par des *embeddings* sur la phrase

— DMLP-MS-AC-Emb$_{CNN}$-FSM : l'application du modèle contextuel probabiliste au système DMLP-MS-AC-Emb$_{CNN}$

Ce tableau présente aussi les gains relatifs en termes de réduction de CER obtenus par les systèmes neuronaux en comparaison avec le système CRF initial.

| **Approches** | **Label erreur** | | | **Globale** | **Gain** % |
|---|---|---|---|---|---|
| | **P** | **R** | **F** | **CER** | |
| CRF | 0,676 | **0,547** | **0,605** | 8,56 | - |
| CRF $\oplus$ pros | 0,686 | 0,541 | 0,605 | 8,46 | 0,86 |
| MLP-MS w2vf-deps | 0,719 | 0,509 | 0,596 | 8,26 | 3,5 |
| MLP-MS-AutoE | 0,696 | 0,578 | 0,632 | 8,07 | 5,72 |
| DMLP-MS-AC | 0,702 | **0,613** | **0,654** | 7,75 | 9,46 |
| DMLP-MS-AC-Emb$_{CNN}$ | 0,724 | 0,578 | 0,643 | 7,69 | 10,16 |
| DMLP-MS-AC-FSM | 0,718 | 0,591 | 0,649 | **7,67** | **10,39** |
| DMLP-MS-AC-Emb$_{CNN}$-FSM | **0,726** | 0,573 | 0,641 | 7,69 | 10,16 |

TABLE 9.1 – Performances des systèmes de détection d'erreurs neuronaux en comparaison avec l'approche CRF sur le corpus Test.

On observe que les informations linguistiques, acoustiques et globales sur la phrase utilisées pour enrichir le système de détection d'erreurs permettent d'améliorer progressivement les gains. La combinaison de toutes ces connaissances apporte un gain significatif de 10,39% par rapport au système initial construit avec des CRF.

Enfin, nous nous sommes intéressés à l'exploitation des mesures de confiance issues de nos systèmes de détection d'erreurs et à l'utilisation des *word embeddings* linguistiques et acoustiques de mots dans différents cadres applicatifs. Nous avons montré que l'intégration de ces mesures de confiance comme information additionnelle dans un système de compréhension de la parole améliore ses performances en termes de taux d'erreurs de concept et taux d'erreurs de couples concept-valeur. De plus, nous avons exploité les *word embeddings* linguistiques et acoustiques de mots pour estimer une mesure de similarité permettant de prédire des listes de confusion. Ces listes représentent les voisins les plus proches linguistiquement et acoustiquement pour un mot donné. Les performances de cette mesure de similarité ont été évaluées dans un cadre de prédictions des erreurs possibles pour des mots rares du vocabulaire, et dans un contexte d'aide à la post-édition manuelle des transcriptions automatiques. Nous avons montré que la mesure de similarité permet de prédire

des erreurs pouvant être générées par un SRAP sur des mots rares. L'utilisation de cette mesure de similarité pour enrichir les réseaux de confusion (CN) permet d'augmenter la précision à 6 de plus de 23% par rapport au CN produit par notre SRAP. Dans une situation où seule l'hypothèse $1-best$ est disponible, cette mesure permet également de proposer dans 20,6% des cas le mot correct parmi 6 propositions alternatives à un mot erroné porteur de sens pour une application de dialogue humain-machine.

## 9.2 Perspectives

À partir des contributions exposées dans ce manuscrit, plusieurs perspectives de recherche peuvent être envisagées.

Tout d'abord, nous proposons d'approfondir l'étude que nous avons faite sur l'évaluation des *word embeddings* linguistiques. Une première limite que nous avons rencontrée dans cette partie est le manque de jeux d'évaluation en français pour les tâches de similarité et d'analogie, ce qui nous a amené à travailler sur des données de langue anglaise.

Nous proposons donc de construire des jeux d'évaluation en français pour ces deux tâches, en tenant compte des problèmes de corpus abordés dans la littérature liés à l'évaluation de ces deux tâches.

Par exemple, afin d'affiner l'évaluation, voire de ne pas la biaiser, il serait important de ne pas mélanger dans la même tâche d'évaluation de la similarité la similarité fonctionnelle, définie en tenant compte des relations lexicales de synonymie ("voiture : automobile") et d'hyponymie ("voiture : véhicule"), et la similarité associative, qui couvre tout type d'association lexicale ou fonctionnelle (partie de relation, comme dans "doigt : main"), antonymie (sens opposés, comme "chaud : froid"). C'est un problème qui existe dans différents jeux d'évaluation comme celui des données MEN.

Autre exemple qui concerne la tâche d'analogie : la fonction utilisée pour l'évaluation s'appuie sur la translation vectorielle et n'est pas toujours fiable. En effet, pour certaines tâches d'analogie, une simple similarité cosinus suffirait pour trouver le mot en question [Linzen, 2016].

Nous proposons également d'utiliser les listes des mots proches acoustiquement et linguistiquement estimées en fonction des *embeddings* linguistiques et acoustiques dans différentes applications.

— Corriger automatiquement des mots reconnus qui auraient été détectés comme étant erronés, en s'appuyant sur les réseaux de confusion enrichis que nous avons vu dans la section 8.3.3.2. En nous appuyant sur les informations linguistiques, acoustiques et contextuelles, nous pourrions utiliser un système de classification automatique pour déterminer le bon mot parmi les mots alternatifs d'une cohorte.

— Nous pouvons également utiliser ces listes dans les applications de recherche documentaire multimedia qui nécessitent des transcriptions automatiques.

Ainsi, dans le cas où un mot clé n'existait pas dans le vocabulaire de l'ASR au moment de la transcription automatique d'un document audiovisuel, nous pourrions construire une stratégie de recherche s'appuyant sur d'autres mots, inclus dans le vocabulaire et proches acoustiquement ou linguistiquement du mot clé recherché.

À partir de notre analyse qualitative des détections d'erreurs réalisées par nos premiers systèmes, nous avons constaté que notre approche ne permet pas de résoudre certains problèmes, notamment ceux liés à la détection d'erreurs isolées, ou encore à la distribution d'erreurs, mesurée à travers la notion d'empan. Les pistes d'amélioration que nous avons explorées à la suite de cette analyse ont eu un impact positif, mais limité. Cela peut s'expliquer par le fait que l'information fournie en entrée de notre système de détection d'erreurs n'est pas suffisante, ou bien que notre système n'est pas capable d'extraire l'information nécessaire présente dans le contexte. L'architecture que nous avons proposée utilise une fenêtre de contexte de taille fixe pour tous les mots : ces informations sont parfois peut-être insuffisantes ou inutiles pour certains mots. C'est pourquoi nous proposons d'exploiter d'autres architectures neuronales pour améliorer la prédiction, par exemple en utilisant un système encodeur/décodeur dont le mécanisme d'attention serait dédié à la tâche de détection d'erreurs.

Une autre perspective intéressante serait l'intégration, dans le système de détection d'erreurs, de la prise en compte de sous-parties de l'espace de recherche construit au cours de la transcription automatique, sous la forme de réseaux de confusion (CN) ou de listes $n-best$. Ces informations, qui portent sur l'environnement concurrentiel d'un mot-hypothèse, apporteraient probablement des connaissances utiles et complémentaire pour la détection d'erreurs.

