Université du Québec
Institut National de la Recherche Scientifique
Centre Énergie Matériaux Télécommunications

ANALYSE SÉMANTIQUE POUR SYSTÈMES DE DIALOGUE VERBAUX

Par

Yacine Benahmed

Thèse présentée pour l'obtention du grade de
Philosophiæ Doctor (Ph. D.) en télécommunications

Jury d'évaluation

Président du jury et examinateur interne	Prof. Falk, Tiago Institut National de la Recherche Scientifique
Examinateur externe	Prof. Cardinal, Patrick École de technologie supérieure
Examinateur externe	Prof. Tadj, Chakib École de technologie supérieure
Directeur de recherche	Prof. O'Shaughnessy, Douglas Institut National de la Recherche Scientifique
Codirecteur de recherche	Prof. Selouani, Sid-Ahmed Université de Moncton

DECLARATION

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other University. This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration, except where specifically indicated in the text.

Yacine Benahmed

REMERCIEMENTS

J’aimerais remercier mon directeur de thèse Prof. Douglas D. O’Shaughnessy ainsi que mon codirecteur de thèse Prof. Sid Ahmed Selouani pour leurs conseils et leur support tout au long de cette thèse.

J’aimerais aussi remercier les membres du comité d’évaluation Prof. Tiago Falk, Prof. Patrick Cardinal et Prof. Chakib Tadj d’avoir accepté de donner de leur temps malgré leur horaire chargé afin d’évaluer ce travail doctoral ainsi que pour leurs conseils et suggestions.

J’aimerais aussi remercier ma famille et ma belle-famille pour leurs encouragements et leur soutien tout au long de ce processus et de m’avoir encouragé à persévérer malgré les nombreuses difficultés rencontrées.

J’aimerais de plus remercier Emmanuelle d'avoir pris le temps de faire un véritable travail d’édition pour cette thèse, ce qui a permis de la rendre beaucoup plus lisible étant donné mes tendances aux phrases longues et aux tournures de phrases non conventionnelles...

Je voudrais aussi remercier mes amis pour m’avoir tenu dans le droit chemin et de m’avoir constamment demandé « pis, quand-est-ce tu finis ? », ce à quoi je peux maintenant répondre « là ».

Finalement, j’aimerais remercier ma conjointe, Anne Fauré, que j’ai rencontrée durant mon parcours et qui se retrouvait dans le même bateau que moi. Merci de m’avoir prêté ton oreille et pour toutes tes lectures et relectures de mon texte. Finalement, merci pour le plus beau cadeau que tu pouvais nous faire, notre fils Michel-Ange.

LISTE DES PUBLICATIONS

Contributions principales à la these

1. Y. Benahmed, S.-A. Selouani, D. O'Shaughnessy, et A. H. Abolhassani, « Real-Life Speech-Enabled System to Enhance Interaction with RFID Networks in Noisy Environments », dans *International Conference on Acoustics, Speech and Signal Processing*, Prague, Czech Republic, 2011, p. 17811784.
2. Y. Benahmed, S.-A. Selouani, et D. O'Shaughnessy, « Ontology-based pattern generator and root semantic analyser for spoken dialogue systems », dans *Electrical & Computer Engineering (CCECE), 2012 25th IEEE Canadian Conference on*, 2012, p. 14.
3. Y. Benahmed, S.-A. Selouani, et D. O'Shaughnessy, « A Bin-Based Ontological Framework for Low-Resource N-Gram Smoothing in Language Modelling », dans *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, 2014, p. 49184922.
4. Y. Benahmed, S.-A. Selouani, et D. O'Shaughnessy, « Evaluation of Graph Metrics for Optimizing Bin-Based Ontologically Smoothed Language Models », dans *European Signal Processing Conference*, Budapest, Hongrie, 2016.

Contributions secondaires

5. S. A. Selouani, T.-H. Lê, Y. Benahmed, et D. O'Shaughnessy, « Enhanced Speech-Enabled Tools for Intelligent and Mobile E-Learning Applications », *Technologies Shaping Instruction and Distance Education: New Studies and Utilizations: New Studies and Utilizations*, p. 147165, 2010.
6. Ayed, S.-A. Selouani, M. Kardouchi, et Y. Benahmed, « Radiological image classification using HMMs and Shape contexts », dans *Information Science, Signal Processing and their Applications (ISSPA), 2012 11th International Conference on*, 2012, p. 87-91.

7. Y. Benahmed, S.-A. Selouani, et D. O'Shaughnessy, « Effects of discriminative training on the RACAD corpus of the French language spoken in the Canadian province of New-Brunswick », dans *Information Science, Signal Processing and their Applications (ISSPA), 2012 11th International Conference on*, 2012, p. 695699.
8. S. Gharsellaoui, A. O. Dahmane, Y. Alotaibi, S. A. Selouani, A. B. Ayed, et Y. Benahmed, « A Rhythm-Based Analysis of Arabic Native and Non-Native Speaking Styles », *International Journal of Signal Processing Systems*, vol. 1, n° 2, p. 202207, 2013.

RÉSUMÉ

Les modèles de langages statistiques font partie intégrante de nombreuses applications telles que la reconnaissance automatique de la parole, la reconnaissance automatique de l'écriture, la saisie prédictive, la traduction automatique du langage, etc. Malgré les développements récents dans le domaine des réseaux de neurones profonds, les *N*-grammes demeurent tout de même pertinentes dans des domaines où les ressource linguistiques ou computationnelles sont limitées. L'objectif de cette thèse est de présenter un nouvel algorithme de lissage des *N*-grammes utilisant l'information sémantique lattente aux ontologies. Comme le langage offre une très grande flexibilité dans sa façon d'exprimer une idée, nous utiliserons les relations sémantiques entre les *N*-grammes observées et les *N*-grammes non-observées afin d'estimer les probabilités d'observation de ces dernières dans de nouvelles conditions. Nos résultats démontrent qu'il est possible d'obtenir une réduction de 0,93 bit de l'entropie du modèle de langage. Par ailleurs, afin d'optimiser ces modèles de langages, nous proposons d'évaluer la performance de divers algorithmes évolutionnaires. Nos résultats montrent que l'optimimsation par algorithmes génétiques permet de couvrir une plus grande partie de l'espace de recherche, au coût d'une plus grande variance dans la solution obtenue. Pour sa part, l'optimisation par essaims particulaires fait preuve d'une moins grande variance dans les solutions obtenues, tout en demeurant aussi performante que l'optimisation par algorithme génétique.

Mots clés : Modèles de langage, ontologies, n-grammes, reconnaissance automatique de la parole, algorithmes évolutionnaires.

ABSTRACT

Statistical language models are an integral part of many applications such as automatic speech recognition, automatic recognition of writing, predictive input, automatic translation of the language, and so on. Despite recent developments in the field of deep neural networks, N-grams remain relevant in areas where linguistic or computational resources are limited. The objective of this thesis is to present a new algorithm of smoothing N-grams using the semantic information latent to the ontologies. Since language offers a very high degree of flexibility in the way it expresses an idea, we will use the semantic relations between the observed N-grams and the non-observed N-grams in order to estimate the observation probabilities of the observed N-grams in new conditions. Our results show that it is possible to obtain a 0.93 bit reduction of the entropy of the language model. Moreover, in order to optimize these language models, we propose to evaluate the performance of various evolutionary algorithms. Our results show that optimization by genetic algorithms makes it possible to cover a larger part of the research space at the cost of a greater variance in the solution obtained. For its part, the optimization by particulate swarms shows less variance in the solutions obtained, while remaining as efficient as the optimization by genetic algorithm.

Keywords: Language models, ontologies, n-grams, automatic speech recognition, evolutionary algorithms.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ACCRONYMES

ADL	Allocation de Dirichlet latente
AE	Algorithmes évolutionnaires
AG	Algorithme génétique
AIML	*Artificial Intelligence Markup Language*
API	Alphabet phonétique international
ASR	*Automatic Speech Recognition*
BBOSLM	*Bin Based Ontologically Smoothed Language Model*
C&C	*Command and Control*
DCT	*Discrete Cosine Transform*
DFT	*Discrete Fourier Transform*
EA	*Evolutionary Algorithms*
EM	Expectation-Maximisation
EMV	Estimation du maximum de vraisemblance
EVD	*EigenValue Decomposition*
GA	*Genetic Algorithm*
GT	Good-Turing
HITS	*Hyperlink Induced Topic Search*
HMM	*Hidden Markov Models*
IVR	*Interactive Voice Response*
JM	Jelienek Mercer
KLT	*Karhunen-Loève Transform*
KN	Kneser Ney
LM	*Language Model*
LSTM	*Long Short Term Memory*
MDL	*Minimum Description Length*

MHV	Mots hors vocabulaire
MKN	*Modified Kneser Ney*
ML	Modèle de langage
MLE	*Maximul Likelyhood Estimation*
OEP	Optimisation par essaims particulaires
OOV	*Out Of Vocabulary*
OWL	*Web Ontology Language*
PCA	*Principal Component Analysis*
PCs	*Principal Components*
PESQ	*Perceptual Evaluation of Speech Quality*
POS	*Part Of Speech*
PP	Perplexité
PPL	*Perplexity*
PR	*Page Rank*
PSO	*Particle Swarm Optimisation*
PY	Pitman-Yor
PYH	Pitman-Yor hiérarchique
RAP	Reconnaissance automatique de la parole
RNN	*Reccurent Neural Network*
RNP	Réseau de neurones profond
RNR	Réseau de neurones récurrents
RFID	*Radio Frequency IDentification*
SS	*Spectral Slope*
SUO-KIF	*Standard Upper Ontology Knowledge Interchange Format*
TALN	Traitement automatique du langage naturel
TEM	Taux d'erreur des mots
TF-IDF	*Term Frequency Inverse Document Frequency*
VER-KLT	Variance de l'erreur de reconstruction de la Karhunen-Loève Transform

VRE *Variance of the Reconstruction Error*

VRE-KLT *Variance of the Reconstruction Error of Karhunen-Loève Transform*

WB Witten Bell

WD *Weighed Degree*

WER *Word Error Rate*

WRR *Word Recognition Rate*

WSS *Weighted Spectral Slope*

WWW *World Wide Web*

XML *eXtended Markup Language*

Analyse sémantique pour systèmes de dialogue verbaux
Synthèse

Felix qui potuit rerum cognoscere causas
Virgile
Sum ergo cogito. Cogito ergo dubito

1 INTRODUCTION

1.1 La modélisation du langage

De façon générale lorsque nous modélisons un langage naturel[1], nous tentons de capturer, de décrire et d'exploiter les régularités et les structures intrinsèques à ce langage. En effet, si le langage était sans règles, sans objectifs, il nous serait très difficile de communiquer efficacement. À son plus simple, un modèle de langage peut être constitué d'une liste des mots et des phrases permises par ledit langage. En effet, dans le contexte de la reconnaissance automatique de la parole, à ses tous débuts, la modélisation du langage consistait en listes de mots (des chiffres) permis [1]. Ce n'est que plus tard, alors que les capacités calculatoires le rendirent possible, que les modèles de langage commencèrent à se complexifier.

D'un point de vue philosophique, les premiers modèles de langage formels auxquels nous sommes initiés sont les dictionnaires ainsi que les manuels de syntaxe et de grammaire. Ceux-ci édictent les constructions du langage, les mots qui sont permis selon un ensemble de règles strictes, formelles et laissant peu de place à l'interprétation. C'est pourquoi les premiers modèles de langage informatisés étaient principalement basés sur la grammaire et étaient majoritairement l'affaire des linguistes. Cependant, on le sait bien, le langage parlé n'est pas, dans la majorité des cas, assujetti à autant de règles et restrictions qu'une rédaction écrite : il laisse beaucoup plus de place à l'improvisation, à la spontanéité, l'immédiateté, voir même l'invention (pensons, par exemple, aux néologismes). Cela dit, il est clair que, même avec les meilleures intentions du

[1] Pour fin de simplification, nous utiliserons le terme plus général « langage » tout au long de cette thèse. Notons que les techniques décrites ici peuvent d'ailleurs tout aussi s'appliquer (avec modifications au besoin) aux langages dits artificiels.

monde, et des ressources temporelles infinies, un modèle de langage conçu à la main ne pourra jamais prétendre répondre à une tâche universelle[2].

Voilà donc pourquoi la grande majorité de la recherche actuelle dans le domaine se base sur une approche probabiliste et où on s'appuiera sur d'énormes corpora[3] pour en déduire la structure probabiliste et, ainsi, être en mesure de « minimiser » l'incertitude par rapport à ce qui a été dit.

Outre la reconnaissance automatique de la parole, les modèles de langage sont utilisés autant en traduction automatique, qu'en récapitulation de texte, reconnaissance de l'écriture, traitement du langage, recherche d'information, etc.

L'application principale visée par cette thèse relève du domaine de la reconnaissance automatique de la parole (RAP). La reconnaissance automatique de la parole s'appuie d'une part d'un modèle acoustique et, d'autre part, d'un modèle de langage et où on cherche à résoudre l'équation suivante :

$$\widehat{W} = \underset{W \in \mathcal{L}}{\operatorname{argmax}} P(W|O). \tag{1.1}$$

Celle-ci se lit comme suit : la phrase reconnue est la phrase ayant la plus haute probabilité étant donné les sons « entendus ».

Il a été démontré à plusieurs occasions que les progrès réalisés en modélisation du langage aboutissent généralement à des améliorations de performance en RAP (diminution du taux d'erreur de reconnaissance des mots, insertions [mots en trop], substitutions, suppressions). Cela se traduit par la suite à des améliorations dans le domaine de la compréhension de la parole (pensez par exemple à Siri [2], mais qui comprend ce que vous dites). En gros, le modèle de langage sert à aider le modèle acoustique en excluant les mots qui sont très peu probables (et, par conséquent, les séquences phonémiques peu probables). Par exemple, sans connaissance du contexte, ou de ces probabilités, il nous serait impossible de savoir si la séquence (observée) suivante de phonèmes (en format API) : `e.tɔ.na.mɑ̃.mɔ.nɔ.tɔn.e.las,` une phrase homophonique, désigne :

[2] Ce qui nous fait forcément penser à la fameuse citation de Jelinek (1985 : traduction libre) « À chaque fois que je vire un linguiste, les performances de la reconnaissance automatique de la parole augmentent ».

[3] Essentiellement des recueils de texte.

Étonnamment monotone et lasse

ou

Est ton âme en mon automne hélas [3].

Un autre exemple qui illustre bien l'importance et le pouvoir prédictif du modèle de langage est justement lorsque l'on a une conversation avec quelqu'un que l'on connaît bien et où nous sommes en mesure de compléter les phrases de notre interlocuteur. On peut donc considérer notre connaissance de l'interlocuteur (son contexte, son vocabulaire habituel, sa façon de présenter/construire ses énoncés) comme un modèle de langage. Ce même modèle de langage nous sert bien aussi lorsque nous ne sommes pas sûr d'avoir bien entendu (signal acoustique déformé ou incomplet), mais nous pouvons exclure certains mots avec un bon degré de certitude (réduction de l'espace de recherche acoustique, détection/correction d'erreur). Pour continuer notre analogie précédente, nous arrivons à prédire ce que notre interlocuteur va dire, car nous l'avons plus souvent entendu dire quelque chose d'une manière, plutôt que d'une autre (probabilité plus élevée, basée sur un nombre d'observations). Un autre exemple serait l'usage d'un patois où les interlocuteurs ont l'habitude d'utiliser des mots et expressions (souvent dites, compte élevé) et donc très probables et, par conséquent, faciles à deviner.

Toutefois, malgré notre maîtrise de la parole et du langage, nous ne sommes toujours pas en mesure, en tant qu'être humain, de tout prédire avec exactitude, et tout comme nous, les ordinateurs ont eux aussi de la difficulté, ce qui nous arrange (les chercheurs), puisqu'on peut toujours faire mieux.

1.2 Motivation

La parole est notre moyen privilégié de communication interpersonnelle et nous permet de décrire le monde ainsi que d'en prendre possession [4]. Le langage nous permet quant à lui de comprendre, de **nous** comprendre [5], [6] et de comprendre non seulement les autres, mais aussi leur vision du monde. Le langage participe donc à la construction de chaque individu en fonction de sa culture, de son identité. Ces mécanismes se mettent en place dès notre jeune âge d'abord avec les membres de notre famille puis pour socialiser à l'extérieur de la cellule familiale et ce processus se poursuit toute la vie pour échanger notre sagesse, nos connaissances, des informations et des données.

Cette facilité d'interaction permet d'expliquer pourquoi de nombreux efforts dans le domaine de l'**interaction automatique par la parole**[4] sont mis en place afin de rendre le plus naturel possible nos interactions avec les divers systèmes informatisés qui sont à notre disposition. Pour preuve, ces systèmes font de plus en plus partie de nos vies, elles-mêmes de plus en plus connectées, ne serait-ce que par l'utilisation de nos téléphones, dits, intelligents, qui proposent des « assistants » virtuels de plus en plus ambitieux et sophistiqués [2], [7], [8]. Nous retrouvons aussi ces capacités d'interaction vocale dans la majorité des voitures (un fort argument de vente de nos jours), qui permettent, en principe, de réduire la distraction des chauffeurs. De plus en plus de centres d'appels nous redirigent vers des agents conversationnels en ligne de premier front. La domotique quant à elle, qui se démocratise de plus en plus, en voit ses fonctionnalités augmentées. À titre d'exemple, à partir du moment où un utilisateur peut interroger le système de façon naturelle, celui-ci pourrait poser la question suivante à « son domicile » :

« Est-ce que la porte est barrée [5]? »

En supposant que l'individu soit couché, cela est, *ipso facto,* à la fois beaucoup plus facile, rapide et pratique que de se lever et d'aller vérifier si la porte est bel et bien verrouillée.

D'autres applications se prêtent bien à l'interaction par la parole ; lorsque nous avons les mains pleines ou encore lorsque le clavier et la souris ne sont pas des modalités idéales. D'ailleurs, dans nos travaux de maîtrise [9] nous montrons comment une modalité vocale peut être équivalente au clavier et à la souris pour certaines tâches et [10] démontre que pour certains cas qu'une interface vocale est plus efficace que le clavier et la souris.

Évidemment, si nous portons notre regard vers le futur nous ne pouvons éviter de penser à la robotique, un rêve presque aussi ancien que la civilisation moderne [11], [12] elle-même. Aujourd'hui, nous avons nos aspirateurs robotisés, ou encore des « compagnons » robotiques tel que Zenbo [13], Pepper [14] ou encore Anybot [15], demain nous aurons, du moins, *nous* le

[4] Nous préférons ici parler d'interaction automatique par la parole plutôt que de reconnaissance automatique de la parole, comme il était courant par le passé, puisque cette première expression est maintenant l'objectif principal de tous les efforts investis dans ce vaste champ de recherche qu'est la réplication de l'intelligence humaine.

[5] Synonyme de l'adjectif verrouillé en acadien/français canadien. Cet exemple illustre bien une des difficultés inhérentes à l'interaction par la parole, à savoir, comment traiter et détecter les dialectes en temps réel et à résoudre les ambiguïtés (de quelle porte parlons-nous ?).

souhaitons, des robots de service sophistiqués à la Data dans Star Trek, Marvin dans *Le Guide du Voyageur Galactique* ou encore C3-P0 de Star Wars[6]. Il est évident que la parole est le moyen privilégié d'interaction avec de tels agents (en complément aux interfaces neuronales directes qui ont connu un intérêt grandissant durant les dernières années dû, entre autres, aux développements en électro-encéphalographie [16]. On peut difficilement imaginer devoir utiliser un clavier de quelque sorte pour les commander, voire discuter avec eux.

C'est dans le contexte de cette problématique que nous devons continuer à chercher des moyens de mieux simuler notre capacité à reconnaître et comprendre la parole. Cela dit, de grands progrès ont été réalisés dans les dernières années. Il faut dire que le « *Big Data* » (mégadonnées), jargonnerie du moment, est à la source de bien des développements (p. ex. : traitement de la parole, traitement de l'image, traitement de dossiers médicaux, analyse d'habitude des consommateurs, prévisions boursières, etc.). En effet, la (relative) facilité avec laquelle nous avons dorénavant accès à de vastes quantités de données (sans oublier les ressources computationnelles) ont créé une sorte de ruée vers l'or où les « *data scientist* » (prospecteur des temps modernes) comme on aime bien les appeler, mettent à l'œuvre une panoplie d'algorithmes, autant anciens que nouveaux, dans l'espoir de (re)découvrir des liens jusqu'à présent inconnus (ou oubliés), ou encore d'améliorer les capacités des systèmes en termes de classification. N'oublions pas non plus les célèbres images réimaginées par Google Deep Dream [17] rendues possibles à la fois aux avancées dans l'apprentissage profond qu'au traitement des mégadonnées. Cependant, comme Mikolov [18] l'explique, ce n'est pas nécessairement en « lisant » l'intégralité de l'*Encyclopædia Britannica* qu'un francophone apprendra l'anglais. Comme quoi, on a beau avoir toutes les données du monde, encore faut-il savoir s'en servir pour en extraire l'essentiel[7]. Voilà justement pourquoi nous nous devons de poursuivre nos efforts afin d'améliorer ceux-ci.

Comme vous pouvez l'imaginer, reconnaître, et surtout, comprendre, la parole humaine n'est pas une tâche aisée. De multiples étapes interviennent tout au long du processus, on pense par exemple au :

1. Rehaussement du signal,

[6] Tentons cependant d'éviter HK-47…

[7] Un pessimiste vous dira qu'on a encore du chemin à faire, un optimiste vous dira que c'est une époque parfaite pour débuter un doctorat en sciences de l'information.

2. Segmentation et analyse du signal de parole (traitement du signal et paramétrage),
3. Modélisation du langage (pour essayer de deviner ou, idéalement, « savoir » ce qui a été dit,
4. Analyse sémantique (assurer un maximum de sens),
5. Analyse pragmatique[8]/compréhension du langage (le comprendre).