Nous proposons aussi de nous concentrer sur l'analyse des sources d'incertitude : mot hors vocabulaire, modèle acoustique ou modèle de langage incertain, phonétisation incorrecte, événement acoustique perturbateur, *etc.* L'élaboration d'un système de caractérisation d'erreurs nécessite l'utilisation d'un corpus annoté en type d'erreurs. L'annotation de ce corpus peut se faire soit d'une manière manuelle par un expert humain ou d'une manière automatique. Pour réaliser l'annotation automatique, nous utiliserions plusieurs sources d'informations qui caractérisent les mots, y compris des informations extraites au niveau du système de reconnaissance de la parole comme les scores du modèle de langage, du modèle acoustique, *etc.*

Enfin, nous envisageons d'injecter les informations linguistiques, acoustiques, et contextuelles que nous avons utilisées pour la détection d'erreurs dans un système de reconnaissance de la parole multi-passes entièrement neuronal, en vue d'améliorer la qualité des transcriptions automatiques en participant à l'essor de la nouvelle génération de système de reconnaissance automatique de la parole.

# BIBLIOGRAPHIE PERSONNELLE

### Conférences d'audience internationale

— Edwin Simonnet, Sahar Ghannay, Yannick Estève, Nathalie Camelin, Renato De Mori, *ASR error management for improving spoken language understanding.* Interspeech 2017, Stockholm, Suède, 20-24 Aout.

— Sahar Ghannay, Yannick Estève, Nathalie Camelin, Paul Deléglise, *Acoustic word embeddings for ASR error detection.* Interspeech 2016, San Francisco (CA, USA), 9-12 Septembre.

— Sahar Ghannay, Benoit Favre, Yannick Estève, Nathalie Camelin, *Word embedding evaluation and combination*, 10th edition of the Language Resources and Evaluation Conference (LREC 2016), Portorož (Slovénie), 23-28 Mai.

— Sahar Ghannay, Yannick Estève, Nathalie Camelin, *Word embeddings combination and neural networks for robustness in ASR error detection,* European Signal Processing Conference (EUSIPCO 2015), Nice (France), 31 Aout-4 Septembre.

— Sahar Ghannay, Yannick Estève, Nathalie Camelin, Camille Dutrey, Fabian Santiago, Martine Adda-Decker, *Combining continous word representation and prosodic features for ASR error prediction*, 3rd International Conference on Statistical Language and Speech Processing (SLSP 2015), Budapest (Hongrie), 24-26 Novembre.

### Workshops d'audience internationale

— Sahar Ghannay, Yannick Estève, Nathalie Camelin, Paul Deléglise, *Evaluation of acoustic word embeddings*, RepEval@ACL 2016 : The 1st ACL Workshop on Evaluating Vector-Space Representations for NLP , Berlin (Allemagne), 12 Aout.

— Yannick Estève, Sahar Ghannay, Nathalie Camelin, *Recent improvements on error detection for automatic speech recognition*, 1st International Workshop on Multimodal Media Data Analytics (MMDA 2016), in Conjunction with the

22nd European Conference on Artificial Intelligence, Pays-Bas (La Haye), 30 Aout.

— Sahar Ghannay, Nathalie Camelin, Yannick Estève, *Which ASR errors are hard to detect ?*, Workshop Errors by Humans and Machines in multimedia, multimodal and multilingual data processing (ERRARE 2015) , Sinaia (Romani), 11-13 Septembre.

— Sahar Ghannay and Loïc Barrault. *Using Hypothesis Selection Based Features for Confusion Network MT System Combination*, Proceedings of the 3rd EACL Workshop on Hybrid Approaches to Translation (HyTra), Göteborg, Suède, 27 Avril.

### Conférences d'audience nationale

— Sahar Ghannay, Yannick Estève, Nathalie Camelin, Camille Dutrey, Fabian Santiago, Martine Adda-Decker, *Utilisation des représentations continues des mots et des paramètres prosodiques pour la détection d'erreurs dans les transcriptions automatiques de la parole*, 5ème èdition de JEP-TALN-RECITAL, Paris (France), 4-8 juillet.

# BIBLIOGRAPHIE


[Abdel-Hamid *et al.*, 2012] Abdel-Hamid, O., Mohamed, A.-r., Jiang, H. et Penn, G. (2012). Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition. *In Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 4277–4280. IEEE.

[Adda-Decker *et al.*, 2008] Adda-Decker, M., Gendrot, C. et Nguyen, N. (2008). Contributions du traitement automatique de la parole à l'étude des voyelles orales du français. *Traitement Automatique des Langues*, 49(3):13–46.

[Amodei *et al.*, 2015] Amodei, D., Anubhai, R., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Chen, J., Chrzanowski, M., Coates, A., Diamos, G. *et al.* (2015). Deep speech 2 : End-to-end speech recognition in english and mandarin. *arXiv preprint arXiv :1512.02595*.

[Anguera *et al.*, 2014] Anguera, X., Rodriguez-Fuentes, L. J., Szöke, I., Buzo, A. et Metze, F. (2014). Query by example search on speech at mediaeval 2014. *In MediaEval*.

[Arora *et al.*, 2016] Arora, S., Li, Y., Liang, Y., Ma, T. et Risteski, A. (2016). A latent variable model approach to pmi-based word embeddings. *Transactions of the Association for Computational Linguistics*, 4:385–399.

[Asadi *et al.*, 1990] Asadi, A., Schwartz, R. et Makhoul, J. (1990). Automatic detection of new words in a large vocabulary continuous speech recognition system. *In Acoustics, Speech, and Signal Processing, 1990. ICASSP-90., 1990 International Conference on*, pages 125–128. IEEE.

[Bahdanau *et al.*, 2014] Bahdanau, D., Cho, K. et Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv :1409.0473*.

[Bansal *et al.*, 2014a] Bansal, M., Gimpel, K. et Livescu, K. (2014a). Tailoring Continuous Word Representations for Dependency Parsing. *In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2 : Short Papers)*, pages 809–815. Association for Computational Linguistics.

[Bansal *et al.*, 2014b] Bansal, M., Gimpel, K. et Livescu, K. (2014b). Tailoring continuous word representations for dependency parsing. *In ACL (2)*, pages 809–815.

[Bechet, 2001] Bechet, F. (2001). LIA-PHON : UN système complet de phonétisation de textes. *TAL. Traitement automatique des langues*, 42(1):47–67.

[Béchet et Favre, 2013] Béchet, F. et Favre, B. (2013). Asr error segment localization for spoken recovery strategy. *In Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6837–6841.

[Bengio et Heigold, 2014] Bengio, S. et Heigold, G. (2014). Word embeddings for speech recognition. *In Proceedings of the 15th Conference of the International Speech Communication Association, Interspeech*.

[Bengio, 2009a] Bengio, Y. (2009a). Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127.

[Bengio, 2009b] Bengio, Y. (2009b). Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127. Also published as a book. Now Publishers, 2009.

[Bengio, 2012] Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. *In Neural Networks : Tricks of the Trade*, pages 437–478. Springer.

[Bengio *et al.*, 2003] Bengio, Y., Ducharme, R., Vincent, P. et Janvin, C. (2003). A neural probabilistic language model. volume 3, pages 1137–1155. JMLR.org.

[Bengio *et al.*, 2007a] Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H. *et al.* (2007a). Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153.

[Bengio *et al.*, 2007b] Bengio, Y., LeCun, Y. *et al.* (2007b). Scaling learning algorithms towards ai. *Large-scale kernel machines*, 34(5).

[Bengio *et al.*, 2005] Bengio, Y., Roux, N. L., Vincent, P., Delalleau, O. et Marcotte, P. (2005). Convex neural networks. *In Advances in neural information processing systems*, pages 123–130.

[Bengio *et al.*, 1994] Bengio, Y., Simard, P. et Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

[Bergstra et Bengio, 2012] Bergstra, J. et Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb): 281–305.

[Bisani et Ney, 2008] Bisani, M. et Ney, H. (2008). Joint-sequence models for grapheme-to-phoneme conversion. *Speech Communication*, 50(5):434–451.

[Bishop, 2006] Bishop, C. M. (2006). Pattern recognition. *Machine Learning*, 128.

[Blei *et al.*, 2003] Blei, D. M., Ng, A. Y. et Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

[Boersma et Weenink, 2001] Boersma, P. et Weenink, D. (2001). Praat, a system for doing phonetics by computer. *Glot International*, 5(9):341–345.

[Bonneau-Maynard *et al.*, 2005] Bonneau-Maynard, H., Rosset, S., Ayache, C., Kuhn, A. et Mostefa, D. (2005). Semantic annotation of the french media dialog corpus. *In Ninth European Conference on Speech Communication and Technology*.

[Bottou, 2010] Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. *In Proceedings of COMPSTAT'2010*, pages 177–186. Springer.

[Bougares, 2012] Bougares, F. (2012). *Attelage de systèmes de transcription automatique de la parole*. Thèse de doctorat, Université du Maine.

[Bougares *et al.*, 2013] Bougares, F., Deléglise, P., Esteve, Y. et Rouvier, M. (2013). LIUM ASR system for Etape French evaluation campaign : experiments on system combination using open-source recognizers. *In International Conference on Text, Speech and Dialogue*, pages 319–326. Springer.

[Brown *et al.*, 1992] Brown, P. F., Desouza, P. V., Mercer, R. L., Pietra, V. J. D. et Lai, J. C. (1992). Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

[Bruni *et al.*, 2014] Bruni, E., Tran, N.-K. et Baroni, M. (2014). Multimodal distributional semantics. *J. Artif. Intell. Res.(JAIR)*, 49(1-47).

[Bullinaria et Levy, 2007] Bullinaria, J. A. et Levy, J. P. (2007). Extracting semantic representations from word co-occurrence statistics : A computational study. *Behavior research methods*, 39(3):510–526.

[Cardinal *et al.*, 2007] Cardinal, P., Boulianne, G., Comeau, M. et Boisvert, M. (2007). Real-time correction of closed-captions. *In Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 113–116. Association for Computational Linguistics.

[Carlin *et al.*, 2011] Carlin, M. A., Thomas, S., Jansen, A. et Hermansky, H. (2011). Rapid Evaluation of Speech Representations for Spoken Term Discovery. *In INTERSPEECH*, pages 821–824.

[Charlet *et al.*, 2001] Charlet, D., Mercier, G. et Jouvet, D. (2001). On combining confidence measures for improved rejection of incorrect data. *In INTERSPEECH*, pages 2113–2116.

[Chase, 1997] Chase, L. L. (1997). *Error-responsive feedback mechanisms for speech recognizers*. Carnegie Mellon University Pittsburgh, PA.

[Chen et Gopalakrishnan, 1998] Chen, S. et Gopalakrishnan, P. (1998). Speaker, environment and channel change detection and clustering via the bayesian information criterion. *In Proc. DARPA Broadcast News Transcription and Understanding Workshop*, volume 8, pages 127–132. Virginia, USA.

[Chen et Goodman, 1996] Chen, S. F. et Goodman, J. (1996). An empirical study of smoothing techniques for language modeling. *In Proceedings of the 34th annual*

*meeting on Association for Computational Linguistics*, pages 310–318. Association for Computational Linguistics.

[Cho *et al.*, 2014a] Cho, K., Van Merriënboer, B., Bahdanau, D. et Bengio, Y. (2014a). On the properties of neural machine translation : Encoder-decoder approaches. *arXiv preprint arXiv :1409.1259*.

[Cho *et al.*, 2014b] Cho, K., van Merrienboer, B., Gulcehre, C., Bougares, F., Schwenk, H. et Bengio, Y. (2014b). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *In EMNLP*.

[Church et Hanks, 1990] Church, K. W. et Hanks, P. (1990). Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1): 22–29.

[Collobert et Bengio, 2004] Collobert, R. et Bengio, S. (2004). Links between perceptrons, mlps and svms. *In Proceedings of the twenty-first international conference on Machine learning*, page 23. ACM.

[Collobert et Weston, 2008] Collobert, R. et Weston, J. (2008). A unified architecture for natural language processing : Deep neural networks with multitask learning. *In Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

[Collobert *et al.*, 2011] Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K. et Kuksa, P. (2011). Natural Language Processing (Almost) from Scratch. volume 12, pages 2493–2537. JMLR.org.

[Cornuéjols et Miclet, 2011] Cornuéjols, A. et Miclet, L. (2011). *Apprentissage artificiel : concepts et algorithmes*. Editions Eyrolles.

[Cox et Dasmahapatra, 2002] Cox, S. et Dasmahapatra, S. (2002). High-level approaches to confidence estimation in speech recognition. *IEEE Transactions on Speech and Audio processing*, 10(7):460–471.

[Davis et Mermelstein, 1980] Davis, S. B. et Mermelstein, P. (1980). Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 28(4):357–366.

[Deléglise *et al.*, 2009] Deléglise, P., Estève, Y., Meignier, S. et Merlin, T. (2009). Improvements to the LIUM French ASR system based on CMU Sphinx : what helps to significantly reduce the word error rate ? *In Interspeech*, Brighton, UK.

[Digalakis et Neumeyer, 1995] Digalakis, V. et Neumeyer, L. (1995). Speaker adaptation using combined transformation and bayesian methods. *In Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 680–683. IEEE.

[Do et Artières, 2006] Do, T. M. T. et Artières, T. (2006). Champs de markov conditionnels pour le traitement de séquences. *In EGC*, pages 639–650.

[Dufour *et al.*, 2012] Dufour, R., Damnati, G. et Charlet, D. (2012). Automatic error region detection and characterization in LVCSR transcriptions of TV news shows. *In Acoustics, Speech and Signal Processing (ICASSP)*.

[Dufour *et al.*, 2014] Dufour, R., Estève, Y. et Deléglise, P. (2014). Characterizing and detecting spontaneous speech : Application to speaker role recognition. *Speech Communication*, 56:1–18.

[Dumais, 2004] Dumais, S. T. (2004). Latent semantic analysis. *Annual review of information science and technology*, 38(1):188–230.

[Elman, 1990] Elman, J. L. (1990). Finding structure in time. *Cognitive science*, 14(2):179–211.

[Erhan *et al.*, 2010] Erhan, D., Bengio, Y., Courville, A., Manzagol, P.-A., Vincent, P. et Bengio, S. (2010). Why does unsupervised pre-training help deep learning ? *Journal of Machine Learning Research*, 11(Feb):625–660.

[Erhan *et al.*, 2009] Erhan, D., Manzagol, P.-A., Bengio, Y., Bengio, S. et Vincent, P. (2009). The difficulty of training deep architectures and the effect of unsupervised pre-training. *In AISTATS*, volume 5, pages 153–160.

[Estève, 2002] Estève, Y. (2002). *Intégration de sources de connaissances pour la modélisation stochastique du langage appliquée à la parole continue dans un contexte de dialogue oral homme-machine*. Thèse de doctorat, Université d'Avignon et des Pays de Vaucluse.

[Estève, 2009] Estève, Y. (2009). Traitement automatique de la parole : contributions. *In Habilitation Diriger des Recherches (HDR), LIUM, Universite du Maine, France*.

[Estève *et al.*, 2010] Estève, Y., Bazillon, T., Antoine, J.-Y., Béchet, F. et Farinas, J. (2010). The EPAC Corpus : Manual and Automatic Annotations of Conversational Speech in French Broadcast News. *In LREC, Malta, 17-23 may 2010*.

[Estève *et al.*, 2015] Estève, Y., Bouallegue, M., Lailler, C., Morchid, M., Dufour, R., Linarès, G., Matrouf, D. et Mori, R. D. (2015). Integration of word and semantic features for theme identification in telephone conversations. *In 6th International Workshop on Spoken Dialog Systems (IWSDS 2015)*.