Malgré tous ces défis, nous ne cessons de construire un futur où les machines seront en mesure de traiter la parole comme nous le pouvons. Il faut reconnaître que nous avons fait d'énormes progrès en termes de reconnaissance automatique de la parole depuis ses débuts en 1952[9] où Davis *et al.* [1] proposèrent un système (circuit analogique), basé sur des filtres, permettant de reconnaître automatiquement une séquence de chiffres. Déjà, 20 ans plus tard, nous étions en mesure d'effectuer de la reconnaissance automatique de la parole en continu (sans marquer de pause après chaque mot) sur un vocabulaire d'environ 100 à 1000 mots. Dès le milieu des années 70 aux années 80 nous passions de modèles principalement par motifs à des modèles stochastiques supportant des milliers de mots [20]. Aujourd'hui, nous disposons de systèmes de reconnaissance automatique de la parole supportant des dizaines, voire des centaines de milliers de mots et où le langage est représenté par un ensemble de modèles statistiques nous permettant de distinguer entre les mots similaires, mais qui ont peu de probabilité de suivre un historique donné.

C'est dans ce contexte que nous nous sommes intéressé à ce domaine et que nous en sommes arrivé à nous questionner sur les manières d'améliorer la prise en compte du sens et de l'importance des liens sémantiques entre les mots. Un élément souvent occulté par les techniques traditionnelles de modélisation du langage est l'objectif du langage en soit : le langage permet aux êtres humains de communiquer ou exprimer des idées, des pensées **qui ont un sens par rapport à un sujet particulier**, donc qui sont sémantiquement et contextuellement liées. De plus, il est possible de dire la même chose de façon plus ou moins précise (généralisation du discours versus spécialisation) et plusieurs sujets similaires peuvent être abordés de façons similaires. Par exemple, on peut avoir un discours similaire à propos d'un chien et d'un chat, par exemple : le chien aboie, le chat miaule, qui peut se résumer à « l'animal de compagnie vocalise ». La question se pose alors

[8] Nous entendons ici par pragmatique le sens de « [la] branche de la linguistique qui s'intéresse aux éléments du langage dont la signification ne peut être comprise qu'en connaissant le contexte de leur emploi » [19]

[9] Ou, du moins, le premier article savant répertorié portant sur le sujet.

à savoir où peut-on trouver ces ressources sémantiques ? Bien qu'il existe plusieurs moyens de déduire des relations sémantiques à l'aide de divers algorithmes, nous privilégierons l'utilisation de ressources plus formelles, soit les ontologies.

L'ontologie est « … l'étude de l'être en tant qu'être… » (Aristote [21]). De ce fait, les ontologies sont utilisées pour structurer les termes et les concepts propres à un (ou plusieurs) domaine de connaissance de façon hiérarchique et relationnelle afin d'ultimement **décrire leur sens** et ainsi nous **permettre de raisonner** à propos de ces objets. Les entités logiques et philosophiques peuvent elles-mêmes être regroupées de façon contextuelle à l'aide d'opérateurs de relation (relations sémantiques, relations de subsomption) nommée, p. ex. : `Terme:chat Relation:manger Terme:souris`.

Puisque les ontologies (informatiques) contiennent une quantité importante d'information et qu'on peut en résumer le rôle comme, étant les « ontologies sont aux données ce que la grammaire est au langage » [22], il devient apparent qu'elles doivent être en mesure de nous permettre d'aider nos modèles de langage à mieux prendre en compte les relations entre les mots et nous donner des modèles qui reflètent notre compréhension du langage.

Cela nous amène ainsi à formuler notre question de recherche principale : comment peut-on utiliser l'information sémantique contenue dans les ontologies pour lisser les modèles de langage ?

L'objectif principal est donc d'exploiter ces ontologies pour introduire une notion de savoir dans les modèles de langage, c'est-à-dire, d'être en mesure d'attribuer des probabilités plus élevées à des *N*-grammes inédits, mais qui sont sémantiquement liées à des *N*-grammes vus ou qui fonctionnent de façon similaire. En d'autres mots nous tenterons à la fois de paraphraser[10] les *N grav*es déjà vus et de générer des *N*-grammes décrivant des sujets similaires. Nous désirons ainsi à la fois réduire la perplexité (une mesure basée sur l'entropie croisée qui nous permet de mesurer la capacité d'un modèle de langage à prédire un corpus de test donné) des modèles de langage et augmenter la performance de la reconnaissance de la parole grâce à nos algorithmes de lissages ontologiques.

[10] paraphrase (n.f.) (grec *paraphrasis*) : Formulation différente d'un énoncé sans altération de son contenu [23].

1.3 Mise en contexte des articles

Le tout a débuté lors de nos travaux de maîtrise où nous avons conçu un navigateur Web pour dispositifs mobiles. Celui-ci utilisait des motifs pour alimenter un système hybride entre le Commande et contrôle et la parole spontanée. C'est en se basant sur ces travaux que nous avons proposé, dans notre *premier article* « Application d'un système basé sur la parole facilitant l'interaction avec des réseaux RFID dans des environnements bruités », un système qui permet à l'utilisateur travaillant dans un environnement très bruyant d'interagir par la parole avec un réseau d'identification par radiofréquence (RFID). Un des problèmes les plus courants attribués aux interfaces opérant sur le modèle de « commande et contrôle » est leur manque de flexibilité par rapport aux énoncés reconnaissables. C'est pourquoi, afin de rendre notre système plus flexible, nous avons proposé un nouveau cadre de dialogue qui donne aux opérateurs humains la capacité de communiquer avec le système d'une façon plus naturelle. Le gestionnaire de dialogue proposé est basé sur le langage AIML (« *Artificial Intelligence Markup Language* ») qui, à la base, a été conçu pour le développement d'agents conversationnels (visant le clavardage). Ce langage permet au système de répondre à des requêtes formulées en langage naturel par l'intermédiaire de détection de motifs (« *pattern matching* »). Cela a aussi comme avantage de rendre le système beaucoup plus robuste par rapport aux erreurs de reconnaissance automatique de la parole, ce qui est particulièrement problématique dans des environnements très bruyants. De plus, afin de faire face à des environnements très bruyants et changeants, une technique en ligne d'amélioration du sous-espace vectoriel du signal basée sur la variance de l'erreur de reconstruction de Karhunen-Loève Transform (ERV-KLT) est proposée.

Finalement, le caractère naturel du dialogue, ainsi que l'efficacité de l'amélioration du signal de la parole sont évalués dans des situations de la vie réelle sous la plateforme Microsoft Windows. Nos résultats démontrent à la fois que l'algorithme ERV-KLT apporte une amélioration de l'évaluation perceptuelle de la qualité de la parole pour les signaux très bruités par rapport aux algorithmes de références et que le système devient effectivement plus robuste face aux erreurs de reconnaissance de la parole par rapport à une grammaire statique.

Bien que le système proposé dans le *premier article* permette une interaction plus naturelle avec le système, il demeure que l'une des difficultés principales rencontrées dans l'élaboration de

systèmes basés sur les commandes est la conception des commandes. Le concepteur doit essayer de couvrir le plus de cas possibles ce qui est à la fois fastidieux et chronophage. Par exemple, une simple requête de recherche d'inventaire nécessite une multitude de motifs pour couvrir tous les lemmes, les conjugaisons et toutes autres variations sémantiques possibles. Ce travail de conception requiert beaucoup de temps et est sujet aux oublis et à l'erreur humaine. C'est pourquoi nous proposons, dans notre *deuxième article*, un système facilitant la conception de ce type de vocabulaire. Comme les ontologies sont utilisées en informatique, pour représenter des sens, des concepts nous nous sommes intéressé à leur utilisation pour rendre le processus de création de commandes le plus semi-supervisé possible. Nous y avons repris la « recette » initiale de dialogue du *premier article.* Le système proposé dans le *deuxième article* devant se charger de complémenter les commandes de base en créant de nouvelles règles par l'exploration d'une ontologie pour trouver d'autres façons de dire la même chose. Nous obtenons alors une plus grande robustesse aux variations de commandes possibles tout en permettant de réduire la charge cognitive des utilisateurs. Finalement, nos résultats démontrent que le système est en mesure de générer de nouveaux motifs à partir des motifs de base tout en minimisant l'impact au niveau de la perplexité du modèle de langage généré.

Bien que le *premier article* et le *deuxième article* proposent un système permettant une interaction plus naturelle avec un système industriel, un problème persiste. Si les modèles de langage ne sont pas en mesure de tenir compte des nouvelles commandes, le système ne sera pas capable de les reconnaître. Nous nous sommes donc tourné vers les modèles de langage qui permettront justement la reconnaissance de telles phrases nouvelles.

Dans notre *troisième article*, nous introduisons une nouvelle méthode de lissage des modèles de langage basée sur les informations sémantiques trouvées dans les ontologies et qui est spécialement adaptée pour la modélisation du langage dans un contexte de ressources limitées. Nous exploitons la connaissance latente de la langue qui est profondément codée dans des ontologies. En tant que tel, cet article examine la possibilité d'utiliser les relations sémantiques et syntaxiques entre les « synsets » (mot+sens) couvertes par l'ontologie WordNet (anglais). Ces propriétés seront alors exploitées afin de générer de nouveaux contextes plausibles pour les événements non vus dans le corpus d'entrainement dans le but de « simuler » un corpus plus vaste. Ces « nouveaux » *N*-grammes sont ensuite interpolés avec un modèle de langue de base lissé à l'aide de l'algorithme

de Witten-Bell [24]. Cela, dans le but de réduire, à la fois, la perplexité du modèle de langage et le taux d'erreur de mots dans une tâche de reconnaissance automatique de la parole. Nos résultats expérimentaux démontrent qu'il est possible d'obtenir une réduction significative de la perplexité du modèle (jusqu'à 9,85 % par rapport au modèle de base). Ces résultats montrent également qu'il est possible de réduire le taux d'erreur de mots du modèle de langage original d'une manière statistiquement significative par rapport au modèle de base lissé à l'aide de l'algorithme de Kneser-Ney entrainé sur le corpus « Wall Street Journal-Based Continuous Speech Recognition Phase II ».

Pour donner suite à ces résultats encourageants, nous nous sommes posé la question à savoir si la théorie des graphes, telle qu'appliquée dans le domaine de la théorie de l'information et des réseaux (sociaux), pouvait apporter de l'information supplémentaire à nos modèles de langage. C'est pourquoi nous avons examiné, dans le *quatrième article*, l'apport des mesures tirées de la théorie des graphes à la généralisation de notre algorithme de lissage de modèle de langage proposé dans le *troisième article*. Plus spécifiquement, nous avons exploré l'effet des mesures HITS, PageRank, Modularité et degré pondéré sur la performance de la reconnaissance automatique de la parole, de même que par rapport à la perplexité des modèles de langage. Comme les ontologies peuvent être représentées sous forme de graphes, nous étudions l'utilisation de paramètres des graphes comme facteur de lissage supplémentaire afin de saisir l'information sémantique ou relationnelle intrinsèque aux ontologies. Plus précisément, nous étudions l'effet des mesures *Hyperlink-Induced Topic Search* (HITS), PageRank, Modularité et degré pondéré à la fois sur la performance de la reconnaissance automatique de la parole, de même que sur la perplexité des modèles de langage. Nos résultats montrent que l'interpolation des bacs originaux à des distances 1, 3 et 5 a donné lieu à une amélioration du taux d'erreur des mots de 0,71 % par rapport à l'interpolation des bacs 1 à 5 proposés ultérieurement dans le *troisième article*. Finalement, nous avons observé que la modularité, le PageRank et l'HITS sont prometteurs et méritent une étude plus approfondie dans le futur.

1.3.1 Au-delà des articles

Les algorithmes d'optimisation par espérance-maximisation ou de Powell sont majoritairement utilisés dans le domaine de la reconnaissance automatique de la parole, notamment pour l'optimisation des paramètres des modèles de langage. Bien qu'assez rapide, nous pouvons

constater empiriquement que les résultats obtenus représentent souvent des solutions très localement optimales.

Ces choix étaient justifiables dans un contexte de ressources computationnelles limitées (ou d'exécution en temps réel). Cependant, comme la plupart des modèles de langage utilisés dans les systèmes commerciaux sont généralement optimisés hors-ligne, la contrainte du temps s'en voit grandement diminuée.

Il nous est donc apparu judicieux d'explorer d'autres algorithmes d'optimisation qui offrent un potentiel accru de trouver une solution globalement satisfaisante. Par conséquent, nous avons choisi d'étudier l'optimisation des poids d'interpolation des modèles de langage par algorithmes évolutionnaires. Plus spécifiquement, nous comparons les algorithmes génétiques et l'algorithme collaboratif d'optimisation par essaims particulaires aux approches traditionnelles. Nos résultats montrent qu'il nous est possible d'obtenir une diminution significative de perplexité pour nos modèles de langage, particulièrement par rapport à l'optimisation par estimation-maximisation. Par ailleurs, nos résultats démontrent qu'une convergence plus rapide que l'algorithme de Powell peut être obtenue en utilisant l'optimisation par algorithmes génétiques et l'optimisation par essaims particulaires.

1.4 Contributions de la thèse

Les contributions principales de notre thèse se résument comme suit :

- Proposition d'un nouvel algorithme de lissage ontologique exploitant l'information sémantique retrouvée dans les ontologies pour le lissage des modèles de langage :
    - Amélioration de la performance de la RAP, en particulier dans un contexte de corpus réduit.
    - Amélioration de la perplexité.
    - Capacité d'adaptation pour les corpora hors contextes :
        - Entrainement sur le corpus *Wall Street Journal*, validation sur le *Billion Word Corpus*.
    - Évaluation des métriques issues de la théorie des graphes pour l'ajustement des poids des *N*-grammes inédits.

    - Utilisation et évaluation d'algorithmes évolutifs pour l'optimisation des poids d'interpolation des modèles de langage.
- Comparaison empirique avec d'autres algorithmes de lissage statistique sur les corpus suivants :
    - Reconnaissance automatique de la parole sur le corpus *Wall Street Journal*.
- Conception d'une suite logicielle permettant de générer les *N*-grammes lissés :
    - Compatible avec le standard C++11.
    - Optimisation de la vitesse de traitement :
        - Traitement parallèle.
        - Algorithmes efficaces.
- Production d'une suite de procédures d'apprentissage pour la reproduction des résultats sous Linux (Python + Bash + Perl + Matlab).

1.5 Organisation du manuscrit

Cette thèse par articles comporte deux grandes sections :

La première section introduit le concept de la modélisation du langage et la met en contexte par rapport à notre parcours et notre démarche scientifique. Un résumé en français de nos algorithmes et modèles, ainsi que les résultats obtenus y sont présentés.

- Le chapitre 2 introduit la modélisation du langage et passe en revue la littérature. Les méthodes de lissage y sont abordées et décrites statistiquement.
- Le chapitre 3 présente de façon succincte, notre démarche ainsi que notre algorithme de lissage basé sur les ontologies.
- Le chapitre 4 présente les améliorations et les évaluations additionnelles que nous avons effectuées suite aux articles présentés dans la deuxième section de cette thèse.

La deuxième section présente les articles publiés ou soumis à des conférences évalués par les pairs.

- Le chapitre 5 présente un système qui permet à l'utilisateur travaillant dans un environnement très bruyant d'interagir par la parole avec un réseau d'identification par radiofréquence (RFID) publié et présenté à ICASSP 2011.

- Le chapitre 6 propose une amélioration au gestionnaire de dialogue proposé au chapitre 5 via la mise en œuvre d'un générateur de motifs basé sur les ontologies. Ce nouveau cadre de dialogue est proposé afin de donner aux opérateurs humains une plus grande flexibilité de communication avec le système. Le système sera dorénavant en mesure de tenir compte des variations possibles de commandes types. Ce travail est publié et présenté à CCECE 2012.
- Dans le chapitre 7, nous introduisons une nouvelle méthode de lissage des modèles de langage basée sur l'information sémantique trouvée dans les ontologies et qui est spécialement adaptée pour la modélisation du langage dans un contexte de ressources limitées. Cette méthode a été publiée et présentée à ICASSP 2014.
- Le chapitre 8 examine l'utilisation de mesures tirées de la théorie des graphes pour améliorer les performances de l'algorithme de lissage de modèle de langage proposé au chapitre 7. Ces travaux ont été publiés et présentés à EUSIPCO 2016.

Finalement, nous concluons notre travail au chapitre 9 en résumant les résultats obtenus et en posant des pistes de recherche futures.

2 MODÉLISATION DU LANGAGE

2.1 Introduction

La plupart des interfaces intégrant l'interaction vocale se répartissent en trois grandes catégories. La première catégorie comprend des interfaces de commande et de contrôle (C & C) qui reposent sur une grammaire étroitement liée à la tâche à accomplir pour permettre à l'utilisateur d'interagir avec le système [10]. Le principal avantage de celle-ci réside dans leur facilité de mise en œuvre et le taux de reconnaissance élevé des commandes, ce qui est largement attribuable à l'espace très restreint de recherche, qu'implique l'utilisation d'un vocabulaire fixe et connu d'avance. Cependant, un de leurs principaux inconvénients est la charge cognitive élevée que ces interfaces induisent lorsque les utilisateurs interagissent avec le système, et ce, en raison de leur manque de flexibilité et du manque de jeux de commandes uniformes. Prenons, par exemple, la reconnaissance vocale utilisée par les fabricants automobiles : un usager qui a l'habitude d'une Ford, utilisant le système SYNC [25], ne sera pas nécessairement en mesure d'utiliser le même vocabulaire dans une voiture Mercedes qui utilise le système LINGUATRONIC [26].

La deuxième catégorie d'interfaces est basée sur les serveurs vocaux interactifs (SVI) qui guident les utilisateurs par le moyen d'invités, afin de faire valider l'énoncé de l'usager à chaque étape [11]. Ce style d'interaction est principalement utilisé dans le cas de la navigation de menus de centres d'appels. Son manque relatif[11] d'efficacité pour une interaction rapide en fait un choix peu adapté à un usage quotidien.

Enfin, les interfaces de la troisième catégorie utilisent le traitement du langage naturel pour analyser la parole de l'utilisateur et pour déterminer l'objectif de sa requête. Cette analyse peut être réalisée par de multiples moyens, tels le traitement automatique du langage naturel (TALN), le traitement et le filtrage par motif [12], ainsi que la sémantique linguistique. Par conséquent, pour être efficace, en raison d'un vocabulaire pratiquement « illimité », ce type d'interface doit avoir recours à une reconnaissance automatique de la parole précise de même qu'à un interpréteur sémantique efficace ou à un gestionnaire de dialogue efficace : la Figure 2.1 présente l'architecture

[11] Par rapport aux interfaces de C&C.

de base d'un tel système de dialogue. La première composante de ce système concerne principalement cette thèse. Il s'agit notamment de la reconnaissance de la parole proprement dite et est illustrée par la Figure 2.2. La reconnaissance de la parole prend comme entrée un signal porteur d'information : l'onde sonore. Le signal est paramétré : découpé en trames (généralement de 10 ms) où l'on considère que le signal est stationnaire pour ensuite en extraire des paramètres que l'on juge pertinents (énergie, formants, etc.). Ce vecteur de paramètres est ce qu'on appelle l'observation O. Cette observation est ensuite comparée à un modèle acoustique pour en déduire la séquence la plus probable de symboles représentatifs (généralement des phonèmes), elle est utilisée de pair avec le modèle du langage pour produire comme sortie une chaîne de symboles (mots) qui sera utilisée comme entrée à la seconde composante du système de dialogue, soit celle de la compréhension de la parole.

Cette étape, telle que définie par la Figure 2.3, permet de déduire le sens, de désambiguïser ou, tout simplement, d'interpréter la chaîne de symboles que le système reçoit comme entrée. Cela est généralement accompli au moyen d'une base de connaissance qui est consultée/utilisée par un ensemble de techniques ; traitement du langage naturel, filtrage par motif, machine à vecteurs de support, etc. pour en arriver à une décomposition arborescente, ou à une commande. Dans ce dernier cas, par exemple, la commande `COMMANDE(de=YQM, a=YUL, date=2018-06-07)` servira d'entrée au gestionnaire de dialogue, illustré par la Figure 2.4, responsable de maintenir les vecteurs suivants ;

- de l'état actuel dans lequel l'utilisateur se trouve ;
- de l'état actuel dans lequel le système se trouve ;
- des actions possibles au moment présent.

La sortie du gestionnaire de dialogue est constituée d'un ensemble de commandes. Par exemple, la commande `CONFIRMER(de=YQM, a=YUL, date=2018-06-07)` servira d'entrée au générateur de réponses. Celui-ci exécutera l'action demandée ou générera un retour d'information pour l'utilisateur (à l'aide d'un générateur automatique de langage naturel dans le cas des interfaces purement vocales).

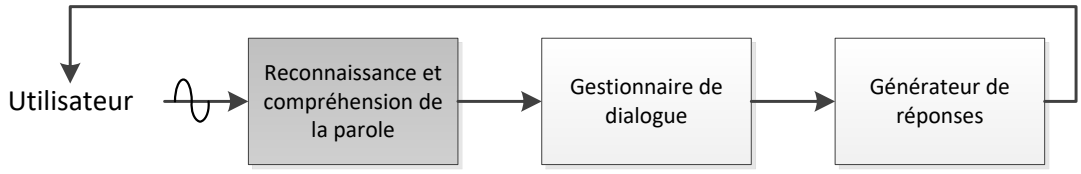


Figure 2.1 : Architecture générale d'un système de dialogue verbal.

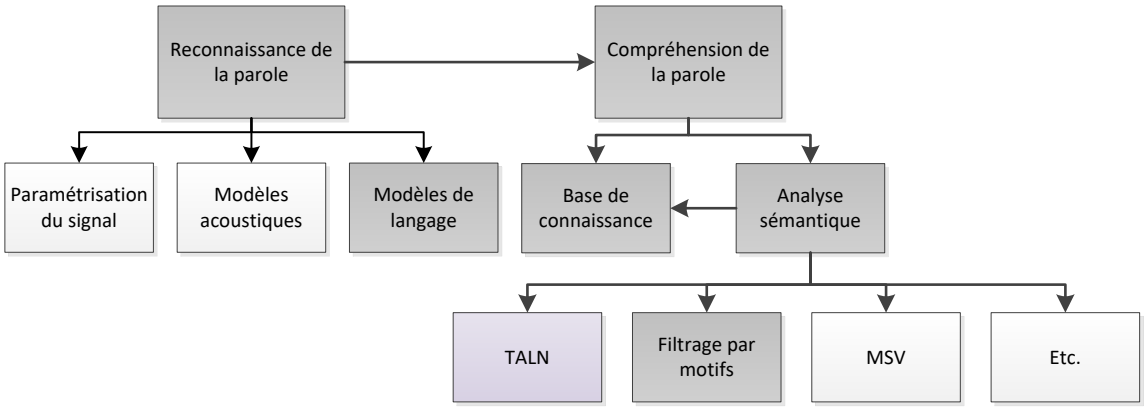


Figure 2.2 : Décomposition des composantes de reconnaissance automatique de la parole et compréhension de la parole dans le contexte d'un système de dialogue verbal.

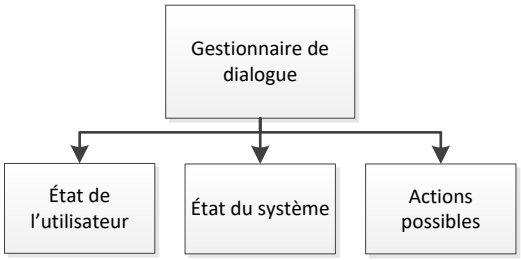


Figure 2.3 : Décomposition de la composante gestionnaire de dialogue dans le contexte d'un système de dialogue verbal.

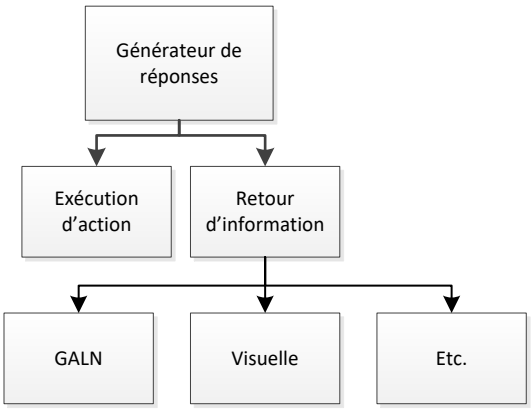


Figure 2.4 : Décomposition de la composante génératrice de réponses dans le contexte d'un système de dialogue verbal.

Dans ce contexte, pourquoi s'intéresser particulièrement à la modélisation du langage ? La réponse est assez simple : une modélisation du langage précise est primordiale pour minimiser la propagation d'erreurs tout au long de la chaîne de traitement. En effet, nous pourrions difficilement imaginer que le module de compréhension de la parole puisse présenter une sortie pertinente au gestionnaire de dialogue si la séquence reçue de la reconnaissance automatique de la parole est truffée d'erreurs. Il est donc facile de voir en quoi ces deux tâches sont intimement liées. Toute amélioration aux modèles de langage ne peut donc qu'avoir un impact positif sur le système dans son ensemble.

Comme nous le verrons dans les sections à venir, le rôle (formel) du modèle de langage est d'attribuer une probabilité à une chaîne W de n caractères donnés :

$$P(W) = P(w_1, w_2, w_3, \dots, w_{n-1}, w_n). \tag{2.1}$$

2.2 Modélisation du langage pour la reconnaissance automatique de la parole

Comme il a déjà été précisé, la **reconnaissance automatique de la parole** (RAP) repose sur un principe fondamental qui consiste à observer une séquence acoustique pour en déduire un ou une séquence de mot(s). Nous pouvons formaliser cela à l'aide d'une simple équation :

$$\widehat{W} = \underset{W \in \mathcal{L}}{\operatorname{argmax}} P(W|O). \tag{2.2}$$

Cette équation se lit de la façon suivante : $\widehat{W}$ le mot reconnu (ou la séquence de mots) est égal à la probabilité maximale d'observation du mot W, étant donné l'observation acoustique O pour un lexique $\mathcal{L}$ donné. Comme l'estimation de $P(W|O)$ est difficile[12], nous pouvons utiliser la règle de Bayes pour la simplifier :

$$P(W|O) = \frac{P(W)P(O|W)}{P(O)}. \tag{2.3}$$

L'objectif de la modélisation du langage est de résoudre l'*a priori*, $P(W)$, de l'équation de la RAP :

$$\widehat{W} = \underset{W \in \mathcal{L}}{\operatorname{argmax}} P(W|O) = \underset{W \in \mathcal{L}}{\operatorname{argmax}} \overbrace{P(W)}^{a\ priori}\ \overbrace{P(O|W)}^{a\ posteriori} \tag{2.4}$$

Où $\widehat{W}$ est le mot (ou la séquence de mots) le plus probablement prononcé, $P(W)$ est l'*a priori*, $P(O|W)$ est la probabilité conditionnelle, ou, *a posteriori*, que l'on ait observé l'événement acoustique O étant donné le mot W[13]. Plus précisément, le modèle de langage nous permet de déterminer la probabilité que le mot W soit prononcé. Notons par ailleurs que le modèle acoustique est généralement modélisé à l'aide de modèles de Markov cachés.

Considérons l'extrait suivant tiré du *Petit Prince* (De Saint-Exupéry [28]). Il est à noter que les symboles `<s>` et `</s>` sont généralement utilisés pour marquer les débuts et fins de phrase dans les corpus utilisés en traitement du langage naturel, le symbole de fin de phrase est particulièrement important pour que le modèle puisse tenir compte de phrases de longueur arbitraire :

[12] Brown [27] illustre bien pourquoi cette estimation est difficile, voire sans intérêt immédiat. Nous sommes moins intéressé (philosophiquement et statistiquement) à connaître toutes les observations acoustiques possibles liées à un mot, une phrase, etc. (essentiellement infinies) qu'à connaître la probabilité d'une observation acoustique particulière étant donné un mot. Par exemple il est évident que la signature acoustique du mot « chat » ne pourrait pas être associée au mot « ristretto », c.-à-d. $P(O|W) = 0$ dans ce cas.

[13] Le lecteur perspicace aura constaté la disparition de $P(O)$ dans l'équation (2.4). En effet, il est possible de l'« éliminer », car cette probabilité revient essentiellement à une constante (d'un point de vue intuitif, on pourrait aussi affirmer que l'on est « certain » d'avoir observé cet événement acoustique $O \rightarrow P(O) = 1$).

<s> s'il vous plaît dessine-moi un mouton </s>

<s> hein </s>

<s> dessine-moi un mouton </s>.

Supposons que l'on cherche à prédire la probabilité que « mouton » soit le prochain mot dans la phrase débutant par les mots suivants : « dessine-moi un… »

Formellement, nous calculons la probabilité d'un mot w étant donné son historique h : $P(w|h)$. Par conséquent, si nous revenons à l'exemple précédent, le travail du modèle de langage est de calculer :

$$P(\texttt{mouton}|\texttt{dessine-moi un}). \tag{2.5}$$

Une méthode simple, permettant de calculer cette probabilité, consiste à diviser le nombre de fois où nous avons observé « dessine-moi un mouton », dans le texte, par le nombre de fois où nous avons observé « dessine-moi un ». Tel que :

$$P(\texttt{mouton}|\texttt{dessine-moi un}) = \frac{C(\texttt{mouton}|\texttt{dessine-moi un})}{C(\texttt{dessine-moi un})}. \tag{2.6}$$

Cette méthode équivaut donc à calculer l'**estimation du maximum de vraisemblance**. Il est alors évident que « mouton » sera le prochain mot avec 100 % de certitude.

Les cas réels ne sont cependant pas si facilement calculables. De plus, l'exemple présenté ci-dessus met en évidence une des lacunes de cette approche : il nous faut un nombre minimal d'occurrences pour pouvoir estimer la vraisemblance d'un événement. Nous obtenons normalement ces occurrences à partir de très larges corpus de texte. Or, cela n'est pas toujours pratique ou fiable, puisque de nouvelles phrases peuvent constamment être construites [29] et qu'**aucun corpus n'existe qui puisse couvrir tout ce qui a déjà été dit ou écrit**.

Par exemple, il serait impossible de prédire avec exactitude une nouvelle phrase ou expression, puisque, par définition, son compte (nombre d'apparitions) est zéro. Par ailleurs, nous pouvons facilement déduire qu'à mesure que la longueur de la phrase augmente et que le vocabulaire croit, cet exercice s'avère peu pratique. C'est pourquoi il nous est nécessaire d'avoir recours à des modèles statistiques, parfois complexes, pour être en mesure de fournir à l'engin de reconnaissance automatique des *a priori* plausibles. C'est notamment le cas des *N*-grammes qui sont utilisés dans

la majorité des applications commerciales et qui servent (presque) toujours de référence aux chercheurs du domaine. Ce sont d'ailleurs ces mêmes modèles qui seront utilisés pour l'ensemble de notre travail et qui seront présentés dans la sous-section suivante.

2.3 Les *N*-grammes

Les *N*-grammes sont des outils versatiles. Elles sont utilisées tant dans le domaine de la reconnaissance automatique de la parole que dans ceux de la traduction automatique, du marquage de parties de discours (*part-of-speech tagging*), de la récapitulation de texte, de la reconnaissance de l'écriture, du traitement du langage, de la recherche d'information, etc.

Comme nous l'avons vu précédemment, l'objectif des modèles de langage dans le contexte de la RAP est de prédire le prochain mot appartenant à une séquence de mots donnée :

$$P(w_n|h) = P(w_1, w_2, \dots, w_{n-1}, w_n), \tag{2.7}$$

où $h = w_1, w_2, \dots, w_{n-1}$. Cela est relativement simple lorsque les séquences sont très courtes. Par contre, cela devient plus difficile à mesure que l'historique croit. Nous pouvons simplifier cette équation en une multiplication de probabilités conditionnelles en utilisant la règle de chaîne :

$$\begin{aligned} P(w_n|h) &= P(w_1)P(w_2|w_1)P(w_3|w_1, w_2) \dots P(w_n|w_1, w_2, \dots, w_{n-1}) \\ &= \prod_{n=1}^{\mathrm{N}} P(w_n|w_1, w_2, \dots, w_{n-1}) \end{aligned} \tag{2.8}$$

Nous pouvons ensuite traduire l'équation (2.8) en estimation du maximum de vraisemblance tel que :

$$P(w_n|h) = \frac{C(w_1)}{\sum_{w_i \in \mathcal{L}} C(w_i)} \cdot \frac{C(w_2|w_1)}{C(w_1)} \cdot \dots \cdot \frac{C(w_n|w_1, w_2, \dots, w_{n-1})}{C(w_1, w_2, \dots, w_{n-1})}. \tag{2.9}$$

où $\sum_{w_i \in \mathcal{L}} C(w_i)$ est la somme de tous les comptes.

Bien que l'équation (2.8) soit (relativement) plus facile à résoudre que l'équation (2.7), il n'en demeure pas moins que, plus la séquence est longue, plus l'exercice est difficile. L'objectif des

modèles de langage basés sur les *N*-grammes est donc de simplifier la prédiction du $N^{\text{ième}}$ mot basé sur les $N-1$ mots précédents.

Retournons à notre exemple précèdent : un modèle bigramme utiliserait seulement le mot précédent pour prédire le prochain mot, un trigramme les deux mots précédents et ainsi de suite. Nous approximons donc la probabilité du $N^{ième}$ mot, w_n, étant donné son historique des $N-1$ mots précédents, $h_{N-1} = w_{n-N+1}^{n-1}$, à l'aide de l'équation

$$\begin{aligned} P(w_n|h) \;\approx\; P(w_n|h_{N-1}) \;&=\; P(w_n|w_{n-N+1}^{n-1}) \\ &=\; \prod_{n=1}^{\mathrm{N}} P(w_n|w_{n-N+1} \ldots w_{n-1}). \end{aligned} \tag{2.10}$$

Ainsi, dans notre exemple, au lieu de calculer $P(\texttt{mouton}|\texttt{s'il vous plaît dessine-moi un})$, nous calculerons $P(\texttt{mouton}|\texttt{un})$ pour un bigramme, $P(\texttt{mouton}|\texttt{dessine-moi un})$ pour une tri-gramme, et ainsi de suite. Nous appelons cette propriété l'**hypothèse de Markov**, « *Markov assumption* », en anglais.

Encore une fois, la façon la plus simple d'estimer la probabilité de ces *N*-grammes est d'avoir recours à l'estimation du maximum de vraisemblance. Nos comptes seront alors normalisés dans le but d'obtenir des (vraies) probabilités se situant entre 0 et 1. L'équation suivante démontre comment on appliquerait la simplification permise par les *N*-grammes dans le contexte d'un bigramme :

$$P(w_n|w_{n-1}) = \frac{C(w_n w_{n-1})}{C(w_{n-1})}. \tag{2.11}$$

Ce qui peut être généralisé pour n'importe quel ordre par

$$P(w_n|h_{N-1}) = \frac{C(w_{n-N+1}^{n})}{\sum_{w_n} C(w_n|w_{n-N+1}^{n-1})} = \frac{C(w_{n-N+1}^{n})}{C(w_{n-N+1}^{n-1})}. \tag{2.12}$$

Néanmoins, un des problèmes inhérents aux *N*-grammes est qu'elles sont sensibles au corpus d'entrainement. Par exemple, un modèle de langage entrainé sur le texte du Petit Prince ne sera certainement pas en mesure de prédire correctement un discours sur l'interdépendance quantique ou encore sur le fonctionnement d'un arbre à cames. Par ailleurs, puisque les écrits et la parole ne

suivent pas les mêmes règles grammaticales et syntaxiques, des *N*-grammes entrainés sur un corpus purement littéraire ne seront pas en mesure de performer adéquatement si on tente de les utiliser dans un contexte de RAP spontanée. Cela nous amène naturellement vers le problème du surajustement dû aux données éparses. En assumant un vocabulaire limité à 10 000 mots, un trigramme idéalement modélisé devra couvrir 10^{12} trigrammes. Des corpus de taille considérables seront alors nécessaires pour entrainer de tels modèles : Wikipédia [30]–[32], BREF [33], One Billion Word Corpus [34], pour n'en nommer que quelques-uns. Admettons maintenant une couverture complète du dictionnaire Larousse (environ 135 000 mots), nous aurons alors ~$2{,}46 \times 10^{15}$ trigrammes possibles, si nous poussons l'exercice et considérons plutôt des 6-grammes, qui sont maintenant monnaie courante dans les systèmes commerciaux, nous en aurons $6{,}053 \times 10^{30}$… Il devient alors évident qu'il est techniquement impossible (pour le moment) de traiter, ou même de stocker une telle quantité d'information. Si l'on considère une moyenne de 19 octets par entrée (18 octets de hachage par 6-gramme + 1 octet par probabilité (quantifiée)), nous pouvons donc estimer qu'environ 115 015 457,67 yottaoctets[14] seront nécessaires pour stocker toutes ces 6-grammes[15]. Même si l'on imagine un algorithme de compression atteignant un taux de compression de 1/1 000 000, la quantité de données (115 Yo) dépasserait largement la capacité mondiale de stockage totale de 44×10^{24} octets prévue pour 2020, et ce, pour le français seulement. C'est pourquoi il est nécessaire d'avoir recours à des **algorithmes de lissage** pour contourner ou minimiser l'effet de ces problèmes. Une autre raison de la nécessité des algorithmes de lissage peut s'illustrer assez facilement à l'aide d'un simple exemple. Admettons, encore une fois, la séquence « `dessine-moi un` », bien que nous ayons déjà observé `mouton` dans notre corpus, il serait tout aussi raisonnable de penser que `chat`, `chien`, `hippopotame` soient tout aussi plausibles comme suite dans la séquence. Cependant, sans les avoir préalablement observés dans notre corpus, nous ne serions pas en mesure de leur attribuer une probabilité autre que 0 sans avoir recours à une autre technique ou un autre type de lissage. Il serait alors impossible pour le système de « retourner » le mot, et ce, même si le signal acoustique est sans équivoque.

Deux approches permettant de pallier les problèmes de pénurie et de dispersion de données sont communément utilisées : le lissage statistique et le lissage sémantique. Le lissage statistique va

[14] 1 yotta = 10^{24} ou un quadrillion (1000 trilliards) selon l'échelle longue (septillion pour les états-unien(ne)s…).

[15] Il est pratique courante dans la littérature de laisser tomber les préfixes à partir du trigramme pour privilégier une notation numérique plus facile à lire (21-gramme est, on l'imagine, plus intuitif qu'icosikaihenagramme…).

essentiellement redistribuer une partie de la masse de probabilité des événements vus aux événements non vus, tandis que le lissage sémantique va plutôt tenter de considérer directement le sens des mots pour maximiser la probabilité des séquences qui « ont un sens ».

2.4 Le lissage statistique

Comme nous faisons souvent face à un problème de pénurie de données avec les *N*-grammes, nous devons trouver un moyen d'en mitiger les effets. La méthode la plus couramment utilisée consiste à modifier les estimations du maximum de vraisemblance, EMV, des *N*-grammes qui n'ont pas (ou peu) été vus. Le lissage est le terme utilisé pour décrire cette redistribution de masse de probabilité. La section qui suit décrit les approches de base ainsi que le lissage Witten-Bell et Kneser-Ney utilisé dans l'ensemble de cette thèse.

2.4.1 Lissage additif

Le lissage additif, ou lissage de Laplace, est la forme la plus simple de lissage et consiste à ajouter une valeur δ (généralement entre 0 et 1) à tous les comptes des *N*-grammes (d'où le nom de « add-one smoothing » lorsque $\delta = 1$) tel que :

$$P_{add}(w_n) = \frac{C(w_n) + \delta}{\sum_{n=1}^{V} C(w_n) + \delta|V|} \tag{2.13}$$

$$P_{add}(w_n|h_{N-1}) = \frac{C(w_n|w_{n-N+1}^{n-1}) + \delta}{C(w_n|w_{n-N+2}^{n-1}) + \delta|V|}. \tag{2.14}$$

Cette méthode de lissage, en particulier le « *add-one smoothing* », ne performe généralement pas de façon satisfaisante et est principalement utilisée comme méthode introductive à la modélisation du langage. Sa principale lacune vient du fait que trop de masse de probabilité est déplacée vers les zéros. Reprenons l'exemple tiré du compte Le Petit Prince tel que résumé dans le Tableau 2.1. Nous constatons que par le simple fait d'ajouter «1» biaise considérablement les proportions statistiques. Cela est particulièrement évident lorsque nous reconstituons les comptes originaux à partir des probabilités lissées avec l'algorithme de Laplace. Nous avons maintenant l'équivalent de trois fois plus d'occurrences de « `un` » dans le corpus que « `s'il` » comparativement à un ratio original de 1 sur 2. Les tableaux 2.1 à 2.6 résument le même scénario, mais cette fois pour les bigrammes.

Tableau 2.1 : Résumé des comptes originaux $C(w_i)$, des comptes ajustés $C_{+1}(w_i)$, des probabilités originales $P(w_i)$, des probabilités ajustées $P_{+1}(w_i)$ et des comptes reconstitués, $C'(w_i)$, à partir de ces dernières.

w_i	$C(w_i)$	$C_{+1}(w_i)$	$P(w_i)$	$P_{+1}(w_i)$	$C'(w_i)$
s'il	15	16	0,000 955 05	0,000 869 707	0,013 045 605
vous	55	56	0,003 501 85	0,003 043 975	0,167 418 601
plaît	3	4	0,000 191 01	0,000 217 427	0,000 652 28
dessine-moi	5	6	0,000 318 35	0,000 326 14	0,001 630 701
un	236	237	0,015 026 11	0,012 882 535	3,040 278 306
mouton	33	34	0,002 101 11	0,001 848 127	0,060 988 205
hein	2	3	0,000 127 34	0,000 163 07	0,000 326 14

Tableau 2.2 : Tableau des comptes du bigramme

	s'il	**vous**	**plaît**	**dessine-moi**	**un**	**mouton**	**hein**
s'il	0	**3**	0	0	0	0	0
vous	**1**	**1**	**2**	0	0	0	0
plaît	0	0	0	**2**	0	0	0
dessine-moi	0	0	0	0	**5**	0	0
un	0	0	0	0	0	**13**	0
mouton	**1**	0	0	0	0	0	0
hein	0	0	0	0	0	0	0

Tableau 2.3 : Probabilités pour tous les bigrammes possibles.

	s'il	**vous**	**plaît**	**dessine-moi**	**un**	**mouton**	**hein**
s'il	0	**0,2**	0	0	0	0	0
vous	**0,018 181 8**	**0,018 181 8**	**0,036 363 6**	0	0	0	0
plaît	0	0	0	**0,666 666 7**	0	0	0
dessine-moi	0	0	0	0	**1**	0	0
un	0	0	0	0	0	**0,055 084 7**	0
mouton	**0,030 303**	0	0	0	0	0	0
hein	0	0	0	0	0	0	0

Tableau 2.4 : Comptes Laplace ajustés pour $\delta = 1$, les comptes qui étaient égaux à 0 apparaissent en gris.

	s'il	vous	plaît	dessine-moi	un	mouton	hein
s'il	1	**4**	1	1	1	1	1
vous	**2**	**2**	**3**	1	1	1	1
plaît	1	1	1	**3**	1	1	1
dessine-moi	1	1	1	1	**6**	1	1
un	1	1	1	1	1	**14**	1
mouton	**2**	1	1	1	1	1	1
hein	1	1	1	1	1	1	1

Tableau 2.5 : Nouvelles probabilités lissées pour $\delta = 1$. Les probabilités pour les bigrammes préexistantes sont en gras.

	s'il	vous	plaît	dessine-moi	un	mouton	hein
s'il	0,000 370	**0,001 478**	0,000 370	0,000 370	0,000 370	0,000 370	0,000 370
vous	**0,000 728**	**0,000 728**	**0,001 092**	0,000 364	0,000 364	0,000 364	0,000 364
plaît	0,000 371	0,000 371	0,000 371	**0,001 114**	0,000 371	0,000 371	0,000 371
dessine-moi	0,000 371	0,000 371	0,000 371	0,000 371	**0,002 226**	0,000 371	0,000 371
un	0,000 342	0,000 342	0,000 342	0,000 342	0,000 342	**0,004 783**	0,000 342
mouton	**0,000 734**	0,000 367	0,000 367	0,000 367	0,000 367	0,000 367	0,000 367
hein	0,000 371	0,000 371	0,000 371	0,000 371	0,000 371	0,000 371	0,000 371

Tableau 2.6 : Comptes reconstitués des bigrammes. Les comptes qui étaient égaux à 0 apparaissent en gris.

	s'il	vous	plaît	dessine-moi	un	mouton	hein
s'il	0,005 543	**0,022 173**	0,005 543	0,005 543	0,005 543	0,005 543	0,005 543
vous	**0,040 058**	**0,040 058**	**0,060 087**	0,020 029	0,020 029	0,020 029	0,020 029
plaît	0,001 114	0,001 114	0,001 114	**0,003 341**	0,001 114	0,001 114	0,001 114
dessine-moi	0,001 855	0,001 855	0,001 855	0,001 855	**0,011 128**	0,001 855	0,001 855
un	0,080 629	0,080 629	0,080 629	0,080 629	0,080 629	**1,128 801**	0,080 629
mouton	**0,024 229**	0,012 115	0,012 115	0,012 115	0,012 115	0,012 115	0,012 115
hein	0,000 743	0,000 743	0,000 743	0,000 743	0,000 743	0,000 743	0,000 743

Cette démonstration, bien que simple, illustre bien le problème inhérent au lissage additif. Comme la majorité des *N*-grammes de niveau supérieur (trigrammes, 4-grammes, etc.) compteront généralement peu d'événements vus par rapport aux non-événements, une quantité disproportionnée de masse statistique sera redistribuée aux comptes égaux à zéro. À titre d'exemple, 75 % de la masse de probabilité pour les bigrammes débutant par « `s'il` » est assignée aux événements qui n'ont pas été vus. Si l'on considère maintenant le bigramme « `dessine-moi un` », on observe que le compte original qui était égal à deux a diminué par un facteur de trois pour passer de 2 à 0,667. C'est pourquoi le lissage par ajout est principalement utilisé comme exercice introductif au lissage des modèles de langage et ainsi permettre aux étudiants de mieux saisir son fonctionnement.