[Falavigna *et al.*, 2002] Falavigna, D., Gretter, R. et Riccardi, G. (2002). Acoustic and word lattice based algorithms for confidence scores. *In INTERSPEECH*.

[Faruqui et Dyer, 2014] Faruqui, M. et Dyer, C. (2014). Community evaluation and exchange of word vectors at wordvectors.org. *In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics : System Demonstrations*, Baltimore, USA. Association for Computational Linguistics.

[Finkelstein *et al.*, 2001] Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G. et Ruppin, E. (2001). Placing search in context : The concept revisited. *In Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM.

[Firat *et al.*, 2016] Firat, O., Cho, K. et Bengio, Y. (2016). Multi-way, multilingual neural machine translation with a shared attention mechanism. *arXiv preprint arXiv :1601.01073*.

[Fiscus et Ajot, 2009] Fiscus, J. et Ajot, J. (May 28-29 2009). The Rich Transcription 2009 Speech-To-Text (STT) and Speaker Attributed STT (SASTT) Results.

[Fiscus *et al.*, 2007] Fiscus, J. G., Ajot, J., Garofolo, J. S. et Doddington, G. (2007). Results of the 2006 spoken term detection evaluation. *In Proc. SIGIR*, volume 7, pages 51–57. Citeseer.

[Freund *et al.*, 1996] Freund, Y., Schapire, R. E. *et al.* (1996). Experiments with a new boosting algorithm. *In ICML*, volume 96, pages 148–156.

[Fukushima, 1980] Fukushima, K. (1980). Neocognitron : A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202.

[Fusayasu *et al.*, 2015] Fusayasu, Y., Tanaka, K., Takiguchi, T. et Ariki, Y. (2015). Word-error correction of continuous speech recognition based on normalized relevance distance. *In IJCAI*, pages 1257–1262.

[Gales, 2000] Gales, M. J. (2000). Cluster adaptive training of hidden markov models. *IEEE transactions on speech and audio processing*, 8(4):417–428.

[Galibert et Kahn, 2013] Galibert, O. et Kahn, J. (2013). The first official repere evaluation. *In SLAM@ INTERSPEECH*, pages 43–48.

[Galliano *et al.*, 2006] Galliano, S., Geoffrois, E., Gravier, G., f. Bonastre, J., Mostefa, D. et Choukri, K. (2006). Corpus description of the Ester evaluation campaign for the rich transcription of French broadcast news. *In 5th international Conference on Language Resources and Evaluation (LREC)*, pages 315–320.

[Galliano *et al.*, 2005] Galliano, S., Geoffrois, E., Mostefa, D., Choukri, K., Bonastre, J.-F. et Gravier, G. (2005). The ESTER phase II evaluation campaign for the rich transcription of French Broadcast News. *In Interspeech*, pages 1149–1152.

[Galliano *et al.*, 2009] Galliano, S., Gravier, G. et Chaubard, L. (2009). The ESTER 2 evaluation campaign for the rich transcription of French radio broadcasts. *In Interspeech*, volume 9, pages 2583–2586.

[Gao *et al.*, 2014] Gao, B., Bian, J. et Liu, T. (2014). WordRep : A Benchmark for Research on Learning Word Representations. *CoRR*, abs/1407.1640.

[Gauthier, 2001] Gauthier, T. D. (2001). Detecting trends using spearman's rank correlation coefficient. *Environmental forensics*, 2(4):359–362.

[Gauvain *et al.*, 2005] Gauvain, J.-L., Adda, G., Lamel, L., Lefèvre, F. et Schwenk, H. (2005). Transcription de la parole conversationnelle. *Traitement Automatique des Langues*, 45(3):35–47.

[Gauvain et Lee, 1994] Gauvain, J.-L. et Lee, C.-H. (1994). Maximum a posteriori estimation for multivariate gaussian mixture observations of markov chains. *IEEE transactions on speech and audio processing*, 2(2):291–298.

[Goldwater *et al.*, 2010] Goldwater, S., Jurafsky, D. et Manning, C. D. (2010). Which words are hard to recognize ? prosodic, lexical, and disfluency factors that increase speech recognition error rates. *Speech Communication*, pages 181–200.

[Gracia et Mena, 2008] Gracia, J. et Mena, E. (2008). Web-based measure of semantic relatedness. *In Web Information Systems Engineering-WISE 2008*, pages 136–150. Springer.

[Graves et Jaitly, 2014] Graves, A. et Jaitly, N. (2014). Towards end-to-end speech recognition with recurrent neural networks. *In ICML*, volume 14, pages 1764–1772.

[Graves *et al.*, 2013] Graves, A., Mohamed, A.-r. et Hinton, G. (2013). Speech recognition with deep recurrent neural networks. *In 2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE.

[Gravier *et al.*, 2012] Gravier, G., Adda, G., Paulsson, N., Carré, M., Giraudel, A. et Galibert, O. (2012). The ETAPE corpus for the evaluation of speech-based TV content processing in the French language. *In Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*.

[Guo *et al.*, 2004] Guo, G., Huang, C., Jiang, H. et Wang, R.-H. (2004). A comparative study on various confidence measures in large vocabulary speech recognition. *In Chinese Spoken Language Processing, 2004 International Symposium on*, pages 9–12. IEEE.

[Gupta *et al.*, 2014] Gupta, V., Kenny, P., Ouellet, P. et Stafylakis, T. (2014). I-vector-based speaker adaptation of deep neural networks for french broadcast audio transcription. *In 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6334–6338.

[Gutmann et Hyvärinen, 2012] Gutmann, M. U. et Hyvärinen, A. (2012). Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13(Feb):307–361.

[Halawi *et al.*, 2012] Halawi, G., Dror, G., Gabrilovich, E. et Koren, Y. (2012). Large-scale learning of word relatedness with constraints. *In Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1406–1414. ACM.

[Hammersley et Clifford, 1971] Hammersley, R. et Clifford, P. (1971). Markov fields on finite graphs and lattices.

[Hannun *et al.*, 2014] Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A. *et al.* (2014). Deep speech : Scaling up end-to-end speech recognition. *arXiv preprint arXiv :1412.5567*.

[Harris, 1954] Harris, Z. S. (1954). Distributional structure. *Word*, 10(2-3):146–162.

[Hazen *et al.*, 2002] Hazen, T. J., Seneff, S. et Polifroni, J. (2002). Recognition confidence scoring and its use in speech understanding systems. *Computer Speech & Language*, 16(1):49–67.

[Hermansky et Cox, 1991] Hermansky, H. et Cox, L.A., J. (1991). Perceptual Linear Predictive (PLP) Analysis-Resynthesis Technique. *In Applications of Signal Processing to Audio and Acoustics, 1991. Final Program and Paper Summaries., 1991 IEEE ASSP Workshop on*, pages 0–37–0–38.

[Hillard et Ostendorf, 2006] Hillard, D. et Ostendorf, M. (2006). Compensating for word posterior estimation bias in confusion networks. *In Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, volume 1, pages I–I. IEEE.

[Hinton *et al.*, 2012] Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N. *et al.* (2012). Deep neural networks for acoustic modeling in speech recognition : The shared views of four research groups. *Signal Processing Magazine, IEEE*, 29(6):82–97.

[Hinton, 1978] Hinton, G. E. (1978). Relaxation and its role in vision.

[Hinton *et al.*, 2006] Hinton, G. E., Osindero, S. et Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.

[Hinton et Zemel, 1994] Hinton, G. E. et Zemel, R. S. (1994). Autoencoders, minimum description length, and helmholtz free energy. *Advances in neural information processing systems*, pages 3–3.

[Hirschberg *et al.*, 2004] Hirschberg, J., Litman, D. et Swerts, M. (2004). Prosodic and other cues to speech recognition failures. *Speech Communication*, 43(1):155–175.