2.4.2 Décompte de Good-Turing

Le décompte de Good-Turing [35], ou **actualisation** Good-Turing, est central à plusieurs autres méthodes de lissage. L'intuition derrière cette méthode de lissage est d'utiliser le compte des événements que l'on a vu **une fois** pour estimer le compte des événements que l'on **n'a jamais vu**. Par ailleurs, Chen et Goodman [36] démontrent que plusieurs de ces méthodes sont des cas spéciaux de l'estimation Good-Turing. De plus, elle est rarement utilisée par elle-même puisqu'elle n'inclut pas les combinaisons de différents ordres (essentiellement les contextes avec, éventuellement, leur distribution unigrammes).

Plus formellement, le lissage Good-Turing utilise le compte N_c des *N*-grammes qui apparaissent c fois pour lisser les comptes qui apparaissent $c-1$ fois. Le vecteur $\boldsymbol{N_c}$ résultant est ce qu'on appelle un vecteur de **fréquence de fréquences**. On aura alors :

$$N_c = \sum_{\{w_n : C(w_n)=c\}} 1. \tag{2.15}$$

Puisque nous allons redistribuer de la masse statistique vers les événements non vus, il est impératif d'ajuster nos comptes originaux afin de nous assurer de toujours avoir des statistiques valides. Nous remplacerons donc les comptes EMV originaux, c, par des comptes c^* **actualisés** selon :

$$c^* = (c+1)\frac{N_{c+1}}{N_c} \tag{2.16}$$

où c est le compte EMV original,

N_c est le nombre d'événements vus « c » fois,

N_{c+1} est le nombre d'événements vus « $c+1$ » fois.

Un des problèmes avec le décompte Good-Turing est que l'équation est indéfinie pour les cas où $N_c = 0$ (division par zéro). Pour remédier à ce problème, plusieurs moyens peuvent être utilisés pour lisser les $N_c = 0$. La plus simple consiste à effectuer une simple régression linéaire de la forme logarithmique, telle que :

$$\log(N_c) = a + b\log(c) \tag{2.17}$$

pour substituer les $N_c = 0$. Cela nous mène donc au calcul des probabilités lissées selon l'algorithme de Good-Turing tel que :

$$P_{GT}(W) = \begin{cases} \dfrac{N_1}{N}, & \text{si } c(W) = 0 \\ \dfrac{c^*}{N}, & \text{si } c(W) > 0 \end{cases}, \tag{2.18}$$

où l'on utilise le ratio de *N*-grammes vus une fois par rapport au nombre total de *N*-grammes vus lorsque le compte du *N*-gramme est de 0, sinon on utilise le compte ajusté c^* divisé par le nombre total N de *N*-grammes vus.

Dans la pratique, nous considérons que les comptes élevés, $c > k$, sont fiables et on évitera généralement de les lisser si possible. Katz [37] suggère une limite $k = 5$, nous définirons donc

$$c^* = c \text{ pour } c > k. \tag{2.19}$$

Par le fait même, nous devons corriger notre équation (2.16) initiale pour prendre en considération cette masse « non lissée » :

$$c^* = \frac{(c+1)\dfrac{N_{c+1}}{N_c} - c\dfrac{(k+1)N_{k+1}}{N_1}}{1 - \dfrac{(k+1)N_{k+1}}{N_1}}, \text{ pour } 1 \leq c \leq k \tag{2.20}$$

Si nous reprenons l'exemple précédent du Petit Prince, et, assumant un vocabulaire de 20 000 mots, nous obtenons les tables des fréquences des fréquences (Tableau 2.7 et 2.9) ainsi que les probabilités (Tableau 2.8 et 2.10) suivantes :

Tableau 2.7 : Fréquence des fréquences des comptes ajustés Good-Turing et des comptes ajustés Katz pour $k = 5$ des unigrammes du texte *Le Petit Prince*.

C	Compte des comptes	C^*	$C^*_{\mathbf{Katz}_{k=5}}$
0	17 309	0,091 3	0,108 604
1	1580	0,526 6	0,436 747
2	416	1,197 1	1,044 761
3	166	2,650 6	2,584 301
4	110	3,636 4	3,567 36
5	80	3,150 0	2,798 946
6	42	6,666 7	6
7	40	4,800 0	7
8	24	7,875 0	8
9	21	9,523 8	9
10	20	8,250 0	10

Tableau 2.8 : Comptes et probabilités d'un échantillon d'unigrammes ajustés selon le lissage Good-Turing et le lissage Katz pour $k = 5$ du texte *Le Petit Prince*.

w_i	$C(w_i)$	$C^*(w_i)$	$P(w_i)$	$P_{GT}(w_i)$	$P_{\mathrm{Katz}_{k=5}}(w_i)$
s'il	15	14,113 416	0,000 955 05	0,000 899	0,000 955 049
vous	55	54,072 391	0,003 501 85	0,003 443	0,003 501 846
plaît	3	2,650 602	0,000 191 01	0,000 169	0,000 168 764
dessine-moi	5	3,150 000	0,000 318 35	0,000 201	0,000 200 56
un	236	235,059 872	0,015 026 11	0,014 966	0,015 026 105
mouton	33	32,083 002	0,002 101 11	0,002 043	0,002 101 108
hein	2	1,197 115	0,000 127 34	0,000 076	$7{,}622\,03 \times 10^{-5}$

Tableau 2.9 : Fréquence des fréquences des comptes ajustés selon le lissage Good-Turing et des comptes ajustés le lissage Katz pour k=5 des unigrammes du texte *Le Petit Prince*. Les comptes originaux sont lissés selon une régression linéaire telle que définie à l'équation (2.17).

C	Compte des comptes (loi de puissance)	C^*	$C^*_{\mathbf{Katz}_{k=5}}$
0	17 309,000 000	0,090 6	0,107 82
1	1568,600 000	0,519 8	0,428 67
2	407,671 076	1,364 0	1,243 26
3	185,348 251	2,286 5	2,151 16
4	105,951 617	3,240 2	3,096 07
5	68,661 695	4,209 4	4,059 41
6	48,171 057	5,187 4	6
7	35,697 814	6,171 0	7
8	27,536 280	7,158 2	8
9	21,901 042	8,147 9	9
10	17,844 822	9,246 4	10

Tableau 2.10 : Comptes et probabilités d'un échantillon d'unigrammes ajustés Good-Turing et Katz pour $k = 5$ du texte *Le Petit Prince*. Les comptes originaux sont lissés selon une régression linéaire telle que définie à l'équation (2.17).

w_i	$C(w_i)$	$\mathbf{C}^*(w_i)$	$P(w_i)$	$P_{\mathbf{GT}}(w_i)$	$P_{\mathbf{GT}_{\mathbf{k=5}}}(w_i)$
s'il	15	13,117 25	0,000 955 05	0,000 835	0,000 955 049
vous	55	54,072 39	0,003 501 85	0,003 443	0,003 501 846
plaît	3	2,286 54	0,000 191 01	0,000 146	0,000 145 584
dessine-moi	5	4,209 43	0,000 318 35	0,000 268	0,000 268 014
un	236	235,059 87	0,015 026 11	0,014 966	0,015 026 105
mouton	33	32,083 00	0,002 101 11	0,002 043	0,002 101 108
hein	2	1,363 95	0,000 127 34	0,000 087	$8{,}684\,26 \times 10^{-5}$

2.4.3 Lissage par interpolation (Jelinek et Mercer)

Bien que les techniques de lissage vues dans les sections précédentes permettent, jusqu'à un certain point, d'améliorer les performances, la généralisation, des modèles de langage, celles-ci ne suffisent pas à elles-mêmes pour obtenir des modèles de langage qui performent bien dans des tâches courantes. C'est pourquoi elles sont habituellement utilisées de pair avec d'autres méthodes

de lissage telles que le lissage par repli « Backoff » ou le lissage par interpolation puisque leur lacune principale est que les *N*-grammes d'ordre supérieurs ($N \geq 2$) ne sont pas en mesure d'exploiter l'information inhérente à la distribution des *N*-grammes d'ordre inférieurs. Prenons l'exemple suivant :

```
dessine-moi un mouton
dessine-moi un xénomorphe
```

Si l'on assume qu'aucun de ces trigrammes n'ait été vu, on devrait normalement, à moins d'être un grand fan de la fiction Alien[16], assigner une plus grande probabilité à l'expression « `dessine-moi un mouton` » que « `dessine-moi un xénomorphe` ». Toutefois, les algorithmes de lissage vus jusqu'à présent ne permettent pas de faire la distinction entre les deux. Pourtant, il serait plus logique d'aller voir la distribution de l'ordre inférieur pour assigner une probabilité aux *N*-grammes jamais vue. C'est l'intuition derrière le lissage par interpolation. La méthode intuitive la plus simple et est l'interpolation linéaire où

$$P_{\text{interp.}}(w_n|w_{n-N+1}^{n-1}) = \sum_{i=1}^{N} \lambda_i \, P_{EMV}(w_n|w_{n-N+i}^{n-1}) \tag{2.21}$$

et où

$$\sum_{i=1}^{N} \lambda_i = 1.$$

À titre d'exemple, supposons disposer du trigramme x, y, z, nous aurons alors :

$$P_{\text{interp.}}(z|x,y) = \lambda_1 P_{EM}(z|x,y) + \lambda_2 P_{EMV}(z|y) + \lambda_3 P_{EM}(z). \tag{2.22}$$

Une version plus élaborée conditionne les probabilités selon des poids λ_i qui dépendent du contexte w_{n-N+1}^{n-1} où

[16] On parlera alors **d'adaptation** du modèle de langage.

$$P_{\text{interp.}}(w_n|w_{n-N+1}^{n-1}) = \sum_{i=1}^{N} \lambda_i(w_{n-N+1}^{n-1}) P_{EMV}(w_n|w_{n-N+i}^{n-1}) \quad \textbf{(2.23)}$$

et où

$$\sum_{i=1}^{N} \lambda_i(w_{n-N+1}^{n-1}) = 1.$$

Afin que la somme des probabilités soit égale à 1 (sinon on ne pourrait évidemment pas parler de statistiques, mais plutôt de pondération[17]). Nous serons donc en mesure d'attribuer plus de poids aux trigrammes ayant un contexte fiable (bigramme ayant un « grand » nombre d'apparitions) et moins de poids aux autres trigrammes ayant un contexte moins fiable. En outre, il nous est possible de reculer jusqu'à l'ordre 0. Une technique courante consiste à assigner une probabilité uniforme telle que

$$P_{unif.}(w_i) = \frac{1}{|V|}. \quad \textbf{(2.24)}$$

Comme on peut le constater par l'équation (2.23), nous aurons idéalement des poids $\lambda_i(w_{n-N+1}^{n-1})$ distincts pour chaque historique distinct. Il est très difficile (pour le moment) de tous les optimiser, en particulier pour les ordres supérieurs, en raison de la croissance exponentielle d'historiques possibles et de la quantité phénoménale de données nécessaire à l'optimisation d'un tel nombre de paramètres (pour ce qui est de la complexité computationnelle, les avancées récentes et à venir en computation quantique risquent de rendre cet obstacle chose du passé). Jelinek et Mercer [39], Brown *et al.* [40], proposent donc de placer les comptes dans un nombre de seaux (« *buckets* ») basé sur leur similarité. L'équation, récursive, qu'ils proposent est la suivante :

$$\begin{aligned} P_{\text{JM}}(w_n|w_{n-N+1}^{n-1}) \;=\; & \lambda(w_{n-N+1}^{n-1}) P_{EMV}(w_n|w_{n-N+1}^{n-1}) \\ & + \Big(1 - \lambda(w_{n-N+1}^{n-1})\Big) P_{JM}(w_n|w_{n-N+2}^{n-1}). \end{aligned} \quad \textbf{(2.25)}$$

[17] Notons que, dans un contexte de modèles de langage très volumineux, la méthode par pondération peut nous donner des résultats plus que satisfaisants [38]

Idéalement ce conditionnement sera effectué sur un corpus dit de développement[18]. Les paramètres λ seront alors optimisés, à l'aide d'un algorithme de notre choix, pour minimiser la **perplexité** (ou tout autre métrique d'intérêt) du modèle par rapport au corpus de développement.

2.4.4 Lissage par repli « backoff »

Comme le lissage par interpolation, le lissage par repli fait appel aux *N*-grammes d'ordre inférieur pour attribuer une probabilité aux événements non vus. Un des algorithmes les plus connus est le repli de Katz [37] : si on rencontre un *N*-gramme ayant un compte de zéro, on l'approxime en se repliant vers sa (N-1)-gramme tel que

$$P_{\text{katz}}(w_n|w_{n-N+1}^{n-1}) = \begin{cases} P^*(w_n|w_{n-N+1}^{n-1}), & \text{si } C(w_n|w_{n-N+1}^{n-1}) > 0 \\ \alpha(w_{n-N+1}^{n-1})\mathrm{P}_{\text{katz}}(w_n|w_{n-N+2}^{n-1}), & \text{sinon} \end{cases} \quad \textbf{(2.26)}$$

où α est un facteur de normalisation nécessaire pour que les probabilités somment à 1,

P^* est la probabilité actualisée.

L'équation (2.26) nous montre que nous utilisons simplement la probabilité actualisée si le *N*-gramme a déjà été vu, sans quoi nous nous replions de façon récursive jusqu'à ce que nous atteignions une soit un (N-1)-gramme déjà vu, ou éventuellement une probabilité unigramme actualisée.

Encore une fois, si l'on reprend l'exemple du trigramme x, y, z, nous aurons alors :

$$P_{\text{katz}}(z|x,y) = \begin{cases} P^*(z|x,y), & \text{si } C(z|x,y) > 0 \\ \alpha(x,y)P_{katz}(y), & \text{sinon si } C(z|y) > 0 \\ P^*(z) & \text{sinon} \end{cases}$$

$$P_{\text{katz}}(z|y) = \begin{cases} P^*(z|y), & \text{si } C(z|y) > 0 \\ \alpha(y)P^*(z), & \text{sinon} \end{cases}. \quad \textbf{(2.27)}$$

Les α sont calculés pour le contexte tel que :

[18] De façon générale, un corpus utilisé pour la modélisation du langage est composé de trois parties : partie entrainement « *training* », comporte la majorité du texte, partie de développement « *dev.* » qui est utilisé strictement pour l'optimisation des paramètres et partie d'évaluation « *test* » qui comme le nom l'indique, est utilisée pour évaluer nos modèles.

$$\alpha(w_{n-N+1}^{n-1}) = \frac{1 - \sum_{\{w_n : C(w_n|w_{n-N+1}^{n-1})>0\}} P^*(w_n|w_{n-N+1}^{n-1})}{1 - \sum_{\{w_n : C(w_n|w_{n-N+1}^{n-1})>0\}} P^*(w_n|w_{n-N+2}^{n-1})} \quad \textbf{(2.28)}$$

ceux-ci représentent un ratio de la masse de probabilité restante au niveau du *N*-gramme par rapport à la masse de probabilité restante au niveau du (N-1)-gramme.

Repli stupide[19] *« Stupid backoff »*

Une technique de repli proposée assez récemment par Brants *et al.*[38] et qui a fait ses preuves est le **repli stupide**. L'intuition derrière cette technique est que puisque la création de modèles de langage volumineux est complexe, que l'on devrait pouvoir se contenter d'utiliser la force brute rendue possible par la disponibilité de corpus de très grande taille. On abandonne donc l'idée de construire des modèles de langage représentant une vraie distribution de probabilité : on se contente de construire un **modèle de scores pondérés**. On peut justifier ce choix en arguant que ce qui importe vraiment dans notre équation (2.4) initiale est que le mot ayant le **score** le plus important soit choisi. L'équation originale proposée ne fait pas appel à des probabilités actualisées et utilise directement les probabilités EMV. Par ailleurs, comme on ne peut plus vraiment parler de probabilités, on notera S le score retourné par l'équation :

$$S(w_n|w_{n-N+1}^{n-1}) = \begin{cases} P(w_n|w_{n-N+1}^{n-1}), & \text{si } C(w_n|w_{n-N+1}^{n-1}) > 0 \\ \alpha S(w_n|w_{n-N+2}^{n-1}), & \text{sinon} \end{cases} \quad \textbf{(2.29)}$$

et où α est le facteur d'ajustement commun à tous les *N*-grammes et où Brants *et al.*[38] ont déterminé de façon empirique qu'une valeur de 0,4 était plus que satisfaisante.

2.4.5 Witten-Bell

L'intuition du lissage Witten-Bell (Witten et Bell [24]) repose sur la supposition que la fréquence avec laquelle un (*N*-1)-gramme constitue un contexte soit proportionnelle à la probabilité estimée qu'elle préfixe un *N*-gramme inédit[20]. Pour illustrer cela, comparons les bigrammes- « `je suis` » et « `je supporte` » rencontrées dans l'œuvre Le Petit Prince, le bigramme- « `je suis` » préfixe beaucoup plus de trigrammes distincts (20) comparativement à seulement une pour « `je`

[19] Les auteurs remarquent que le nom « *Stupid backoff* » avait été donné à la technique parce qu'ils pensaient que quelque chose d'aussi simple (« *stupid* ») ne pourrait jamais fonctionner. Ils avaient tort, mais ont décidé de garder le nom quand même (qui est d'ailleurs bien apprécié par tout scientifique avec, ne serait-ce, qu'un grain d'humour).

[20] On fait souvent référence à cette propriété dans la littérature en employant l'expression **diversité des mots prédits**.

`supporte` ». Il serait raisonnable de supposer que « `je suis` *grand* » est beaucoup plus probable que « `je supporte` *grand* » même si ces deux trigrammes n'ont jamais été vus. Le lissage Witten-Bell modélise donc **la probabilité d'observer un *N*-gramme pour la première fois**. L'équation générale du lissage Witten-Bell (Witten et Bell [24], Chen et Goodman [36]), qui est un cas particulier du lissage Jelinek et Mercer, est définie par :

$$P_{\mathrm{WB}}(w_n|w_{n-N+1}^{n-1}) = \lambda(w_{n-N+1}^{n-1})P(w_n|w_{n-N+1}^{n-1}) + (1-\lambda(w_{n-N+1}^{n-1}))P_{WB}(w_n|w_{n-N+2}^{n-1}) \quad \textbf{(2.30)}$$

Pour calculer les paramètres d'ajustement λ on se base sur le nombre de mots distincts qui suivent un historique w_{n-N+1}^{n-1} tel que :

$$N_{1+}(w_{n-N+1}^{n-1}\,\bullet) = |\{w_n : C(w_n|w_{n-N+1}^{n-1}) > 0\}| \quad \textbf{(2.31)}$$

où $N_{1+}(w_{n-N+1}^{n-1}\,\bullet)$ est le nombre de *N*-grammes vus une ou plusieurs fois et qui débutent par w_{n-N+1}^{n-1},

• est une variable libre (n'importe quel mot ou unigramme) suivant l'historique w_{n-N+1}^{n-1}.

Le reste de l'équation suit une notation mathématique plutôt classique où $|x|$ réfère à la cardinalité d'un ensemble (le nombre d'éléments distincts), $\{y\}$ représente un ensemble quelconque et $\{w_n : C(w_n|w_{n-N+1}^{n-1}) > 0\}$ est l'ensemble de tous les *N*-grammes se terminant par w_n ayant un compte supérieur à zéro. La notation N_{1+} peut quant à elle être généralisée par $N_{k\pm}$ pour signifier toute expression qui apparait k fois, k ou plusieurs fois $(k+)$, ou k ou moins de fois $(k-)$. Nous verrons plus tard que cette notation est utilisée dans d'autres algorithmes de lissage, tel que le lissage Kneser-Ney. Par conséquent, on peut lire $N_{1+}(w_{n-N+1}^{n-1}\,\bullet)$ comme étant le nombre de mots distincts que le (N-1)-gramme w_{n-N+1}^{n-1} précède. Pour calculer les $\lambda_{w_{n-N+1}^{n-1}}$, sensibles au contexte, Witten et Bell proposent :

$$1-\lambda(w_{n-N+1}^{n-1}) = \frac{N_{1+}(w_{n-N+1}^{n-1}\,\bullet)}{N_{1+}(w_{n-N+1}^{n-1}\,\bullet) + C(w_{n-N+1}^{n-1})}. \quad \textbf{(2.32)}$$

En substituant (2.32) dans (2.30) on obtient [36] :

$$P_{\mathrm{WB}}(w_n|w_{n-N+1}^{n-1}) = \frac{C(w_n|w_{n-N+1}^{n-1}) + N_{1+}(w_{n-N+1}^{n-1}\bullet)P_{WB}(w_n|w_{n-N+2}^{n-1})}{N_{1+}(w_{n-N+1}^{n-1}\bullet) + C(w_{n-N+1}^{n-1})}. \quad \textbf{(2.33)}$$

Le lissage Witten-Bell (selon (2.30)) accordera ainsi plus de poids à un *N*-gramme déjà vu et, sinon, sera en mesure d'estimer la probabilité de rencontrer une nouvelle *N*-gramme.

2.4.6 Décompte absolu (Absolute discounting)

Le décompte absolu (Ney, Essen, and Kneser [41]) est une variante du lissage par interpolation. Au lieu de multiplier la probabilité d'ordre supérieur (le *N*-gramme) par un facteur λ, on va, à l'inverse du lissage additif, soustraire une constante $0 \leq D \leq 1$ aux *N*-grammes vus une fois ou plus pour ensuite redistribuer cette masse aux *N*-grammes jamais vus :

$$P_{abs}(w_n|w_{n-N+1}^{n-1}) = \frac{\max\{C(w_n|w_{n-N+1}^{n-1}) - D, 0\}}{C(w_{n-N+1}^{n-1})} + \left(1 - \lambda_{w_{n-N+1}^{n-1}}\right)P_{abs}(w_n|w_{n-N+2}^{n-1}). \quad \textbf{(2.34)}$$

Pour que la distribution de probabilité se somme à un, nous définissons

$$1 - \lambda_{w_{n-N+1}^{n-1}} = \frac{D}{C(w_{n-N+1}^{n-1})} N_{1+}(w_{n-N+1}^{n-1}\bullet). \quad \textbf{(2.35)}$$

Ney, Essen et Kneser [41] suggèrent une valeur de D :

$$D = \frac{N_1}{N_1 + 2N_2}. \quad \textbf{(2.36)}$$

Où N_1 et N_2 respectent la notation vue précédemment (N_1 est le nombre de *N*-grammes vus exactement une fois et N_2 le nombre de *N*-grammes vus exactement deux fois). Church et Gale montrent empiriquement dans [42] comment les comptes supérieurs à 3 ont essentiellement un décompte Good-Turing relativement constant. Cette démonstration inspirera d'ailleurs Kneser et Ney à proposer le lissage Kneser-Ney dans [43].

2.4.7 Kneser-Ney

Le lissage Kneser-Ney (Kneser et [43]) est, tel que mentionné précédemment, basé sur le décompte absolu. À l'inverse du lissage Witten-Bell qui tente de modéliser la diversité des mots prédits, le lissage Kneser-Ney va plutôt tenter de modéliser la **diversité des historiques**. C'est-à-dire qu'à l'inverse de Witten-Bell, qui se résume à considérer le nombre de *N*-grammes différents constitués par l'historique h_{N-1}, le lissage Kneser-Ney va plutôt considérer le nombre d'historiques dans

lesquels le mot w_n apparait. On parle alors ici de **probabilité de continuation** : c'est-à-dire la probabilité qu'un mot complète un nouvel historique. Admettons que l'on doit reculer à un unigramme pour compléter la phrase suivante :

`<s> j'ai le _____ que toi </s>`

Le mot « `même` » semblerait à première vue beaucoup plus probable que « `prince` » pour compléter cette phrase. Toutefois, si on se fie aux comptes du texte Le Petit Prince, le mot « `prince` » est beaucoup plus fréquent (172 occurrences) que « `même` » (24 occurrences) et donc, un unigramme choisira plutôt « `prince` » pour compléter la phrase. Le lissage Kneser-Ney exploite l'intuition que même si « `prince` » est fréquent, il l'est seulement après le mot « `petit` », c'est-à-dire dans le bigramme « `petit prince` ». Le mot « même » quant à lui complète beaucoup plus de bigrammes différents.

C'est sur cette intuition de continuité proposée par Kneser et [43], mais présenté de façon beaucoup plus claire par Chen et Goodman [36] que s'appuie l'estimation Kneser-Ney où :

$$P_{KN}(w_n|w_{n-N+1}^{n-1}) = \frac{\max\{C(w_n|w_{n-N+1}^{n-1}) - D, 0\}}{C(w_{n-N+1}^{n-1})} + \frac{D \cdot N_{1+}(w_{n-N+1}^{n-1}\,\bullet)}{C(w_{n-N+1}^{n-1})} P_{KN}(w_n|w_{n-N+2}^{n-1}).$$

(2.37)

Pour trouver la distribution unigramme, on a :

$$P_{KN}(w_n) = \frac{N_{1+}(\bullet\, w_n)}{N_{1+}(\bullet\bullet)}$$

(2.38)

où $N_{1+}(\bullet\, w_n) = |\{w_{n-1} : C(w_n|w_{n-1}) > 0\}|$ est le nombre bigrammes vus que le mot w_n complète,

$N_{1+}(\bullet\bullet) = \sum_{w_n} N_{1+}(\bullet\, w_n)$ est le nombre total de **types** [21]de bigrammes vus.

En généralisant (2.37) pour les *N*-grammes de plus grand ordre, on obtient :

[21] Le terme type fait ici référence aux uniques, par exemple la phrase « `un mouton dans un champ` » compte cinq mots, mais seulement quatre types (`un, mouton, dans, champ`).

$$P_{KN}(w_n|w_{n-N+2}^{n-1}) = \frac{N_{1+}(\bullet\, w_{n-N+2}^{n})}{N_{1+}(\bullet\, w_{n-N+2}^{n-1}\, \bullet)} \tag{2.39}$$

où

$$N_{1+}(\bullet\, w_{n-N+2}^{n}) = |\{w_{n-\mathrm{N}+1} : C(w_n|w_{n-N+2}^{n-1}, w_{n-N+1}) > 0\}| \tag{2.40}$$

et

$$\begin{aligned} N_{1+}(\bullet\, w_{n-N+2}^{n-1}\, \bullet) &= |\{(w_n, w_{n-N+1}) : C(w_n|w_{n-N+2}^{n-1}, w_{n-N+1}) > 0\}| \\ &= \sum_{w_n} N_{1+}(\bullet\, w_{n-N+2}^{n}) \end{aligned} \tag{2.41}$$

où $N_{1+}(\bullet\, w_{n-N+2}^{n})$ est le nombre de *N*-grammes que le (N-1)-gramme w_{n-N+2}^{n} complète.

$N_{1+}(\bullet\, w_{n-N+2}^{n-1}\, \bullet)$ est le nombre total de types de *N*-grammes différents vus avec w_{n-N+2}^{n-1}[22].

Par exemple, pour le 4-gramme s, x, y, z nous aurions

$$N_{1+}(\bullet\, x, y, z) = |\{s : C(s, x, y, z) > 0\}|$$

$$N_{1+}(\bullet\, x, y\, \bullet) = |\{(s, z) : C(s, x, y, z) > 0\}| = \sum_{z} N_{1+}(\bullet\, x, y, z)$$

2.4.8 Kneser-Ney modifié (Modified Kneser-Ney)

Le lissage Kneser-Ney modifié (Chen et Goodman [44]) est l'une des versions les plus performantes du lissage Kneser-Ney. Au lieu d'utiliser un seul facteur de décompte, trois (4 si on compte $D_0 = 0$) différentes valeurs sont utilisées, soit D_1, D_2 et D_{3+} et respectivement appliqués aux *N*-grammes avec un compte d'un, deux et trois ou plus, respectivement :

[22] Cette équation n'est que très rarement abordée formellement d'un point de vue des ordres supérieurs dans la littérature. Ceci nous laisse donc deux interprétations possibles, soit le nombre total de types de *N*-grammes vus, ou le nombre total de types de *N*-grammes ayant comme cœur $w_2, w_3, \ldots, w_{n-1}$. D'ailleurs Chen et Goodman [44] simplifient à $P_{KN}(w_n|w_{n-N+1}^{n-1}) = \frac{\max\{N_{1+}(w_{n-N+1}^{n}) - D, 0\}}{\sum_{w_n} N_{1+}(w_{n-N+1}^{n})} + \frac{D \cdot N_{1+}(w_{n-N+1}^{n-1}\bullet)}{\sum_{w_n} N_{1+}(w_{n-N+1}^{n})} P_{KN}(w_n|w_{n-N+2}^{n-1})$ pour les ordres inférieurs.

$$D = \begin{cases} 0 & \text{si } C = 0 \\ D_1 & \text{si } C = 1 \\ D_2 & \text{si } C = 2 \\ D_{3+} & \text{si } C \geq 3 \end{cases} \tag{2.42}$$

Si on modifie (2.37) pour tenir compte de cette modification, on obtient :

$$P_{MKN}(w_n|w_{n-N+1}^{n-1}) = \frac{C(w_n|w_{n-N+1}^{n-1}) - D\big(C(w_n|w_{n-N+1}^{n-1})\big)}{C(w_{n-N+1}^{n-1})} + \gamma(w_{n-N+1}^{n-1}) P_{MKN}(w_n|w_{n-N+2}^{n-1}). \tag{2.43}$$

Pour que le tout somme à 1, on définit

$$\gamma(w_{n-N+1}^{n-1}) = \frac{D_1 N_1(w_{n-N+1}^{n-1}\,\bullet) + D_2 N_2(w_{n-N+1}^{n-1}\,\bullet) + D_{3+} N_{3+}(w_{n-N+1}^{n-1}\,\bullet)}{C(w_{n-N+1}^{n-1})} \tag{2.44}$$

et où on définit généralement (pour la simplicité et parce que ça fonctionne bien) :

$$\begin{aligned} Y &= \frac{N_1}{N_1 + 2N_2} \\ D_1 &= 1 - 2Y\frac{N_2}{N_1} \\ D_2 &= 2 - 3Y\frac{N_3}{N_2} \\ D_{3+} &= 3 - 4Y\frac{N_4}{N_3}. \end{aligned} \tag{2.45}$$

Cela est cohérent avec l'observation de Church et Gale [42] que les comptes supérieurs à 3 ont essentiellement un décompte Good-Turing relativement constant.

2.4.9 Basé sur les classes

Le lissage des *N*-grammes basé sur les classes repose sur l'intuition que des mots appartenant à la même classe auront un comportement similaire. Un exemple souvent utilisé est celui d'un système de réservation de vols commerciaux aériens. On veut modéliser le bigramme « `vers Bathurst` », mais on a seulement rencontré « `vers Toronto` », « `vers Montréal` », « `vers Moncton` » dans notre corpus d'entrainement. Si on sait que « `bathurst` » est une ville, on pourra alors prédire la probabilité qu'il apparaisse après « `vers` ». L'algorithme de lissage par classe le plus simple est le regroupement IBM (*IBM clustering*) [40] et est généralisé par Uszkoreit et Brants dans [45] :

$$P(w_n|w_{n-N+1}^{n-1}) \approx P\big(w_n\big|c(w_n)\big) \times P(c(w_n)|w_{n-N+1}^{n-1}) \tag{2.46}$$

où $P(w_n|c(w_n)) = \frac{C(w_n)}{C(c_n)}$ la fréquence relative du mot par rapport à la classe,

$P(c(w_n)|w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n-1}c(w_n))}{C(w_{n-N+1}^{n-1})}$ la probabilité que la classe c_n suive la séquence de mots w_{n-N+1}^{n-1}.

Les classes pourront être définies de façon manuelle, ou automatique selon les besoins. Par ailleurs, les mots peuvent appartenir à plusieurs classes dans le cas du regroupement « soft », ou à une seule classe dans le cas du regroupement « hard ». Finalement, ces *N*-grammes sont habituellement interpolés avec des *N*-grammes conventionnels afin d'obtenir des performances adéquates.

2.5 Un résumé d'autres techniques récentes de lissage

Bien que cette thèse se concentre sur le lissage des *N*-grammes, cette section présente d'autres méthodes de modélisation et de lissage du langage que nous n'avons pas pu couvrir dans le cadre de cette thèse, mais que nous jugions tout de même important de mentionner étant donné leur adoption grandissante dans le domaine de la modélisation du langage.

2.5.1 Lissage bayésien

Bien que l'approche bayésienne ait été largement adoptée par la communauté en apprentissage machine, il demeure que son application dans le domaine du traitement de la parole est relativement limitée [46]. Une des approches bayésiennes ayant connu un fort succès est la modélisation par processus Pitman-Yor (hiérarchiques) [47]–[49]. Les modèles basés sur le processus PY sont illustrés par l'équation suivante :

$$G \sim PY(d, \alpha_0, G_0). \tag{2.47}$$

où $G = [G(w)]_{w\in\mathcal{L}}$ est le vecteur des probabilités unigrammes $G(w)$ des mots du vocabulaire $\mathcal{L}$.

$G_0 = [G_0(w)]_{w \in \mathcal{L}}$ est une distribution de base pour les w et où on assume généralement que $G_0 = 1/|\mathcal{L}|$ pour tout $w \in \mathcal{L}$ dans la pratique.

$0 \leq d \leq 1$ est un paramètre de décompte.

α_0 est un paramètre d'influence et est utilisé avec d pour contrôler le degré de variabilité autour de G_0.

La métaphore la plus souvent utilisée pour expliquer ce processus est celle du restaurant chinois [50]. Cette métaphore définit la probabilité qu'un client entrant au restaurant s'assoie à la table k est proportionnel au nombre de clients déjà assis à cette table $c_k - d$, ou à une nouvelle table de façon proportionnelle à $\alpha_0 + d_m$, et, où $m.$ est le nombre de tables déjà occupées. Nous obtenons alors :

$$p(z_i = k | \boldsymbol{z}^{-i}, d, \alpha_0) = \begin{cases} \dfrac{c_k - d}{\alpha_0 + c.} & 1 \leq k \leq m. \\ \dfrac{\alpha_0 + d_m}{\alpha_0 + c.} & k = m.+1 \end{cases} . \tag{2.48}$$

La probabilité que le $i$ème client choisit la table, indexée par z_i, k étant donné la disposition des sièges choisie par les $i-1$ clients précédents et où $\boldsymbol{z}^{-i} = \{z_1, ..., z_{i-1}\}$ et $c.$ est le nombre total de clients. Cela nous permettra par la suite de définir le lissage Pitman-Yor hiérarchique, tel que :

$$\begin{aligned} P_{PYH}(w_n | h_{N-1}, S, d_{|h_{N-1}|}, \alpha_{|h_{N-1}|}) &= \frac{C(w_{n-N+1}^n) - d_{|h_{N-1}|} N_{1+}(\bullet\, w_n)}{\alpha_{|h_{N-1}|} + C(w_{n-N+1}^{n-1})} \\ &+ \frac{\alpha_{|h_{N-1}|} + d_{|h_{N-1}|} N_{1+}}{\alpha_{|h_{N-1}|} + C(w_{n-N+1}^{n-1})} \\ &\times P_{PYH}(w_n | h_{N-2}, S, d_{|h_{N-2}|}, \alpha_{|h_{N-2}|}) \end{aligned} , \tag{2.49}$$

où les inférences du modèle, S, font appel à l'échantillonnage de Gibbs ;

$0 \leq d_{|h_{N-1}|} \leq 1$ est un paramètre de décompte propre au contexte h_{N-1}, (ou à sa longueur);

$\alpha_{|h_{N-1}|}$ est un paramètre d'influence propre au contexte h_{N-1} (ou à sa longueur) et est utilisé avec $d_{|h_{N-1}|}$ pour contrôler le degré de variabilité autour de G_0, la distribution unigramme globale moyenne.

Par ailleurs, Teh démontre dans [47] comment le lissage Kneser-Ney interpolé peut être généralisé par un modèle de langue Pitman-Yor hiérarchique. Une autre des propriétés intéressantes du processus Pitman-Yor est qu'il produit une loi de puissance qui est similaire à ce que l'on retrouve dans le langage naturel [47] (voir la loi de Zipf–Mandelbrot [51], [52]).

La difficulté principale qui a fait en sorte que ces techniques bayésiennes ne soient pas adoptées à plus grande échelle est due à leur complexité computationnelle (nombre de paramètres très élevés, entre autres). Cependant, comme pour le cas des réseaux de neurones, les avancés en termes de puissances de calcul, autant du point de vue des unités centrales multicœurs et des processeurs graphiques (*Graphical Processing Unit [GPU]*) font en sorte que nous nous attendons à l'arrivée de l'application de techniques complètement bayésiennes pour le traitement de la parole et, notamment, pour la modélisation du langage.

Allocation de Dirichlet latente

Le lissage, ou l'adaptation des modèles de langage basés sur la sémantique est un domaine de recherche qui s'appuie sur la recherche effectuée dans la sphère de la recherche d'information/indexation des données. En effet, les moteurs de recherche, par exemple, ont pour but de lier des mots, des expressions, à des documents pour retourner les plus pertinents à l'utilisateur. Ces mêmes principes peuvent être exploités pour nous permettre de faire ressortir les sujets traités dans un discours, et du même coup, nous permettre de donner une sorte de mémoire à long terme à nos modèles de langage qui pourront alors privilégier les termes importants au sujet ou aux sujets en cours.

L'allocation de Dirichlet latente [53] (ADL) est un modèle hiérarchique bayésien à trois niveaux basé à la fois sur les mots qui sont modélisés comme une mixture finie sur un ensemble sous-jacent de sujets et la mixture de ces sujets [23](*topic*) propres à l'ensemble des documents (vus, et éventuellement jamais vus) d'une collection donnée. C'est donc un modèle probabiliste **génératif** de corpus qui nous permet de déterminer le sujet d'un document basé sur sa distribution de sujets qui sont eux-mêmes définis par une distribution de mots telle que :

[23] Notons que par sujet nous entendons sujet d'un point de vue majoritairement statistique et pas nécessairement du point de vue de la fonction grammaticale qui donne au verbe son nombre et son genre.

$$P(d_l, t_l, \theta_{d_l}|\alpha, \beta) = P(\theta_{d_l}|\alpha) \prod_{i=1}^{N} P(t_{i,l}|\theta_{d_l}) P(w_{i,l}|t_{i,l}, \beta), \tag{2.50}$$

où w est un mot, $d_l = [w_{1,l}, w_{2,l}, \dots, w_{N,l}]$ est un document et est composé d'une séquence de mots où $w_{n,l}$ est le $n$ième mot dans la séquence du l^e document et $t_l = [t_{1,l}, t_{2,l}, \dots, t_{N,l}]$ est l'ensemble de sujets assignés à chaque mot $w_{n,l}$ du document d_l et où $t_{i,l} \in \{t_1, t_2, \dots, t_K\}$ est tiré de l'ensemble des K sujets définissant le corpus. d_l est alors un document appartenant à un corpus de M documents tel que $D = \{d_1, d_2, \dots, d_M\}$. Les paramètres α et β sont des paramètres au niveau du corpus et sont échantillonnés une seule fois lors de « l'entrainement » pour maximiser la densité marginale du « document » d'entrainement. Les paramètres $\alpha = \{\alpha_1, \dots, \alpha_K\}$ représentent les paramètres Dirichlet pour les K sujets latents au corpus et $\beta_{t_k, w_n} = P(w_n|t_k)$ est le paramètre Dirichlet propre aux mots. Finalement, θ_{d_l} est un vecteur de variables propres à chaque document d_l et représente sa distribution de sujets.

En généralisant pour les *N*-grammes, nous obtenons un modèle de langage ADL basé sur l'historique [54] :

$$P_{ADL}(w_n|h_{N-1}, \alpha, \beta) = \sum_{k=1}^{K} \beta_{t_k, w_n} \frac{g(h_{N-1}, \alpha_k)}{\sum_{j=1}^{K} g(h_{N-1}, \alpha_j)}, \tag{2.51}$$

où l'historique h_{N-1} est transformé en un espace de sujets à l'aide d'une fonction $g(\cdot)$ qui prend le paramètre Dirichlet α comme entrée pour trouver les hyperparamètres Dirichlet *a priori* dépendants des sujets qui sont utilisés pour générer le mot w_n prédit.

2.5.2 Modèle de langage basé sur les réseaux de neurones

Finalement, aucune thèse contemporaine portant sur la modélisation du langage ne pourrait exister sans, au moins, mentionner les avancées récentes, principalement à partir de 2003 [55] et plus récemment [18], [56]–[62] dans les modèles de langage basés sur les réseaux de neurones récurrents (RNR). Un de leurs principaux avantages est leur capacité de regroupement sémantique automatique des mots, de la réduction de la dimensionnalité[24] et de leur capacité à la généralisation. Par ailleurs, un de leurs avantages distincts par rapport aux *N*-grammes conventionnels est la

[24] Un antidote, en quelque sorte, contre le fléau de la dimensionnalité (« *curse of dimensionality* ») telle que décrite par Richard E. Bellman dans [63], [64]

capacité des RNR à tenir compte des dépendances lointaines et donc de contourner les limitations inhérentes à l'hypothèse de Markov propre aux *N*-grammes : nous ne sommes plus limités aux $N-1$ derniers mots. Cela est particulièrement vrai pour les RNR implémentant une longue mémoire à court terme « *Long Short Term Memory* » (LSTM) originalement proposé par Hochreiter et Schmidhuber [65] et ensuite amélioré par Gers dans [66] où il propose un mécanisme de commande synaptique permettant aux neurones d'oublier l'information considérée comme désuète. Sa première implémentation connue dans le domaine de la modélisation du langage fut proposée par Sundermeyer *et al.* dans [67] et appliqué et étendu dans un contexte de modélisation du langage de grande échelle dans [62] par Jozefowicz *et al*, Merity *et al.* [68] en régularisant et optimisant d'énormes modèles ou encore Shazeer *et al.* qui ajoutent une mixture d'experts (Mixture-of-Experts) entre chaque couche LSTM [69] afin de réduire encore plus la perplexité de leurs modèles de langage.