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

[Hopfield, 1982] Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558.

[Hopfield, 1984] Hopfield, J. J. (1984). Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the national academy of sciences*, 81(10):3088–3092.

[Hovy *et al.*, 2006] Hovy, E., Marcus, M., Palmer, M., Ramshaw, L. et Weischedel, R. (2006). Ontonotes : the 90% solution. *In Proceedings of the human language technology conference of the NAACL, Companion Volume : Short Papers*, pages 57–60. Association for Computational Linguistics.

[Hubel et Wiesel, 1962] Hubel, D. H. et Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106–154.

[Ian Goodfellow, 2016] Ian Goodfellow, Yoshua Bengio, A. C. (2016). Deep learning. Book in preparation for MIT Press.

[Ian Goodfellow et Courville, 2016] Ian Goodfellow, Y. B. et Courville, A. (2016). Deep learning. Book in preparation for MIT Press.

[Ioffe et Szegedy, 2015] Ioffe, S. et Szegedy, C. (2015). Batch normalization : Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv :1502.03167*.

[Jaitly *et al.*, 2012] Jaitly, N., Nguyen, P., Senior, A. W. et Vanhoucke, V. (2012). Application of Pretrained Deep Neural Networks to Large Vocabulary Speech Recognition. *In INTERSPEECH*. Citeseer.

[Jalalvand et Falavigna, 2015] Jalalvand, S. et Falavigna, D. (2015). Stacked auto-encoder for ASR error detection and word error rate prediction. *In INTERSPEECH 2015, 16th Annual Conference of the International Speech Communication Association, Dresden, Germany, September 6-10, 2015*, pages 2142–2146.

[Jean-Paul Haton, 2016] Jean-Paul Haton, Christophe Cerisara, D. F. Y. L. K. S. (2016). *Reconnaissance automatique de la parole : Du Signal à son Interprétation*. Dunod.

[Jelinek, 1976a] Jelinek, F. (1976a). Continuous speech recognition by statistical methods. *Proceedings of the IEEE*, 64(4):532–556.

[Jelinek, 1976b] Jelinek, F. (1976b). Continuous speech recognition by statistical methods. *Proceedings of the IEEE*, 64(4):532–556.

[Jelinek *et al.*, 1977] Jelinek, F., Mercer, R. L., Bahl, L. R. et Baker, J. K. (1977). Perplexity—a measure of the difficulty of speech recognition tasks. *The Journal of the Acoustical Society of America*, 62(S1):S63–S63.

[Ji *et al.*, 2015] Ji, S., Yun, H., Yanardag, P., Matsushima, S. et Vishwanathan, S. V. N. (2015). Wordrank : Learning word embeddings via robust ranking. *CoRR*, abs/1506.02761.

[Jiang, 2005] Jiang, H. (2005). Confidence measures for speech recognition : A survey . *Speech Communication*, 45(4):455 – 470.

[Jordan, 1997] Jordan, M. I. (1997). Serial order : A parallel distributed processing approach. *Advances in psychology*, 121:471–495.

[Juan et Flora, 2015] Juan, S. et Flora, S. (2015). *Exploiting resources from closely-related languages for automatic speech recognition in low-resource languages from Malaysia*. Thèse de doctorat, Grenoble Alpes.

[Kamper *et al.*, 2015] Kamper, H., Wang, W. et Livescu, K. (2015). Deep convolutional acoustic word embeddings using word-pair side information. *In arXiv preprint arXiv :1510.01032*.

[Katz, 1987] Katz, S. M. (1987). Estimation of probabilities from sparse data for the language model component of a speech recognizer. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 35(3):400–401.

[Kemp *et al.*, 1997] Kemp, T., Schaaf, T. *et al.* (1997). Estimating confidence using word lattices. *In EuroSpeech*.

[Kiela *et al.*, 2015] Kiela, D., Hill, F. et Clark, S. (2015). Specializing word embeddings for similarity or relatedness. *In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2044–2048, Lisbon, Portugal. Association for Computational Linguistics.

[Kneser et Ney, 1995] Kneser, R. et Ney, H. (1995). Improved backing-off for m-gram language modeling. *In Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 181–184. IEEE.

[Lafferty *et al.*, 2001] Lafferty, J., McCallum, A. et Pereira, F. C. (2001). Conditional random fields : Probabilistic models for segmenting and labeling sequence data.

[Laurent *et al.*, 2011] Laurent, A., Meignier, S., Merlin, T. et Deléglise, P. (2011). Computer-assisted transcription of speech based on confusion network reordering. *In Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 4884–4887. IEEE.

[Lavergne *et al.*, 2010] Lavergne, T., Cappé, O. et Yvon, F. (2010). Practical very large scale CRFs. *In Proceedings the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 504–513. Association for Computational Linguistics.

[Le et Mikolov, 2014] Le, Q. V. et Mikolov, T. (2014). Distributed representations of sentences and documents. *In ICML*, volume 14, pages 1188–1196.

[Lecouteux *et al.*, 2009] Lecouteux, B., Linares, G. et Favre, B. (2009). Combined low level and high level features for out-of-vocabulary word detection. *In INTERSPEECH*, pages 1187–1190.

[LeCun *et al.*, 1990] LeCun, B. B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. et Jackel, L. D. (1990). Handwritten digit recognition with a back-propagation network. *In Advances in neural information processing systems*. Citeseer.

[LeCun, 1985] LeCun, Y. (1985). Une procedure d'apprentissage pour reseau a seuil asymmetrique (a learning scheme for asymmetric threshold networks).

[LeCun *et al.*, 1998] LeCun, Y., Bottou, L., Bengio, Y. et Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[LeCun *et al.*, 2010] LeCun, Y., Kavukcuoglu, K. et Farabet, C. (2010). Convolutional networks and applications in vision. *In Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pages 253–256. IEEE.

[LeCun *et al.*, 1988] LeCun, Y., Touresky, D., Hinton, G. et Sejnowski, T. (1988). A theoretical framework for back-propagation. *In The Connectionist Models Summer School*, volume 1, pages 21–28.

[LeCun *et al.*, 2012] LeCun, Y. A., Bottou, L., Orr, G. B. et Müller, K.-R. (2012). Efficient backprop. *In Neural networks : Tricks of the trade*, pages 9–48. Springer.

[Lee *et al.*, 2011] Lee, H., Grosse, R., Ranganath, R. et Ng, A. Y. (2011). Unsupervised learning of hierarchical representations with convolutional deep belief networks. *Communications of the ACM*, 54(10):95–103.

[Lee *et al.*, 1990] Lee, K.-F., Hon, H.-W. et Reddy, R. (1990). An overview of the SPHINX speech recognition system. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 38(1):35–45.

[Leggetter et Woodland, 1995] Leggetter, C. J. et Woodland, P. C. (1995). Maximum likelihood linear regression for speaker adaptation of continuous density hidden markov models. *Computer Speech & Language*, 9(2):171–185.

[Levin *et al.*, 2013] Levin, K., Henry, K., Jansen, A. et Livescu, K. (2013). Fixed-dimensional acoustic embeddings of variable-length segments in low-resource settings. *In Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 410–415. IEEE.

[Levy et Goldberg, 2014] Levy, O. et Goldberg, Y. (2014). Dependencybased word embeddings. *In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 302–308.

[Levy *et al.*, 2015] Levy, O., Goldberg, Y. et Dagan, I. (2015). Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

[Levy *et al.*, 2014] Levy, O., Goldberg, Y. et Ramat-Gan, I. (2014). Linguistic regularities in sparse and explicit word representations. *In CoNLL*, pages 171–180.

[Li et Jurafsky, 2015] Li, J. et Jurafsky, D. (2015). Do multi-sense embeddings improve natural language understanding ? *arXiv preprint arXiv :1506.01070*.

[Li *et al.*, 2002] Li, Y., Erdogan, H., Gao, Y. et Marcheret, E. (2002). Incremental on-line feature space mllr adaptation for telephony speech recognition. *In INTERSPEECH*.

[Liang *et al.*, 2014] Liang, Y., Iwano, K. et Shinoda, K. (2014). An efficient error correction interface for speech recognition on mobile touchscreen devices. *In IEEE Workshop on Spoken Language Technology (SLT)*, pages 454–459. IEEE.

[Lin *et al.*, 2015] Lin, Y., Lei, H., Wu, J. et Li, X. (2015). An empirical study on sentiment classification of chinese review using word embedding. *arXiv preprint arXiv :1511.01665*.

[Linzen, 2016] Linzen, T. (2016). Issues in evaluating semantic spaces using word analogies. *arXiv preprint arXiv :1606.07736*.

[Liu et Lane, 2016] Liu, B. et Lane, I. (2016). Attention-based recurrent neural network models for joint intent detection and slot filling. *arXiv preprint arXiv :1609.01454*.

[Lleida et Rose, 1996] Lleida, E. et Rose, R. C. (1996). Likelihood ratio decoding and confidence measures for continuous speech recognition. *In Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on*, volume 1, pages 478–481. IEEE.

[Luong *et al.*, 2013] Luong, M.-T., Socher, R. et Manning, C. D. (2013). Better word representations with recursive neural networks for morphology. *CoNLL-2013*, 104.

[Maas *et al.*, 2012] Maas, A. L., Miller, S. D., O'neil, T. M., Ng, A. Y. et Nguyen, P. (2012). Word-level acoustic modeling with convolutional vector regression. *In ICML Workshop on Representation Learning, Edinburgh, Scotland*.

[Maaten et Hinton, 2008] Maaten, L. v. d. et Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605.

[Mangu *et al.*, 2000] Mangu, L., Brill, E. et Stolcke, A. (2000). Finding consensus in speech recognition : word error minimization and other applications of confusion networks. *Computer Speech & Language*, 14(4):373–400.

[Marcus *et al.*, 1993] Marcus, M. P., Marcinkiewicz, M. A. et Santorini, B. (1993). Building a large annotated corpus of english : The penn treebank. *Computational linguistics*, 19(2):313–330.

[Markel et Gray, 1982] Markel, J. E. et Gray, A. H. (1982). *Linear Prediction of Speech*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[Martinez *et al.*, 2015] Martinez, M. G., Barrault, L., Rousseau, A., Deléglise, P. et Estève, Y. (2015). The lium asr and slt systems for iwslt 2015. *In 12th International Workshop on Spoken Language Translation (IWSLT 2015)*.

[Mauclair, 2006] Mauclair, J. (Décembre 2006). Mesures de confiance en traitement automatique de la parole et applications. *In Thèse de doctora, LIUM Université du Maine, Le Mans France*.

[Medsker et Jain, 1999] Medsker, L. et Jain, L. C. (1999). *Recurrent neural networks : design and applications*. CRC press.

[Meignier et Merlin, 2010] Meignier, S. et Merlin, T. (2010). Lium spkdiarization : an open source toolkit for diarization. *In CMU SPUD Workshop*, volume 2010.

[Memisevic, 2011] Memisevic, R. (2011). Gradient-based learning of higher-order image features. *In Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1591–1598. IEEE.

[Mesnil *et al.*, 2015] Mesnil, G., Dauphin, Y., Yao, K., Bengio, Y., Deng, L., Hakkani-Tur, D., He, X., Heck, L., Tur, G., Yu, D. *et al.* (2015). Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):530–539.

[Mesnil *et al.*, 2013] Mesnil, G., He, X., Deng, L. et Bengio, Y. (2013). Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. *In INTERSPEECH*, pages 3771–3775.

[Mikolov *et al.*, 2013a] Mikolov, T., Chen, K., Corrado, G. et Dean, J. (2013a). Efficient Estimation of Word Representations in Vector Space. *In Proceedings of Workshop at ICLR.*

[Mikolov *et al.*, 2010] Mikolov, T., Karafiát, M., Burget, L., Cernockỳ, J. et Khudanpur, S. (2010). Recurrent neural network based language model. *In Interspeech*, volume 2, page 3.

[Mikolov *et al.*, 2011a] Mikolov, T., Kombrink, S., Burget, L., Černockỳ, J. et Khudanpur, S. (2011a). Extensions of recurrent neural network language model. *In 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5528–5531. IEEE.

[Mikolov *et al.*, 2011b] Mikolov, T., Kombrink, S., Burget, L., Cernocky, J. H. et Khudanpur, S. (2011b). Extensions of recurrent neural network language model. *In Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5528–5531. IEEE.

[Mikolov *et al.*, 2013b] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. et Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. *In Advances in neural information processing systems*, pages 3111–3119.

[Mikolov *et al.*, 2013c] Mikolov, T., Yih, W.-t. et Zweig, G. (2013c). Linguistic regularities in continuous space word representations. *In HLT-NAACL*, pages 746–751.

[Minsky et Papert, 1969] Minsky, M. et Papert, S. A. (1969). Perceptrons. mit press.

[Mnih et Hinton, 2007] Mnih, A. et Hinton, G. (2007). Three new graphical models for statistical language modelling. *In Proceedings of the 24th international conference on Machine learning*, pages 641–648. ACM.

[Mnih et Teh, 2012] Mnih, A. et Teh, Y. W. (2012). A fast and simple algorithm for training neural probabilistic language models. *arXiv preprint arXiv :1206.6426.*

[Mohri *et al.*, 2002] Mohri, M., Pereira, F. et Riley, M. (2002). Weighted finite-state transducers in speech recognition. *Computer Speech and Language*, 16(1):69 – 88.

[Moreau *et al.*, 2000] Moreau, N., Charlet, D. et Jouvet, D. (2000). Confidence measure and incremental adaptation for the rejection of incorrect data. *In Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*, volume 3, pages 1807–1810. IEEE.

[Moreno *et al.*, 2001] Moreno, P. J., Logan, B. et Raj, B. (2001). A boosting approach for confidence scoring. *In INTERSPEECH*, pages 2109–2112.

[Morin et Bengio, 2005] Morin, F. et Bengio, Y. (2005). Hierarchical probabilistic neural network language model. *In Aistats*, volume 5, pages 246–252. Citeseer.

[Murphy *et al.*, 1999] Murphy, K. P., Weiss, Y. et Jordan, M. I. (1999). Loopy belief propagation for approximate inference : An empirical study. *In Proceedings*

*of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 467–475. Morgan Kaufmann Publishers Inc.

[Nasr *et al.*, 2011] Nasr, A., Béchet, F., Rey, J.-F., Favre, B. et Le Roux, J. (2011). 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*, pages 86–91. Association for Computational Linguistics.

[Neyman et Pearson, 1992] Neyman, J. et Pearson, E. S. (1992). On the problem of the most efficient tests of statistical hypotheses. *In Breakthroughs in Statistics*, pages 73–108. Springer.

[Ogawa et Hori, 2015] Ogawa, A. et Hori, T. (2015). Asr error detection and recognition rate estimation using deep bidirectional recurrent neural networks. *In Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 4370–4374.

[Olshen *et al.*, 1984] Olshen, L., Stone, C. J. *et al.* (1984). Classification and regression trees. *Wadsworth International Group*, 93(99):101.

[Parada *et al.*, 2010] Parada, C., Dredze, M., Filimonov, D. et Jelinek, F. (2010). Contextual Information Improves OOV Detection in Speech. *In Human Language Technologies : The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

[Pennington *et al.*, 2014] Pennington, J., Socher, R. et Manning, C. D. (2014). Glove : Global vectors for word representation. *In Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, volume 12.