Une façon simple de représenter un modèle de langage par RNR LSTM est illustrée à la Figure 2.5. Chaque mot du vocabulaire est encodé dans un vecteur $\boldsymbol{w}(n)$ selon un codage 1-de-V, où V est le nombre de mots dans le vocabulaire (essentiellement un vecteur binaire où seul l'index du mot w_v est égal à 1). Les vecteurs $\boldsymbol{s}(\cdot)$ représentent l'état actuel et passé du réseau. À la sortie, $\boldsymbol{y}(n)$, on emploie une fonction d'activation « *softmax* » (fonction exponentielle normalisée : une normalisation des fonctions logistiques) afin de s'assurer d'avoir de vraies probabilités. On entrainera le réseau à l'aide d'algorithmes de rétropropagation du gradient pour finalement obtenir une sortie telle que :

$$\boldsymbol{y}(n) \to P\big(w_{n+1}\big|w_n, \boldsymbol{s}(n-1)\big). \tag{2.52}$$

et où

$$s_j(n) = f\left(\sum_i x_i(n)u_{ji}\right) \tag{2.53}$$

$$y_k(n) = g\left(\sum_j s_j(n)v_{kj}\right) \tag{2.54}$$

où $f(z)$ est la fonction d'activation (de neurone) sigmoïde :

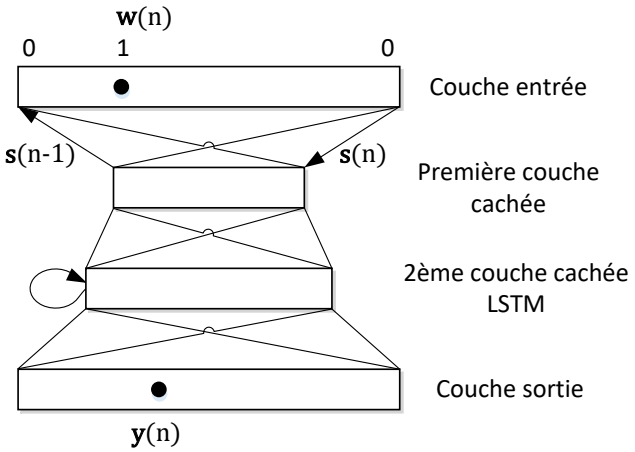


Figure 2.5 : Architecture d'un ML basé sur les réseaux de neurones récurrents avec LSTM.

$$f(z) = \frac{1}{1 + e^{-z}} \tag{2.55}$$

et $g(z)$ est la fonction *softmax* (fonction exponentielle normalisée) :

$$g(z_w) = \frac{e^{z_w}}{\sum_k e^{z_k}}. \text{ pour tout } w \in \{1, \ldots, K\} \tag{2.56}$$

où $\sum g(z_w) = 1$

Afin de réduire la complexité de calcul d'un très grand vocabulaire, il est possible d'emprunter l'intuition des modèles basés sur les classes (disjointes) afin de prédire $P(w_n|w_1^{n-1})$:

$$P(w_n|w_1^{n-1}) \approx P\Big(w_n\Big|w_1^{n-1}, c(w_n)\Big) \times P(c(w_n)|w_1^{n-1}) \tag{2.57}$$

La tendance actuelle en termes de classification à l'aide des réseaux de neurones se concentre principalement à l'application du « *Deep Learning* », ou apprentissage profond, popularisé par Krizhevsky *et al.* dans [70] et qui consiste tout simplement en un réseau de neurones avec une hiérarchie de plusieurs couches cachées (réseau de neurones profond (RNP)). Chaque couche du réseau sert essentiellement à détecter des motifs de plus en plus complexes et est ce qui en fait leur force principale.

Notons d'ailleurs que bien que tout réseau de neurones à multiples couches cachées puisse être représenté par un réseau de neurones équivalent à une couche cachée, Montúfar et Morton [71] démontrent comment la profondeur permet à un RNP de représenter une machine Boltzmann restreinte ayant un nombre de paramètres exponentiellement plus grand que le nombre de paramètres requis par le RNP. Montúfar *et al.* [72] proposent que chaque couche divise l'espace de paramètres en un nombre exponentiel d'ensembles, ce qu'un réseau à une seule couche cachée ne peut accomplir.

Un des problèmes pratiques de l'implémentation de l'apprentissage profond découle de la complexité de configuration des paramètres du modèle initial (l'architecture du modèle). Zoph et Le proposent ainsi une stratégie d'optimisation d'hyperparamètres, le *Neural Architecture Search* [73]. Ils basent leur travail sur l'observation que la structure et la connectivité d'un réseau de neurones peuvent être spécifiées par une chaîne de caractères de longueur variable. Zoph et Le proposent donc d'utiliser un RNR pour déterminer les paramètres initiaux optimaux de l'architecture (représenté par une chaîne de caractères) qui servira ensuite à générer un réseau neuronal enfant qui sera entrainé de façon conventionnelle. Leur expérimentation empirique démontre que cette stratégie permet d'automatiquement générer des architectures performantes ce qui devrait permettre aux chercheurs de plus rapidement évaluer différents modèles de réseaux de neurones en évitant de devoir passer énormément de temps à optimiser les paramètres initiaux de ceux-ci.

Bien que la majorité des modèles présentés jusqu'à présent dans le contexte de l'aprentissage profond reposent des réseaux de neurones récurrents, plus précisément sur les LSTM en traitement de la parole, Benes *et al.* [74] proposent un modèle basé sur un modèle de réseau de neurones à propagation avant (*feedforward network*), soit le *Residual Memory Networks* et où les couches sont réparties dans le temps avec des lien résiduel et des connexions à temporisation à toutes les $t-3$ couches qui permettent au réseau de tenir compte de contextes temporels longs. Un de ses avantages est qu'il peut permettre une réduction substantielle de la taille des modèles comparativement à un réseau LSTM équivalent tout en maintenant une performance similaire sur une tâche de modélisation du langage basée sur le PennTreeBank [74].

Finalement, bien que les réseaux de neurones font maintenant partie intégrante d'une grande majorité de la recherche dans le domaine de la modélisation du langage, il ne faut toutefois pas

croire que les modèles statistiques, telles que les N-grammes, devraient être relégués aux oubliettes. En effet, Jozefowickz et al. dans [62] soulignent l'importance des N-grammes dans les modèles de langage de très grande taille :

> *« Despite the fact that simpler models, such as N-grams, only use a short history of previous words to predict the next word, they **are still a key component** to high quality, low perplexity LMs. Indeed, most recent work on large scale LM has shown that RNNs are great in combination with N-grams, as they may have different strengths that complement N-gram models, but worse when considered in isolation »*
> *(Jozefowicz et al. [62])*

Nous noterons aussi la complexité de réplication des résultats inhérente aux modèles de langages basés sur les réseaux de neurones, tel que souligné par Henderson et al. dans [75], Islam et al. dans [76] ou encore Reimers et Gurevych dans [77].

2.6 Évaluation de la performance des modèles de langage

L'évaluation et, par le fait même, la comparaison des modèles de langage se font habituellement à l'aide de deux mesures principales[25], soit la perplexité et le taux d'erreur des mots. La perplexité, étroitement liée à l'entropie, nous donne un indice sur la capacité d'un ML de prédire un (ou un ensemble de) corpus. Bien que cette mesure est généralement corrélée à une amélioration de la reconnaissance automatique de la parole, cela n'est pas toujours le cas [78], mais demeure tout de même utile. C'est pourquoi, pour les applications spécifiques à la RAP, nous évaluons les modèles de langage par rapport au taux d'erreur des mots. Les sous-sections ci-dessous introduisent ces concepts importants un peu plus en détail.

2.6.1 Perplexité

L'évaluation **extrinsèque** des modèles de langage est généralement un exercice assez long. Nous devons, par exemple, effectuer une évaluation complète (de bout en bout) de la RAP sur un ou plusieurs corpus de test donné pour pouvoir comparer des modèles entre eux. Celui qui obtiendra le plus faible taux d'erreurs des mots sera généralement considéré comme étant le meilleur dans

[25] Loin d'être idéales [21], mais qui nous donnent tout de même un bon aperçu du pouvoir prédictif des modèles de langage.

la pratique. Par contre, cela est généralement assez long, pouvant prendre de plusieurs heures à plusieurs jours dépendamment de la rigueur et de la complexité du test.

C'est pourquoi nous allons généralement débuter par une évaluation **intrinsèque** de nos modèles de langage. La **perplexité** est le moyen le plus souvent utilisé pour l'évaluation rapide des modèles de langage. La perplexité, une mesure basée sur l'entropie croisée, nous permet de mesurer la capacité du modèle de langage à prédire un corpus de test donné (généré à l'aide d'une distribution de référence p inconnue). Par ailleurs, plus sa capacité de prédiction est bonne (une haute probabilité sera attribuée), plus le modèle de langage est considéré comme bon. Par ailleurs, la perplexité est souvent utilisée comme fonction objective lors de l'optimisation des modèles de langage.

Formellement, la perplexité, $PP(T)$, d'un modèle de langage sur un ensemble de test est une fonction de la probabilité que le modèle de langage va attribuer à l'ensemble de test. Pour un ensemble de test $T = w_1, w_2, \dots, w_N$ d'une longueur N :

$$PP(T) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1, \dots, w_{i-1})}}. \tag{2.58}$$

Puisque nous utilisons des *N*-grammes, cela peut se généraliser à

$$PP(T) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|h)}}. \tag{2.59}$$

Une propriété intéressante de la perplexité est qu'en prenant son logarithme nous obtenons l'**entropie**[26] du modèle de langage, propriété familière à tout praticien de la théorie de l'information. La réduction de l'entropie était d'ailleurs l'objectif initial du lissage Witten-Bell qui était destiné à la compression de texte.

2.6.2 Taux d'erreur des mots (Word Error Rate)

Le taux d'erreur des mots (TEM), ou « *Word Error Rate* (WER) », en anglais, est généralement reconnu comme étant la mesure principale de la performance de la reconnaissance automatique de

[26] Quantité moyenne d'information par symbole. Un symbole peut être un mot, une lettre, etc. Le logarithme à la base 2 est habituellement choisi puisque celui-ci **nous donne le nombre moyen de bits/symbole**, ce qui est très utile dans tout processus de télécommunication en général**.**

la parole. Le TEM relève de l'application de la distance de Levenshtein au niveau des mots plutôt que des phonèmes : un outil nous permettant de mesurer la similarité entre deux chaînes de caractères. Pour ce faire nous tentons d'aligner les deux chaînes et nous calculons le nombre minimal de caractères (ou mots dans notre cas) qu'il faut supprimer, insérer ou remplacer pour passer d'une chaîne (l'hypothèse de reconnaissance) à l'autre (le texte de référence). Par ailleurs, contrairement à la distance de Levenshtein où nous sommons simplement les erreurs, le WER comptera le nombre de suppressions, de substitutions et d'insertions tel que :

$$TEM = \frac{(S + O + I)}{N}, \tag{2.60}$$

où N est le nombre de mots dans la phrase de référence ;

S est le nombre de substitutions (mots incorrectement reconnus) par rapport à la phrase de référence ;

O est le nombre de suppressions (mots omis) par rapport à la phrase de la référence ;

I est le nombre d'insertions (mots ajoutés) à la phrase de référence.

Parfois (peut-être pour sonner un peu plus positif), on rapporte plutôt le taux de reconnaissance des mots « *Word Recognition Rate* » (WRR) :

$$WRR = \frac{N - (S + O + I)}{N} \tag{2.61}$$

La majeure partie de la recherche effectuée dans le domaine du traitement du signal et de la modélisation du langage vise donc en fin de compte à réduire le taux d'erreur. Cependant, on peut arguer que ce n'est pas nécessairement la meilleure mesure de performance réelle [79], en particulier dans un contexte de compréhension de la parole.

3 LISSAGE DES *N*-GRAMMES BASÉ SUR LES ONTOLOGIES

3.1 Mise en contexte

Nos travaux de thèse ont tout d'abord débuté dans le but d'être en mesure de concevoir rapidement des interfaces basées sur le dialogue pour la gestion d'inventaire assistée par des réseaux d'identification par radiofréquence (RFID), en particulier dans des environnements bruyants. Les avantages d'un tel système, basé sur le dialogue, complètement mains libres et combiné à la technologie RFID, promet d'améliorer la rapidité, l'ergonomie ainsi que la sécurité de plusieurs opérations telles que la prise de commandes, la gestion de la qualité, le suivi et la surveillance des actifs, les opérations d'expédition, et autres opérations découlant. Notre objectif premier était de proposer à une compagnie locale un système complet de suivi d'inventaire qui mettrait à l'avant-plan notre expertise dans le domaine.

Le système qui a résulté de cette première étape de nos travaux permet à l'utilisateur travaillant dans un environnement très bruyant d'interagir par la parole avec un réseau d'identification RFID. Un nouveau cadre de dialogue fut proposé dans [80] afin de donner aux opérateurs humains la capacité de communiquer avec le système d'une façon plus naturelle. Tel que nous l'avons mentionné dans l'introduction, un des problèmes les plus courants attribués aux interfaces opérant sur le modèle de « commande et contrôle » est leur manque de flexibilité par rapport aux énoncés reconnaissables. Afin de rendre notre système plus flexible, nous avons choisi d'implémenter un gestionnaire de dialogue basé sur le langage AIML (*Artificial Intelligence Markup Language*) qui a été conçu pour le développement d'agents conversationnels (visant le clavardage). Ce langage permet au système de répondre à des requêtes formulées en langage naturel par l'intermédiaire de détection de motifs (*pattern matching*). Cela a aussi comme avantage de rendre le système beaucoup plus robuste par rapport aux erreurs de reconnaissance automatique de la parole (les erreurs sur les mots sans importance sont ignorées), ce qui est particulièrement problématique dans des environnements très bruyants. De plus, afin de faire face à des environnements très bruyants et changeants, une technique en ligne d'amélioration du sous-espace vectoriel du signal basé sur

la variance de l'erreur de reconstruction de Karhunen-Loève Transform (ERV-KLT), basée sur les travaux de [81], [82], est proposé par notre collègue. Nous avons démontré dans [80] le caractère naturel du dialogue et sa robustesse face aux erreurs de reconnaissance automatique de la parole, ainsi que l'efficacité de l'amélioration du signal d'un point de vue perceptuel. Cependant, il demeure qu'un des désavantages principaux que nous avons rencontrés avec le langage AIML est la nécessité de créer des motifs pour toutes les variations d'une commande possible. Par exemple, un des problèmes rencontrés est qu'une simple requête de recherche d'inventaire nécessite une multitude de motifs pour couvrir tous les lemmes, les conjugaisons et toutes autres variations possibles. Ce travail de conception requiert beaucoup de temps et est sujet aux oublis et à l'erreur humaine.

3.1.1 Générateur de motif et analyseur sémantique pour systèmes de dialogue verbaux

C'est avec l'idée de faciliter la conception de vocabulaire que nous avons proposé une amélioration au gestionnaire de dialogue dans [83]. La solution proposée vise à faciliter la conception de grammaire/motifs types de domaine via la mise en œuvre d'un générateur de motif basé sur les ontologies. Ce nouveau cadre de dialogue est proposé afin de donner aux opérateurs humains une plus grande flexibilité de communication avec le système qui sera dorénavant en mesure de tenir compte des variations possibles des commandes types. Avec ce nouveau cadre, le concepteur n'a qu'à spécifier un motif par action ou requête possible et le système va automatiquement générer le plus de variations possible en se basant sur les relations ontologiques de ses termes principaux.

Nous avons choisi de nous orienter vers les ontologies du fait de leur capacité de relier des concepts de façon à la fois relationnelle et hiérarchique et en nous inspirant, entre autres, des travaux d'Estival *et al*. [84]. L'algorithme génératif est quant à lui inspiré des travaux de Kauchak *et. al.* [85] sur la paraphrase dans le contexte de l'évaluation automatique des résultats de traduction automatique du langage.

Nos résultats [83] démontrent que le système est en mesure de générer de nouveaux motifs à partir des motifs de base tout en minimisant l'impact au niveau de la perplexité du modèle de langage généré. De plus, ces mêmes motifs permettent une plus grande flexibilité d'élocution de la part des utilisateurs dans la mesure où ils n'ont pas à retenir un vocabulaire fixe, mais ont simplement besoin de comprendre ce qu'ils veulent faire.

Cependant, un problème persiste. Si les modèles de langage ne sont pas en mesure de tenir compte de ces nouvelles variations, le système ne sera pas capable de les reconnaître. Nous avons donc focalisé notre attention vers les modèles de langage qui permettent justement la reconnaissance de telles phrases nouvelles.

3.2 Lissage ontologique

Dans cette section nous résumons notre contribution principale, soit une méthode originale de lissage de modèles de langage visant à améliorer la performance des *N*-grammes. Un des problèmes avec les techniques actuelles est qu'elles ne se généralisent pas nécessairement aux mots se trouvant dans la même classe/sous-classe sémantique, surtout s'ils n'ont jamais été vus. Prenons l'exemple suivant traduit (traduction libre) de la chanson enfantine « `Mary had a little lamb` » :

`Marie avait un petit agneau.`

Les comptes des mots tels que chaton, chiot, souris, etc. sont tous des animaux et il serait raisonnable de les lisser de façon proportionnelle au mot agneau, et ce, même s'ils n'ont jamais été vus (il en va de même pour le verbe avoir et l'adjectif petit, voir même le prénom). En effet, les phrases :

- `Marie voudrait un petit chaton.`
- `Marie a un grand chien.`
- `Jean avait une petite souris.`

sont toutes des phrases sémantiquement similaires et valides en français. Alors, comment-fait-on pour modéliser convenablement ces phrases si nous ne les avons jamais vues ? Bien que la modélisation basée sur les classes pourrait potentiellement nous aider, qu'advient-il si nous n'avons jamais rencontré ces mots en particulier ? C'est pour répondre à cette question que nous avons proposé dans [86], et amélioré dans [87], une méthode de lissage qui tient compte de ces relations en mettant à profit l'information latente inhérente aux ontologies.

3.2.1 Les ontologies

L'ontologie est « … l'étude de l'être en tant qu'être… » (Aristote [21]). De ce fait, les ontologies sont utilisées pour structurer les termes et les concepts propres à un (ou plusieurs) domaine de connaissance de façon hiérarchique et relationnelle afin d'ultimement **décrire leur sens** et ainsi nous **permettre de raisonner** à propos de ces objets. Les entités logiques et philosophiques peuvent elles-mêmes être regroupées de façon contextuelle à l'aide d'opérateurs relationnels (relations sémantiques, relations de subsomption) nommés, p. ex. : T_{erme}:chat $R_{elation}$:manger T_{erme}:souris. La figure suivante (Figure 3.1) montre une partie de l'ontologie WordNet par rapport au mot « cat » (chat en anglais) pour une distance (chemin le plus court) de 1 et où les relations sont celles de synonymie.

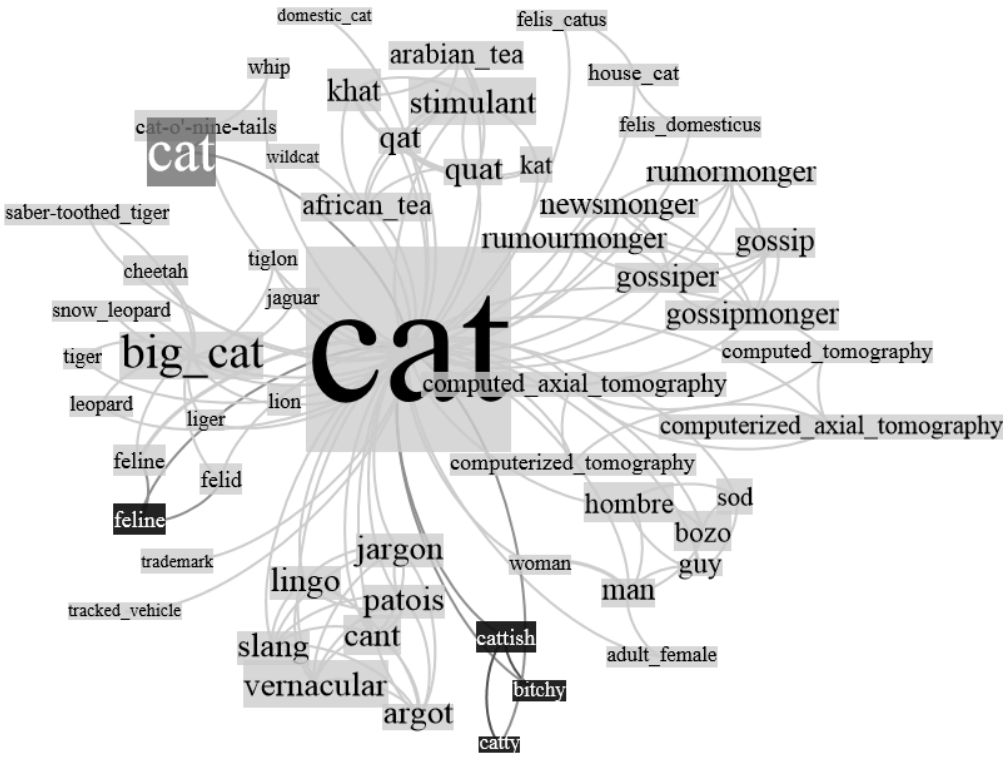


Figure 3.1 : Échantillon de l'ontologie complète ayant comme nœud central le terme chat. La taille des nœuds représente leur degré de connectivité et leur couleur leur catégorie grammaticale : adjectifs (noir), adverbes (non représentés, blanc), noms (gris clair) et des verbes (gris). Les couleurs des arêtes représentent le type de relation entre les nœuds. On notera que la longueur des lignes n'a aucune signification autre que l'amélioration de la clarté de l'illustration.

Source : Y. Benahmed, S.-A. Selouani, et D. O'Shaughnessy, « A Bin-Based Ontological Framework for Low-Resource N-Gram Smoothing in Language Modelling », dans Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on, 2014, p. 4918-4922.