[Pincus *et al.*, 2013] Pincus, E., Stoyanchev, S. et Hirschberg, J. (2013). Exploring features for localized detection of speech recognition errors. *In Proc. of the SIGDIAL Conference. ACL*, pages 132–136.

[Pitz *et al.*, 2000] Pitz, M., Wessel, F. et Ney, H. (2000). Improved mllr speaker adaptation using confidence measures for conversational speech recognition. *In INTERSPEECH*, pages 548–551.

[Plaut *et al.*, 1986] Plaut, D. C. *et al.* (1986). Experiments on learning by back propagation.

[Povey *et al.*, 2011] Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P. *et al.* (2011). The kaldi speech recognition toolkit. *In IEEE 2011 workshop on automatic speech recognition and understanding*, numéro EPFL-CONF-192584. IEEE Signal Processing Society.

[Povey et Woodland, 2002] Povey, D. et Woodland, P. C. (2002). Minimum phone error and i-smoothing for improved discriminative training. *In Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*, volume 1, pages I–105. IEEE.

[Rabiner, 1989a] Rabiner, L. (1989a). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

[Rabiner et Juang, 1993] Rabiner, L. et Juang, B.-H. (1993). Fundamentals of speech recognition.

[Rabiner, 1989b] Rabiner, L. R. (1989b). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

[Rao *et al.*, 2015] Rao, K., Peng, F., Sak, H. et Beaufays, F. (2015). Grapheme-to-phoneme conversion using long short-term memory recurrent neural networks. *In 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4225–4229. IEEE.

[Raymond et Riccardi, 2007] Raymond, C. et Riccardi, G. (2007). Generative and discriminative algorithms for spoken language understanding. *In Interspeech*, pages 1605–1608.

[Ren *et al.*, 2016] Ren, Y., Wang, R. et Ji, D. (2016). A topic-enhanced word embedding for twitter sentiment classification. *Information Sciences*, 369:188–198.

[Rifai *et al.*, 2011] Rifai, S., Vincent, P., Muller, X., Glorot, X. et Bengio, Y. (2011). Contractive auto-encoders : Explicit invariance during feature extraction. *In Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 833–840.

[Rose *et al.*, 1995] Rose, R. C., Juang, B.-H. et Lee, C.-H. (1995). A training procedure for verifying string hypotheses in continuous speech recognition. *In Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 281–284. IEEE.

[Rosenblatt, 1957] Rosenblatt, F. (1957). *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory.

[Rosenblatt, 1958] Rosenblatt, F. (1958). The perceptron : a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6): 386.

[Rousseau *et al.*, 2014] Rousseau, A., Boulianne, G., Deléglise, P., Estève, Y., Gupta, V. et Meignier, S. (2014). LIUM and CRIM ASR System Combination for the REPERE Evaluation Campaign. *In Text, Speech and Dialogue*, pages 441–448. Springer.

[Rudy et Taylor, 2014] Rudy, J. et Taylor, G. (2014). Generative class-conditional denoising au-toencoders. *arXiv preprint arXiv :1412.7009*.

[Rumelhart *et al.*, 1985] Rumelhart, D. E., Hinton, G. E. et Williams, R. J. (1985). Learning internal representations by error propagation. Rapport technique, DTIC Document.

[Rumelhart *et al.*, 1986] Rumelhart, D. E., Hinton, G. E. et Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323:533–536.

[Rumelhart *et al.*, 1988] Rumelhart, D. E., Hinton, G. E. et Williams, R. J. (1988). Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1.

[Sainath *et al.*, 2013] Sainath, T. N., Mohamed, A.-r., Kingsbury, B. et Ramabhadran, B. (2013). Deep convolutional neural networks for lvcsr. *In 2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8614–8618. IEEE.

[Sak *et al.*, 2014] Sak, H., Senior, A. W. et Beaufays, F. (2014). Long short-term memory recurrent neural network architectures for large scale acoustic modeling. *In INTERSPEECH*, pages 338–342.

[Salakhutdinov et Hinton, 2009] Salakhutdinov, R. et Hinton, G. E. (2009). Deep boltzmann machines. *In AISTATS*, volume 1, page 3.

[Sanchis *et al.*, 2003] Sanchis, A., Juan, A. et Vidal, E. (2003). Estimating confidence measures for speech recognition verification using a˜ smoothed naive bayes model. *In Iberian Conference on Pattern Recognition and Image Analysis*, pages 910–918. Springer.

[Sarikaya *et al.*, 2005] Sarikaya, R., Gao, Y., Picheny, M. et Erdogan, H. (2005). Semantic confidence measurement for spoken dialog systems. *Speech and Audio Processing, IEEE Transactions on*, 13(4):534–545.

[Schaul *et al.*, 2013] Schaul, T., Zhang, S. et LeCun, Y. (2013). No more pesky learning rates. *ICML (3)*, 28:343–351.

[Schuster et Paliwal, 1997] Schuster, M. et Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

[Schwenk, 2007] Schwenk, H. (2007). Continuous space language models. *Computer Speech & Language*, 21(3):492–518.

[Schwenk, 2013] Schwenk, H. (2013). CSLM-a modular open-source continuous space language modeling toolkit. *In INTERSPEECH*, pages 1198–1202.

[Schwenk *et al.*, 2006] Schwenk, H., Dchelotte, D. et Gauvain, J.-L. (2006). Continuous Space Language Models for Statistical Machine Translation. *In Proceedings of the COLING/ACL on Main Conference Poster Sessions*, COLING-ACL '06, pages 723–730, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Seigel, 2013] Seigel, M. S. (2013). *Confidence Estimation for Automatic Speech Recognition Hypotheses*. Thèse de doctorat, Ph. D. thesis, University of Cambridge.

[Sermanet *et al.*, 2012] Sermanet, P., Chintala, S. et LeCun, Y. (2012). Convolutional neural networks applied to house numbers digit classification. *In Pattern*

*Recognition (ICPR), 2012 21st International Conference on*, pages 3288–3291. IEEE.

[Servan *et al.*, 2006] Servan, C., Raymond, C., Béchet, F. et Nocéra, P. (2006). Conceptual decoding from word lattices : application to the spoken dialogue corpus media. *In The Ninth International Conference on Spoken Language Processing (Interspeech 2006-ICSLP)*.

[Simonnet *et al.*, 2015] Simonnet, E., Camelin, N., Deleglise, P. et Esteve, Y. (2015). Exploring the use of attention-based recurrent neural networks for spoken language understanding. *In NIPS*.

[Socher *et al.*, 2011] Socher, R., Lin, C. C., Manning, C. et Ng, A. Y. (2011). Parsing natural scenes and natural language with recursive neural networks. *In Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 129–136.

[Soto *et al.*, 2014] Soto, V., Cooper, E., Mangu, L., Rosenberg, A. et Hirschberg, J. (2014). Rescoring confusion networks for keyword search. *In Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 7088–7092. IEEE.

[Stemmer *et al.*, 2002] Stemmer, G., Steidl, S., Nöth, E., Niemann, H. et Batliner, A. (2002). Comparison and combination of confidence measures. *In Text, Speech and Dialogue*, pages 181–188. Springer.

[Stolcke *et al.*, 1997] Stolcke, A., Konig, Y. et Weintraub, M. (1997). Explicit word error minimization in n-best list rescoring. *In Eurospeech*, volume 97, pages 163–166. Citeseer.

[Stoyanchev *et al.*, 2012a] Stoyanchev, S., Salletmayr, P., Yang, J. et Hirschberg, J. (2012a). Localized detection of speech recognition errors. *In Spoken Language Technology Workshop (SLT), 2012 IEEE*, pages 25–30.

[Stoyanchev *et al.*, 2012b] Stoyanchev, S., Salletmayr, P., Yang, J. et Hirschberg, J. (2012b). Localized detection of speech recognition errors. *In Spoken Language Technology Workshop (SLT), 2012 IEEE*, pages 25–30. IEEE.

[Sutskever *et al.*, 2013] Sutskever, I., Martens, J., Dahl, G. E. et Hinton, G. E. (2013). On the importance of initialization and momentum in deep learning. *ICML (3)*, 28:1139–1147.

[Tam *et al.*, 2014] Tam, Y.-C., Lei, Y., Zheng, J. et Wang, W. (2014). Asr error detection using recurrent neural network language model and complementary asr. *In Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 2312–2316. IEEE.

[Tang *et al.*, 2016] Tang, D., Wei, F., Qin, B., Yang, N., Liu, T. et Zhou, M. (2016). Sentiment embeddings with applications to sentiment analysis. *IEEE Transactions on Knowledge and Data Engineering*, 28(2):496–509.

[Thibodeau-Laufer, 2014] Thibodeau-Laufer, E. (2014). Algorithmes d'apprentissage profonds supervisés et non-supervisés : applications et résultats théoriques.

[Thiolliere *et al.*, 2015] Thiolliere, R., Dunbar, E., Synnaeve, G., Versteegh, M. et Dupoux, E. (2015). A hybrid dynamic time warping-deep neural network architecture for unsupervised acoustic modeling. *In Proc. Interspeech*.

[Thomas et Trancoso, 2010] Thomas, P. et Trancoso, I. (2010). Improving ASR error detection with non-decoder based features. *In Interspeech*, pages 1950–1953.

[Tjong Kim Sang et Buchholz, 2000] Tjong Kim Sang, E. F. et Buchholz, S. (2000). Introduction to the conll-2000 shared task : Chunking. *In Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning-Volume 7*, pages 127–132. Association for Computational Linguistics.

[Tjong Kim Sang et De Meulder, 2003] Tjong Kim Sang, E. F. et De Meulder, F. (2003). Introduction to the conll-2003 shared task : Language-independent named entity recognition. *In Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics.

[Tompson *et al.*, 2014] Tompson, J. J., Jain, A., LeCun, Y. et Bregler, C. (2014). Joint training of a convolutional network and a graphical model for human pose estimation. *In* Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D. et Weinberger, K. Q., éditeurs : *Advances in Neural Information Processing Systems 27*, pages 1799–1807. Curran Associates, Inc.

[Turian *et al.*, 2010] Turian, J., Ratinov, L. et Bengio, Y. (2010). Word representations : A simple and general method for semisupervised learning. pages 384–394.

[Turney, 2006] Turney, P. D. (2006). Similarity of semantic relations. *Computational Linguistics*, 32(3):379–416.

[Turney, 2012] Turney, P. D. (2012). Domain and function : A dual-space model of semantic relations and compositions. *Journal of Artificial Intelligence Research*, 44:533–585.

[Turney *et al.*, 2010] Turney, P. D., Pantel, P. *et al.* (2010). From frequency to meaning : Vector space models of semantics. *Journal of artificial intelligence research*, 37(1):141–188.

[Uhrik et Ward, 1997] Uhrik, C. et Ward, W. (1997). Confidence metrics based on n-gram language model backoff behaviors. *In EUROSPEECH*.

[Uszkoreit et Brants, 2008] Uszkoreit, J. et Brants, T. (2008). Distributed word clustering for large scale class-based language modeling in machine translation. *In ACL*, pages 755–762.

[Vapnik et Kotz, 1982] Vapnik, V. N. et Kotz, S. (1982). *Estimation of dependences based on empirical data*, volume 40. Springer-verlag New York.

[Vincent *et al.*, 2008] Vincent, P., Larochelle, H., Bengio, Y. et Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. *In Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM.

[Vincent *et al.*, 2010] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y. et Manzagol, P.-A. (2010). Stacked denoising autoencoders : Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408.

[Walker *et al.*, 2004] Walker, W., Lamere, P., Kwok, P., Raj, B., Singh, R., Gouvea, E., Wolf, P. et Woelfel, J. (2004). Sphinx-4 : A flexible open source framework for speech recognition.

[Wang *et al.*, 2014] Wang, J., Song, Y., Leung, T., Rosenberg, C., Wang, J., Philbin, J., Chen, B. et Wu, Y. (2014). Learning fine-grained image similarity with deep ranking. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1386–1393.

[Weintraub *et al.*, 1997] Weintraub, M., coise Beaufays, F., Rivlin, Z., Konig, Y. et Stolcke, A. (1997). Neural-network based measures of confidence for word recognition. *hypothesis (according to our definition)*, 2:3.

[Weiss et Freeman, 2001] Weiss, Y. et Freeman, W. T. (2001). Correctness of belief propagation in gaussian graphical models of arbitrary topology. *Neural computation*, 13(10):2173–2200.

[Wessel *et al.*, 1999] Wessel, F., Macherey, K. et Ney, H. (1999). A comparison of word graph and n-best list based confidence measures. *In EuroSpeech*.

[Wessel *et al.*, 2001] Wessel, F., Schluter, R., Macherey, K. et Ney, H. (2001). Confidence measures for large vocabulary continuous speech recognition. *IEEE Transactions on speech and audio processing*, 9(3):288–298.

[Weston *et al.*, 2011] Weston, J., Bengio, S. et Usunier, N. (2011). Wsabie : Scaling up to large vocabulary image annotation. *In IJCAI*, volume 11, pages 2764–2770.

[Woodland *et al.*, 1999] Woodland, P. C., Odell, J., Hain, T., Moore, G. L., Niesler, T., Tuerk, A. et Whittaker, E. W. (1999). Improvements in accuracy and speed in the htk broadcast news transcription system. *In EUROSPEECH*.

[Yao *et al.*, 2014] Yao, K., Peng, B., Zhang, Y., Yu, D., Zweig, G. et Shi, Y. (2014). Spoken language understanding using long short-term memory neural networks. *In Spoken Language Technology Workshop (SLT), 2014 IEEE*, pages 189–194. IEEE.

[Yao et Zweig, 2015] Yao, K. et Zweig, G. (2015). Sequence-to-sequence neural net models for grapheme-to-phoneme conversion. *arXiv preprint arXiv :1506.00196*.

[Yik-Cheung *et al.*, 2014] Yik-Cheung, T., Lei, Y., Zheng, J. et Wang, W. (2014). ASR error detection using recurrent neural network language model and complementary ASR. *In Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 2312–2316.

[Young et Woodland, 1994] Young, S. J. et Woodland, P. C. (1994). State clustering in hidden markov model-based continuous speech recognition. *Computer Speech & Language*, 8(4):369–383.

[Yu *et al.*, 2011] Yu, D., Li, J. et Deng, L. (2011). Calibration of Confidence Measures in Speech Recognition. *In IEEE Transactions on Audio, Speech, and Language Processing*, volume 19, pages 2461–2473.

[Zbontar et LeCun, 2016] Zbontar, J. et LeCun, Y. (2016). Stereo matching by training a convolutional neural network to compare image patches. *Journal of Machine Learning Research*, 17:1–32.

[Zeiler, 2012] Zeiler, M. D. (2012). Adadelta : an adaptive learning rate method. *arXiv preprint arXiv :1212.5701.*

[Zeiler et Fergus, 2014] Zeiler, M. D. et Fergus, R. (2014). Visualizing and understanding convolutional networks. *In European Conference on Computer Vision*, pages 818–833. Springer.

[Zhang et Rudnicky, 2001] Zhang, R. et Rudnicky, A. I. (2001). Word level confidence annotation using combinations of features.