Puisque les ontologies (informatiques) contiennent une quantité importante d'information et qu'on peut en résumer le rôle comme « [les] ontologies sont aux données ce que la grammaire est au langage » [22]. Il devient apparent que les ontologies devraient être en mesure de nous permettre d'aider nos modèles de langage à mieux prendre en compte les relations entre les mots et nous donner des modèles qui reflètent notre compréhension du langage.

L'objectif principal est donc d'exploiter ces ontologies pour introduire une notion de savoir dans les modèles de langage, c'est-à-dire, d'être en mesure d'attribuer des probabilités plus élevées à des *N*-grammes inédits, mais qui sont sémantiquement liées à des *N*-grammes vus ou qui fonctionnent de façon similaire. En d'autres mots nous tenterons à la fois de paraphraser[27] les *N*-grammes déjà vus et de générer des *N*-grammes décrivant des sujets similaires. Nous désirons ainsi à la fois réduire la perplexité des modèles de langage et augmenter la performance de la reconnaissance de la parole grâce à nos algorithmes de lissages ontologiques.

De cette façon, en analysant l'information contenue à l'intérieur de celle-ci, nous serons en mesure de lisser nos modèles de langage en lissant ses probabilités avec celles des termes connexes, c'est-à-dire utiliser les comptes des *N*-grammes apparentés $C(w_r|h)$ pour chaque mot $w_r \in \boldsymbol{W}_R$ apparenté à w_n et son contexte h et où $\boldsymbol{W}_R$ est un vecteur de R mots apparentés à w_n. Inversement, pour lisser les *N*-grammes jamais vus, nous pouvons utiliser le compte $C(w_n|h)$ des *N*-grammes connus pour générer des comptes lissés $C(w_r|h)$ sémantiquement plausibles.

Nous pensons que cette hypothèse est raisonnable, surtout pour les dialogues parlés où les locuteurs peuvent échanger des mots pour des termes connexes qui leur viennent à l'esprit plus rapidement. En outre, étant donné que ce travail vise à aider à réduire la dispersion des données qui sont utilisées pour le lissage, nous proposons la technique suivante basée sur l'interpolation pour lisser les modèles de langage à faibles ressources en ajoutant intelligemment des informations latentes. En d'autres mots, nous cherchons à remplir, de façon intelligente, une matrice creuse.

L'ontologie WordNet

Pour ce travail, nous avons utilisé l'ontologie WordNet [88]. Il s'agit essentiellement d'une grande base de données lexicale de la langue anglaise et où l'unité principale est le sens des mots

[27] paraphrase (n.f.) (grec *paraphrasis*) : Formulation différente d'un énoncé sans altération de son contenu [23].

représenté par des *synsets* (synonymes cognitifs). Nous avons utilisé la forme relationnelle où les relations principales caractérisent la synonymie (l'état d'être un synonyme). Chaque lemme est classifié selon sa catégorie grammaticale et compte : 21 479 adjectifs, 4 481 adverbes, 117 798 noms et 11 529 verbes pour un total de 155 287 **sens** uniques. Nous avons choisi cette ontologie en particulier, car plusieurs librairies existaient déjà pour la traiter [89], [90] en plus d'être bien documentée et utilisée dans plusieurs travaux de recherche.

3.2.2 Bin-Based Ontological Smoothing

Le lissage consiste à créer B_d bacs de comptes lissés qui tiennent compte de tous les mots reliés w_r^d trouvés à une distance d, le chemin le plus court (nombre d'arcs à parcourir), de l'ontologie à partir du mot original w_n et où $d = 1,2,\ldots,d_{max}$ où d_{max} est la distance maximale de w_n qui sera considérée. Par exemple, dans la Figure 3.1 la distance $d(\text{cat},\text{tiger}) = 1$, $d(\texttt{feline},\texttt{cattish}) = 2$, etc. Plus formellement, pour chaque $C(w_n|w_{n-N+1}^{n-1}) \in B_d$:

$$C'^{(w_n|w_{n-N+1}^{n-1})} = \frac{1}{R}\sum_{d=1}^{d_{max}}\sum_{r=1}^{R_d}\frac{C(w_r|w_{n-N+1}^{n-1})}{d} \tag{3.1}$$

où R est le nombre total de mots qui ont une relation avec w_n jusqu'à une distance d_{max}. De cette façon, R est le nombre total de mots reliés à chaque mot w_n et $R_d \leq R = \sum_{d=1}^{d_{max}} R_d$ est le nombre de mots connexes qui sont à une distance d. Le principe derrière l'utilisation du chemin le plus court en tant que mesure de la distance conceptuelle et sémantique est justifié dans [91], soit « *...when is-a hierarchies are defined ... shortest path length can be used to measure conceptual distance between concepts* » qui peut se traduire comme « lorsque des hiérarchies "**est-un**" sont définies, le chemin le plus court peut être utilisé pour mesurer la distance conceptuelle entre deux concepts ».

Les relations que nous considérons sont celles de synonymie, d'hyperonymie et d'hyponymie. Le *Larousse* définit ces termes comme suit :

- Synonymie : « Relation qu'entretiennent entre eux divers termes ou expressions ayant le même sens ou un sens voisin ».
- Hyperonymie : « Rapport d'inclusion entre des unités lexicales, considéré comme orienté du plus général au plus spécifique. (C'est l'inverse de l'hyponymie.) [Chien est dans un

rapport d'hyperonymie avec basset, caniche, etc.] ».
Indicatif d'une **descente vers une feuille** dans la hiérarchie de l'ontologie.

- Hyponymie : « Rapport d'inclusion entre des unités lexicales, considéré comme orienté du plus spécifique au plus général. (C'est l'inverse de l'hyperonymie.) [Chien est dans un rapport d'hyponymie avec carnivore, animal, etc.] ».
  Indicatif d'une **remontée vers la racine** dans la hiérarchie de l'ontologie.

Les comptes générés ontologiquement sont ensuite utilisés pour générer des modèles de langage LM_{B_d} pour chaque bac. Enfin, ces modèles de langage sont interpolés avec le modèle de langue principal lissé à l'aide de l'algorithme de Witten-Bell afin d'obtenir un modèle final :

$$P'(w_n|w_{n-N+1}^{n-1}) = \lambda_1 P(w_n|w_{n-N+1}^{n-1}) + \sum_{d=1}^{d_{max}} \lambda_{d+1} P_{B_d}(w_n|w_{n-N+1}^{n-1}) \quad \textbf{(3.2)}$$

où d_{max} est la distance maximale des arêtes considérées ;
$P_{B_d}(w_n|w_{n-N+1}^{n-1})$ sont les probabilités obtenues à partir des comptes lissés de chaque bac ;

λ sont les poids de mixture, avec $\sum_{i=1}^{D} \lambda_i = 1$ pour s'assurer que le tout somme à 1 et peuvent être obtenus grâce à n'importe quel algorithme d'optimisation linéaire. La raison principale qui nous a amené à utiliser le lissage Witten-Bell, plutôt que le lissage Kneser-Ney, vient premièrement de son lissage basé sur la diversité des contextes et deuxièmement, de façon plus pragmatique, nous l'avons choisi, car nous ne sommes pas au courant de l'existence d'une version **continue** de l'algorithme de lissage Kneser-Ney, ce qui le rend incompatible avec les comptes fractionnels générés par notre algorithme de lissage.

3.3 Méthodologie expérimentale

La méthodologie expérimentale générale que nous avons utilisée pour l'évaluation de notre algorithme de lissage novateur peut se décrire comme suit. L'ensemble des expérimentations effectuées dans cette thèse ont été réalisées à l'aide du corpus WSJ1 [92] (Continuous Speech Recognition Phase II) basé sur le Wall Street Journal. Nous formons nos modèles de langage de

base sur les 76 136 phrases fournies par l'ensemble de formation du corpus WSJ1 pour un total de 1 243 340 mots.

L'ensemble de développement/validation, comportant 4 340 phrases et un total de 71 759 mots, est utilisé pour l'optimisation des paramètres pertinents. Nous nous en servons donc pour optimiser les poids d'interpolation globaux, définis par l'équation 3.2, en utilisant soit la perplexité, soit le taux d'erreur des mots, comme fonction objective à minimiser. Dans cette section, nous optimisons nos poids d'interpolation grâce à l'algorithme d'optimisation de Powell [93] tiré de la bibliothèque SciPy [111] avec comme fonction objective le taux d'erreur des mots.

Les boîtes à outils SRILM [93] et HTK [112] ont été utilisées pour nos expériences de modélisation du langage et de reconnaissance automatique de la parole, respectivement. Nous obtenons tout d'abord les comptes bruts pour nos modèles à l'aide de la boîte à outils SRILM. Ces comptes bruts sont ensuite utilisés pour générer des bacs de comptes lissés ontologiquement pour chaque groupe de distance ontologique. Le vocabulaire des comptes lissés n'étant pas restreint, de nouveaux mots pourront s'introduire aux bacs de comptes lissés, ceux-ci seront par la suite filtrés lors de la création des modèles de langage finaux (options « `-vocab [dictionnaire] -limit-vocab` » du programme `ngram` de la suite SRILM). Notons que, conformément à l'équation (7.2), la majorité des comptes sont des comptes fractionnaires. Ensuite, des ML Witten-Bell sont créés pour chaque bac. Enfin, un modèle de langage mixte est créé en mélangeant chaque ML à base de bacs avec le LM Witten-Bell original. Malheureusement, la trousse d'outils SRILM ne prend pas en charge les comptes fractionnaires pour le lissage de Kneser-Ney (*continuous Kneser Ney counts*) et, par conséquent, nous ne pouvons pas interpoler correctement le modèle de base de KN avec nos bacs.

3.3.1 Résultats expérimentaux

Comme première évaluation de notre méthode de lissage ontologique, nous avons utilisé les ensembles d'évaluation/test Hub 1 (lecture de base du WSJ) et Spoke 1 (adaptation des modèles de langage) (10 347 mots dans 582 phrases) du corpus WSJ1 sans filtrer les phrases avec des mots hors vocabulaire. Le dictionnaire fermé WSJ de 20 k mots est utilisé pour un taux total de mots hors vocabulaire (MHV) de 5,41 %.

Nos résultats expérimentaux [86] démontrent qu'il est possible d'obtenir, d'une part, une réduction significative de la perplexité du modèle (jusqu'à 9,85 % par rapport au modèle de base) et, d'autre part, de réduire le taux d'erreur de mots d'une manière statistiquement significative par rapport à

la fois au modèle original lissé à l'aide de l'algorithme de Witten-Bell et d'un autre modèle de langue de base lissé à l'aide de l'algorithme de Kneser-Ney entrainé sur le corpus « Wall Street Journal-Based Continuous Speech Recognition Phase II » [92].

Nous sommes d'avis que ces résultats intéressants permettent de poser les bases théoriques pour que d'autres chercheurs puissent appliquer cette technique de lissage ontologiques pour les langues où les ressources textuelles propres à la modélisation du langage sont limitées.

Suite à ces résultats encourageants, nous nous sommes posé la question à savoir si la théorie des graphes, telle qu'appliquée dans le domaine de la théorie de l'information et des réseaux (sociaux), pouvait apporter de l'information supplémentaire à nos modèles. C'est pourquoi nous avons par la suite examiné l'apport des mesures tirées de la théorie des graphes à la généralisation de notre algorithme de lissage de modèle de langage. Plus spécifiquement, nous avons exploré l'effet des mesures HITS, PageRank, Modularité et degré pondéré sur la performance de la reconnaissance automatique de la parole, de même que par rapport à la perplexité des modèles de langage.

3.4 Théorie des réseaux

La théorie des graphes, ou, (philosophiquement) plus spécifiquement dans notre cas, l'étude des réseaux, est un outil qui nous permet de quantifier et de qualifier l'influence des nœuds et des arcs à l'intérieur d'un réseau. Ce champ d'études sert aussi à détecter des communautés qui demeureraient cachées autrement. Son utilité, autant dans les domaines de la biologie, de la sociologie, de l'économie, en dit long sur sa capacité de modélisation des relations. Ce sont ces propriétés qui nous intéressent dans ce travail.

Les types de mesures que nous avons choisis sont les mesures de centralité. Ces mesures soulignent l'importance des nœuds dans un graphe. Dans le contexte de la modélisation du langage, cela pourrait, par exemple, être l'importance d'un mot dans une ontologie, ou par rapport à un document (nombre d'occurrences pour un document D). Cette intuition est d'ailleurs à la base de multiples algorithmes en recherche de l'information et de classification des documents tels que le TF-IDF, HITS, PageRank, Modularité, etc.

Le choix des mesures HITS et PageRank et Modularité est motivé par leur simplicité, leur capacité de rapidement calculer l'importance relative des nœuds, que l'algorithme PageRank peut être considéré comme une modélisation d'un processus Markovien [94] et parce que les approximations de premier ordre de plusieurs algorithmes de classement donnent les mêmes résultats [95]. Le choix de la mesure de modularité est quant à elle dû à sa capacité de rapidement classer les nœuds en communautés. Finalement, le degré pondéré est utilisé comme référence de base.

3.4.1 Degré

Le degré et son dérivé, le degré pondéré, sont les mesures les plus simples de centralité d'un graphe. Ils comptent essentiellement le nombre d'arcs incidents, tel que défini par

$$\begin{aligned} d_i^- &= \sum_j A(j,i) \\ d_i^+ &= \sum_j A(i,j) \\ d_i &= d_i^- + d_i^+ \end{aligned} \tag{3.3}$$

où d_i^- est le degré entrant pondéré, le nombre d'arcs qui entrent vers le nœud i si $A(j,i) = 1 \forall\ M(\bar{i},j)$,

d_i^+ est le degré sortant pondéré, le nombre d'arcs qui sortent du nœud i si $A(i,j) = 1 \forall\ M(\overset{+}{i},j)$,

d_i est le degré pondéré,

$A(i,j)$ est le poids de l'arc entre les nœuds i et j et est égal à un (1) dans le cas du degré non pondéré.

Nous pouvons alors réécrire cette équation dans le contexte d'un réseau de mots en tant que

$$d(w_i) = \sum_{w_j \in M(w_i)} A(w_i, w_j), \tag{3.4}$$

où $d(w_i)$ est le degré du mot w_i

$w_j \in M(w_i)$ est l'ensemble des mots w_j qui sont liés à w_i

$A(w_i, w_j)$ est le poids de l'arc entre les mots w_i et w_j.

Si l'on reprend l'ontologie vue à la Figure 3.1, nous pouvons par exemple calculer le degré du mot « `cat` » :

$$d(\text{cat}) = d^{-}(\text{cat}) + d^{+}(\text{cat}) = 57 \tag{3.5}$$

Afin d'éviter les débordements arithmétiques, il n'est pas rare de normaliser les degrés sur une échelle de 0 à 1.

Deux autres mesures peuvent par ailleurs être utilisées soit le degré entrant d^{-}, et le degré sortant d^{+} qui quantifient respectivement le poids des arcs vers le nœud w_i et le poids des arcs hors du nœud w_i. Ces mesures sont d'ailleurs utilisées comme base pour d'autres mesures.

3.4.2 Hyperlink-Induced Topic Search

L'algorithme HITS (*Hyperlink-Induced Topic Search*) [96] a originalement été conçu pour évaluer l'autorité des pages Web en analysant leurs liens. Il retourne une liste de centralité de hubs et d'autorité pour chaque nœud dans le graphe. La relation entre hub et autorité est maintenue à l'aide d'un algorithme mutuellement récursif :

$$aut(w_i) = \sum_{w_j^{-} \in M(w_i)} hub(w_j, w_i)\,, hub(w_i) = \sum_{w_j^{+} \in M(w_i)} aut(w_i, w_j) \tag{3.6}$$

où $aut(w_i)$ et $hub(w_i)$ sont les scores d'autorité et de hub du mot w_i respectivement. L'algorithme procède de la sorte :

1. On assigne 1 comme valeur d'autorité et de hub pour chaque mot.
2. On met à jour la valeur autorité de chaque mot en sommant la valeur hub des mots $w_j^{-} \in M(w_i)$ qui pointent vers w_i (liens entrants).
3. On met à jour la valeur hub de chaque mot en sommant la valeur autorité des mots $w_j^{+} \in M(w_i)$ vers lesquels w_i pointe (liens sortants).
4. On recommence jusqu'à la stabilisation du système : $|aut_n(w_i) - auth_{n-1}(w_i)| \leq \epsilon$ et $|hub_n(w_i) - hub_{n-1}(w_i)| \leq \epsilon$.

Cet algorithme nous permettra donc de déterminer les nœuds d'importance dans le « graphe », en d'autres mots, une bonne *autorité* est pointée par plusieurs bons hubs, et qu'un bon hub pointe vers plusieurs bonnes autorités. Dans le cas des graphes non dirigés, les valeurs de hub et d'autorité sont égales. Le choix de l'une au l'autre revient donc à un choix personnel/philosophique. Notre raisonnement pour l'utilisation de cette mesure est basé sur l'intuition que les hubs représenteront des mots spécifiques, alors que les autorités représenteront des mots plus généraux.

3.4.3 PageRank

L'algorithme PageRank [97] est un algorithme itératif qui nous donne la probabilité qu'un utilisateur parcourant le Web se rende de façon aléatoire à n'importe quelle page donnée. Nous assumons que le mot w_i a des mots $w_j \in M(w_i)$ qui pointent vers lui ; $M(w_i)$ est définis précédemment comme étant l'ensemble des mots liés par w_i. Le PageRank d'un mot w_i est donc :

$$PR(w_i) = \frac{1-\delta}{N} + \delta \left(\sum_{w_j \in M(w_i)} \frac{PR(w_j)}{L(w_j)} \right) \tag{3.7}$$

Où $PR(w_i)$ est le PageRank du mot w_i, $L(w_j)$ est le nombre de mots liés par w_j. Le paramètre δ est un facteur d'amortissement qui prend une valeur entre 0 et 1 et est habituellement égal à $\delta = 0.85$.

Un des avantages de l'algorithme PageRank est qu'il peut aussi être appliqué aux graphes (notons que le WWW est en fait un graphe et que le bût de l'algorithme PageRank est essentiellement d'attribuer une cote d'importance relative à chacun des nœuds ($PR(x_s) \in \{S_1, S_2, \dots, S_s\} \in \{WWW\}$). L'hypothèse de ce travail est qu'un faible indice PageRank suggère une faible probabilité de substitution d'un mot w_i pour un mot w_j. Le PageRank sert donc essentiellement de facteur de pondération où $0 \leq PR \leq 1$ avec 0 indiquant aucune probabilité de substitution et 1 une « garantie » de substitution. En pratique, le PageRank est toujours inférieur à 1 puisqu'une garantie de substitution est fondamentalement illogique (le mot original ne serait jamais réutilisé).

La supposition de ce travail est qu'un faible PageRank suggère une faible probabilité de substitution du mot w_i par le mot w_j, nous l'utiliserons donc comme un facteur de pondération sachant que $0 \leq PR \leq 1$.

3.4.4 Modularité

La modularité est utilisée pour découvrir et caractériser la structure des communautés dans un réseau. La qualité $-1 \leq Q \leq 1$ de ces communautés C est définie [7] par leur Modularité [98] telle que :

$$Q = \frac{1}{2m} \sum_{w_i, w_j} \left[A(w_i, w_j) - \frac{d(w_i), d(w_j)}{2m} \right] \sigma(c(w_i), c(w_j)), \tag{3.8}$$

où $A(w_i, w_j)$ le poids de l'arc entre les mots w_i, w_j,

$c(w)$ retourne la communauté à laquelle appartient le mot w,

$d(w)$ le degré (non pondéré) du mot w,

$m = \frac{1}{2} \sum_{w_i, w_j} A(w_i, w_j)$ est un facteur de normalisation du degré pondéré,

la fonction $\sigma(u, v)$ est égale à 1 si $u = v$ et 0 sinon (en d'autres mots, 1 s'ils font partie de la même communauté et 0 sinon).

L'intuition derrière le choix de cette mesure est basée sur l'agrégation par thème (*topic clustering*) où nous supposons que la transition d'une classe de modularité à une autre classe de modularité indique un changement de contexte. Pour ce travail, la modularité est donc utilisée comme un facteur de pénalisation lorsque les mots liés n'appartiennent pas à la même communauté. Nous nous en servons aussi pour découvrir des sujets inconnus ou des classes de lemmes à l'intérieur de l'ontologie et pénaliser les comptes des mots de différentes communautés dans l'équation (3.1). Nous avons choisi de définir le facteur de pénalité α tel que :

$$\alpha(w_n, w_r) = \begin{cases} 1 & si\ c(w_n) = c(w_r) \\ \frac{1}{e} & sinon \end{cases}. \tag{3.9}$$

3.4.5 Méthodologie expérimentale

Tel que définie à la section 3.3, notre méthodologie expérimentale de base se base sur le corpus *Wall Street Journal-based Continuous Speech Recognition Phase II* et les boîtes à outils SRILM et HTK.

Tel que décrit à la section 3.3, nous avons utilisé la perplexité du modèle de langage sur l'ensemble de développement (~4,3 k mots) comme fonction objective pour optimiser les poids d'interpolation de nos modèles de langage (voir équation 3.2).

Nous avons par la suite évalué nos résultats sur les ensembles d'évaluation Hub 1 (lecture de base du WSJ), Spoke 1 (adaptation des modèles de langage), Spoke 2 (indépendance du domaine) et Spoke 9 (dictée spontanée du WSJ) pour un total de 23 k mots dans 1 285 sentences sans filtrage. Le dictionnaire fermé WSJ de 20 k mots est utilisé pour l'évaluation, donnant un taux total de MHV de 5,64 %.

3.4.6 Résultats expérimentaux

L'ensemble des parties d'un ensemble (*powerset*) des bacs a été évalué pour chaque mesure de réseau. Le *powerset* désigne l'ensemble de tous les sous-ensembles d'un ensemble S, incluant S lui-même, ainsi que l'ensemble vide. Par exemple, admettons l'ensemble $S = \{x, y, z\}$, son ensemble des parties d'un ensemble, $\wp(S)$, sera [99] :

$$\wp(S) = \{\{\}, \{x\}, \{y\}, \{z\}, \{x, y\}, \{x, z\}, \{y, z\}, \{x, y, z\}\}$$

Nos résultats [87] montrent que l'interpolation des bacs originaux à des distances $d = 1, 3$ et 5 donne lieu à une amélioration du taux d'erreur des mots de 0,71 % par rapport à l'interpolation des bacs 1 à 5 tout en réduisant la taille des modèles. Finalement, nous avons trouvé que la modularité, le PageRank et l'HITS sont prometteurs et méritent une étude plus approfondie dans le futur. Par ailleurs, les relations d'antonymie mériteraient aussi d'être explorées dans le futur dans un contexte de paraphrase puisque les expressions de négation ou antonymiques sont tout aussi sémantiquement plausibles, par exemple :

- `Jean a un chat`
- `Jean n'a pas de chat`
- `Marie aime les moutons`
- `Marie déteste les moutons`

Comme nos travaux dans [87] ont démontré qu'il pouvait exister des combinaisons irrégulières des bacs de distances, nous avons décidé de pousser notre analyse plus en profondeur jusqu'aux bacs d'une distance de 7.

Finalement, nous avons observé que les algorithmes évolutionnaires nous offrent la possibilité d'explorer de vastes espaces de recherche de façon moins « séquentielle » que les algorithmes d'optimisation classiques. Ces algorithmes stochastiques devraient nous permettre d'élargir l'espace de recherche pour pouvoir ensuite éliminer les bacs ainsi que les ordres qui ne contribuent pas beaucoup au modèle et donc de faire ressortir les paramètres les plus importants de nos modèles de langage. Par ailleurs, puisque ces problèmes d'optimisation sont complexes en raison du grand nombre de paramètres, le choix des algorithmes évolutionnaires nous semble particulièrement pertinent. Dans ce contexte, nous en profiterons pour comparer la performance de différents algorithmes d'optimisation dans le but de trouver les meilleurs poids d'interpolation des modèles de langage.

4 OPTIMISATION DE LA MODÉLISATION DU LANGAGE PAR LES ALGORITHMES ÉVOLUTIONNAIRES

4.1 Introduction

Les *algorithmes évolutionnaires* (AE) ont une longue histoire dans le domaine de l'optimisation. Ces algorithmes permettent, par le biais de la simulation de processus évolutionnaires biologiques (stochastiques), de couvrir une grande partie de l'espace de recherche, pour autant qu'ils soient bien adaptés au problème à l'étude (ce qui est d'ailleurs une des faiblesses souvent décriées de l'optimisation par algorithmes évolutionnaires), de façon à trouver rapidement une solution globalement satisfaisante. Un avantage considérable est qu'ils ont généralement moins tendance à tomber dans le piège d'une solution locale, contrairement à plusieurs méthodes d'optimisation classiques. Cela est principalement dû au fait que la plupart de ces algorithmes intègrent un principe de mutation qui leur permet de découvrir des solutions globalement satisfaisantes.

Des centaines d'algorithmes évolutionnaires ont été utilisés avec succès pour optimiser une grande variété de problèmes au courant des dernières décennies.

Pour ce qui est du domaine de la parole, ils ont été utilisés avec succès pour le rehaussement de la parole, la reconnaissance de locuteurs, l'adaptation au locuteur, la reconnaissance automatique de la parole, et le traitement du langage [100]. Par contre, outre leur utilisation pour l'optimisation des réseaux de neurones (modèles de langage basés sur les réseaux de neurones), les AE n'ont pas, à notre connaissance, été utilisés de façon notable dans l'optimisation des modèles de langage.

Nous proposons donc d'évaluer la capacité des AE à trouver des solutions optimales pour l'interpolation de nos modèles de langage et de les comparer aux algorithmes classiques principalement utilisés dans le domaine : la méthode de Powell [101] et l'optimisation par l'algorithme espérance-maximisation.

Nous désirions aussi évaluer les algorithmes évolutionnaires pour déterminer leur capacité de dynamiquement minimiser la taille de nos modèles de langage en réduisant le nombre de bacs à interpoler avec le modèle de langage principal. Nous avons montré dans [87] qu'il est possible de réduire le nombre de bacs tout en préservant des performances optimales.

Contrairement aux algorithmes conventionnels, nous croyons qu'il est possible pour les algorithmes évolutionnaires de réaliser cette opération en une seule exécution, due à leurs propriétés stochastiques, contrairement aux algorithmes conventionnels qui requièrent l'exploration complète de l'ensemble des parties d'un ensemble[28] possibles.

Dans ce travail, nous nous intéresserons principalement aux algorithmes évolutionnaires bioinspirés que l'on retrouve dans deux grandes catégories[29] :

1. Les algorithmes génétiques, qui tentent d'adapter une solution à un espace de recherche en simulant les processus (darwiniens) évolutifs d'une espèce : croisement des gènes, mutations, survie du plus apte.
2. Les algorithmes collaboratifs, qui simulent les stratégies de survie de groupes d'individus : le mouvement d'un banc de poissons pour se nourrir ou minimiser la prédation, le mouvement d'une nuée d'oiseau dans une volée mixte d'alimentation, le comportement d'une colonie de fourmis à la recherche de nourriture (principe du superorganisme), etc.

Plus précisément, nous avons principalement examiné un algorithme génétique et un algorithme basé sur les colonies, soit l'optimisation par essaims particulaires (OEP).

Les sections 4.2 et 4.3 présentent, respectivement, une introduction aux algorithmes génétiques et à l'optimisation par essaims particulaires.

[28] Pour l'ensemble $E = \{x, y, z\}$, l'ensemble des parties d'un ensemble, $\mathcal{P}(E)$, ou, le *Power Set* en anglais, est $\mathcal{P}(E) = \{\{\}, \{x\}, \{y\}, \{z\}, \{x, y\}, \{x, z\}, \{y, z\}, \{x, y, z\}\}$.

[29] Nous retrouvons aussi maintenant de plus en plus d'algorithmes inspirés par divers phénomènes physiques, tels que l'optimisation par (champ de) force centrale (*Central Force Optimization*, CFO) [102], [103] et, plus récemment, l'algorithme gravitationnel de recherche (*Gravitational Search Algorithm*, GSA) [104], [105]. Le premier est un algorithme déterministe de recherche multidimensionnel, simulant une sonde interstellaire qui parcourt l'espace de recherche sous l'influence de la gravité. Le second simule un ensemble de masses (solutions potentielles) qui interagissent entre elles selon les lois newtoniennes de la gravité et du mouvement.

Dans la section 4.4, nous évaluons les différents algorithmes d'optimisation et comparons leur performance face à la méthode de Powell qui est couramment utilisée pour optimiser les modèles de langage [36], [78], [106].

Finalement, la section 4.5 conclut notre travail et propose des directions pour des travaux futurs.

4.2 Algorithmes génétiques

Les algorithmes génétiques (AG), *Genetic Algorithms* en anglais, sont probablement les algorithmes évolutionnaires les plus connus. En s'inspirant de la théorie de l'évolution, les AG reproduisent la **sélection naturelle** où les meilleurs individus (qui **survivent**) d'une espèce vont se **reproduire** dans le but de rendre l'espèce plus forte, et, du même coup, profiter des **mutations** aléatoires inhérentes à la biologie qui ont permis à la vie **d'évoluer et de s'adapter** à différents écosystèmes.

4.2.1 Individus

À la base de tout algorithme évolutionnaire repose le concept d'individu. L'individu représente un membre de l'espèce que l'on veut faire évoluer. Plus concrètement, dans le domaine de l'optimisation, les individus représentent une solution possible au problème. Ils sont généralement représentés par leur gène qui consiste en un vecteur de paramètres à « optimiser » et où chaque élément du vecteur constitue un allèle.

4.2.2 Génération

Chaque génération comporte un nombre plus ou moins élevé d'individus. On appelle la première génération la génération initiale. Plusieurs méthodes d'initialisation de la génération initiale existent et nous en choisissons une en fonction de nos connaissances du problème à l'étude. Lorsque nos connaissances du problème sont limitées, nous allons souvent procéder à une initialisation dite aléatoire[30] des paramètres des individus. À chaque génération un certain nombre d'individus seront **sélectionnés** pour se **croiser**, passer à la génération suivante ou mourir en fonction de leur performance (telle que mesurée selon une fonction objective).

[30] Voir le chapitre 6 de [107] pour une discussion sur le hasard et la chance.

4.2.3 Sélection

La sélection est l'action de choisir des individus qui se reproduiront et/ou survivront à la prochaine génération. C'est cette étape que l'on considère comme étant l'analogue à la sélection naturelle. Comme pour le processus biologique, on aura tendance à choisir les individus les mieux **adaptés** (meilleure performance) pour se reproduire et générer des enfants qui seront évalués à la prochaine génération. L'objectif est donc de prendre les meilleures solutions connues jusqu'à présent et les **croiser** dans le but d'obtenir progressivement de meilleurs résultats. Certains de ces enfants performeront bien, et d'autres moins bien, dans un processus de diversification qui convergera idéalement vers une solution satisfaisante.

C'est cette étape qui est la plus longue puisqu'elle exige une évaluation de tous les individus de la population. C'est justement cette fonction d'évaluation, la **fonction objective**, qui peut exiger un temps de calcul considérable. Prenons par exemple le cas de l'évaluation d'un modèle de langage pour la reconnaissance automatique de la parole. La meilleure fonction objective, donc la **mesure de performance absolue**, est celle qui utiliserait le taux d'erreur de reconnaissance. Par contre, cela nous contraint énormément en termes de temps, puisque l'évaluation complète d'un corpus de test demande du temps :

> Par exemple, admettons 10 minutes par évaluation par individu avec une population de 100 individus sur 200 générations. Nous aurons alors l'équivalent de 139 jours de calcul !

Il serait peut-être alors plus judicieux d'utiliser une autre mesure proxy de performance, une **mesure de performance relative**, par exemple la perplexité qui se calcule beaucoup plus rapidement. Toutefois, nous devons nous assurer de bien choisir/concevoir notre fonction objective puisqu'ultimement ce sera elle qui déterminera le sort de notre optimisation.

Plusieurs algorithmes de sélection sont possibles, mais le type de sélection le plus simple est celui de la sélection par roulette (roue de la Fortune).

4.2.3.1 Roulette

Dans la sélection par roulette, les chances de sélection pour la reproduction sont proportionnelles à la performance de chaque individu. Admettons cinq individus, chacun ayant la mesure de performance suivante :

- Individu 1 : score = 10
- Individu 2 : score = 20
- Individu 3 : score = 30
- Individu 4 : score = 40
- Individu 5 : score = 50

Ici, l'individu 5 est le mieux adapté et l'individu 1 est le moins adapté. Nous créerons alors une roue de la Fortune où chaque secteur représentera la probabilité d'être choisi en fonction de la performance relative de chaque individu. Nous choisirons alors les paires de parents en « tournant » la roulette. Dans notre exemple, l'individu 1 a 7 % de chance d'être choisi, l'individu 2, 13 % et ainsi de suite (voir Figure 4.1 pour l'ensemble des probabilités). Une fois le premier parent choisi, nous procédons à un deuxième tour de roulette, et ce, jusqu'à ce qu'un parent différent soit choisi. Nous pouvons éviter d'avoir la chance de retomber sur le premier individu choisi en mettant à jour notre roue pour tenir compte des probabilités ajustées (Figure 4.2).

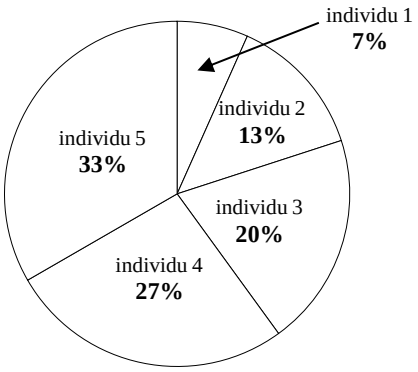


Figure 4.1 : Illustration du concept de la roulette dans le contexte d'un algorithme génétique. Chaque secteur du graphique en secteurs représente les probabilités, relatives à leur performance, d'être choisi comme parent pour chaque individu de la population.

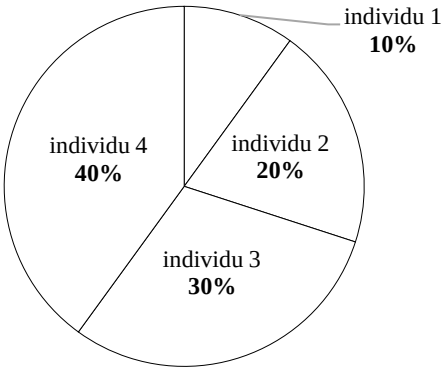


Figure 4.2 : Illustration des probabilités relatives restantes ajustées après le tirage au premier tour de l'individu 5.

4.2.3.2 Tournoi

Pour la sélection par tournoi, on choisit $\tau \geq 2$ individus pour participer à un tournoi. Le tournoi consiste à comparer la performance des individus choisis pour déterminer le gagnant qui servira de parent pour la reproduction.

Dans sa version la plus simple, le **tournoi strict**, l'individu ayant la performance la plus élevée sera toujours choisi, et, si l'individu le plus performant de la population est choisi dans un tournoi, il gagnera avec 100 % de probabilité. Dans un **tournoi souple**, tous les individus ont une chance $0 < p < 1$ de gagner.

4.2.3.3 Logique de l'étalon

Dans la logique de l'étalon, l'individu avec la meilleure performance de chaque génération est toujours utilisé pour toutes les opérations de recombinaison. Simon [108] montre comment cette logique de sélection tend à beaucoup mieux performer que les algorithmes qui ne les mettent pas en œuvre. Par ailleurs, cette logique a comme avantage de réduire le coût computationnel de sélection puisque l'on a un seul autre individu au minimum (versus 2) à choisir par opération de reproduction.

4.2.3.4 Logique de l'élitisme

Dans la logique de l'élitisme, nous choisirons toujours les E meilleurs individus pour passer à la prochaine génération. Une des motivations derrière cette logique est que les meilleurs individus agissent comme une mémoire des meilleures solutions potentielles. En effet, dû aux effets de

croisement et de mutation, nous n'avons aucune garantie que les individus de la génération $g+1$ soient mieux adaptés que ceux de la génération g.

Nous pouvons accomplir cela soit en produisant $N-E$ descendants où N est la taille de la population et E est le nombre d'élites à conserver. Une autre façon serait de produire N descendants et remplacer les E moins performants par les E élites de la génération précédente.

De façon générale, les AE qui implémentent l'élitisme performent généralement mieux que les AE qui ne l'implémentent pas, sauf pour certains cas particuliers (fonctions objectives dispendieuses ou dynamiques [108]).

4.2.4 Croisement

La reproduction des individus se fait par croisement[31], un processus analogue à l'enjambement des chromosomes. Pour le **croisement simple**, nous choisissons une position de croisement aléatoirement pour chaque paire d'individus (parents) choisis pour la reproduction et nous effectuerons un croisement entre les deux à partir du point choisi. La Figure 4.3 présente un exemple d'un croisement entre deux individus dans un cas d'AG binaire : pour chaque paire de parents, une paire d'enfants est conçue. C'est cette étape qui est responsable de l'exploration de l'espace de recherche par le biais de la création de nouveaux individus qui intègrent les paramètres de leurs parents et qui nous permettent (idéalement) de **diversifier** la population.

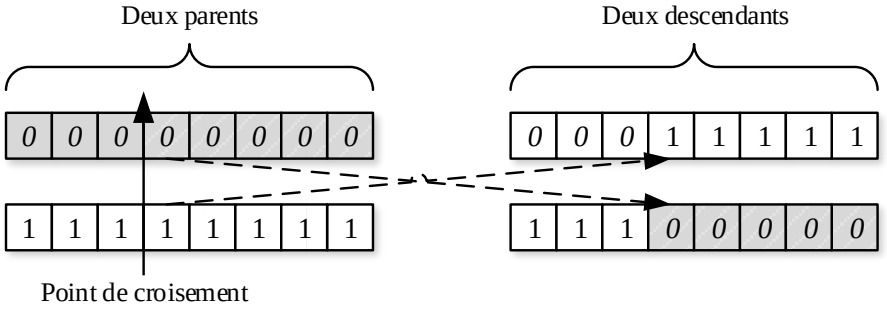


Figure 4.3 : Illustration du processus de croisement, à partir du quatrième allèle, pour un algorithme génétique binaire. Le point de croisement est généralement choisi de façon aléatoire.

Nous pouvons généraliser le croisement simple en croisement en Mpoints par l'équation :

[31] Enjambement pour reprendre le terme biologique, ou encore entrecroisement sont tous deux des termes acceptables et couramment utilisés dans la littérature.

$$
\begin{array}{lcll}
\boldsymbol{y_1}(g) & \leftarrow & [& x_a(1) \quad ... \quad x_a(m_1) \\
 & & & x_b(m_1+1) \quad ... \quad x_b(m_2) \\
 & & & ... \quad ... \quad ... \\
 & & & x_a(m_{m-1}+1) \quad ... \quad x_a(m_m) \\
 & & & x_b(m_m+1) \quad ... \quad x_b(m_{m+1}) \\
 & & & ... \quad ... \quad ... \\
 & & & x_b(m_{M-1}+1) \quad ... \quad x_b(m_M) \\
 & & & x_a(m_M+1) \quad ... \quad x_a(N) \quad] \\
\boldsymbol{y_2}(g) & \leftarrow & [& x_b(1) \quad ... \quad x_b(m_1) \\
 & & & x_a(m_1+1) \quad ... \quad x_a(m_2) \\
 & & & ... \quad ... \quad ... \\
 & & & x_b(m_{m-1}+1) \quad ... \quad x_b(m_m) \\
 & & & x_a(m_m+1) \quad ... \quad x_a(m_{m+1}) \\
 & & & ... \quad ... \quad ... \\
 & & & x_a(m_{M-1}+1) \quad ... \quad x_a(m_M) \\
 & & & x_b(m_M+1) \quad ... \quad x_b(N) \quad]
\end{array}
\qquad \textbf{(4.1)}
$$

où $\boldsymbol{y_i}(g)$ est le $i^{\text{ème}}$ descendant de la génération g des parents $\boldsymbol{x_a}$ et $\boldsymbol{x_b}$,

$x_\beta(k)$ représente le $k^{\text{ème}}$ allèle du chromosome $\boldsymbol{x_\beta}$,

N est le nombre total d'allèles dans le chromosome,

m_m sont des points de croisement,

M est le nombre de points de croisement, si $M = 0$ alors $\boldsymbol{y_i}(g)$ seront des clones et si $M = 1$ on aura un croisement simple, $M = 2$, un croisement double, etc.

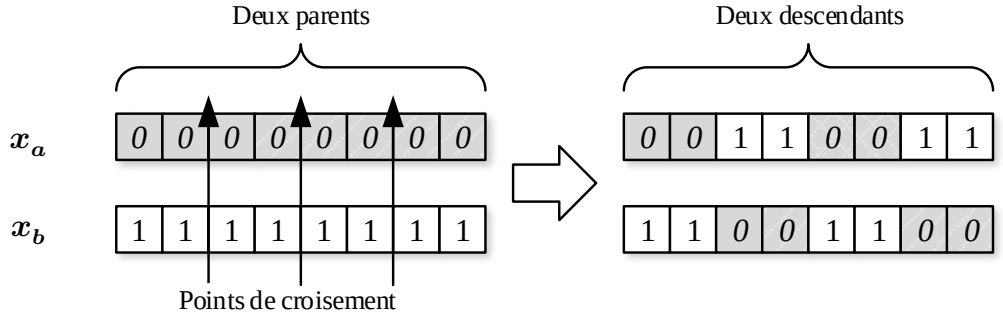


Figure 4.4 : Illustration du processus de croisement multipoints, pour un algorithme génétique binaire. Le point de croisement est généralement choisi de façon aléatoire.

Le cas des algorithmes génétiques continus peut, pour sa part, être un peu plus complexe que le cas des AG binaires. En effet, il n'est pas toujours approprié d'effectuer une reproduction par

croisement. Nous présentons ci-dessous deux méthodes qui nous ont servi pour notre travail de thèse (voir chapitres 8 et 9 pour plus de détails), soit le croisement arithmétique et le croisement heuristique.

4.2.4.1 Arithmétique

Le croisement arithmétique est utilisé exclusivement avec les algorithmes génétiques continus. Il s'apparente au processus d'interpolation utilisé dans la modélisation du langage

$$\begin{aligned} \boldsymbol{y}(g) &\leftarrow U[x_a(k),\ x_b(k)] \\ &= \alpha x_a(k) + (1-\alpha)x_b(k) \end{aligned} \tag{4.2}$$

où $\alpha \sim U[0,1]$ est tiré d'une distribution uniforme (ou tout autre type de distribution) entre 0 et 1,

$0 < k < N$ est le $k$ième allèle dans un chromosome.

$$\begin{aligned} \boldsymbol{y}_1(g) &= \alpha x_a(k) + (1-\alpha)x_b(k) \\ \boldsymbol{y}_2(g) &= (1-\alpha)x_a(k) + \alpha x_b(k) \end{aligned}. \tag{4.3}$$

Un des avantages du croisement arithmétique est qu'il nous est possible de garantir le respect des bornes est des contraintes linéaires [109].

4.2.4.2 Heuristique

L'algorithme de croisement heuristique décrit ici est celui utilisé que nous avons utilisé tel que défini par [109]. Ce croisement heuristique va retourner un seul enfant qui se retrouve à une courte distance du parent ayant la meilleure performance dans une direction opposée à celle du parent avec la pire performance et est défini par l'équation :

$$\boldsymbol{y}(g) = \boldsymbol{x_b} + R * (\boldsymbol{x_a} - \boldsymbol{x_b}), \tag{4.4}$$

où la distance de l'enfant par rapport au meilleur parent est déterminée par le ratio R et où $\boldsymbol{y}(g) = \boldsymbol{x_a}$ quand $R = 1$.

4.2.5 Mutation

Un des problèmes avec les AG est que de bons candidats initiaux peuvent monopoliser la reproduction et biaiser la solution pour nous amener vers un optimum local. Pour contrer cela,

nous faisons appel à la mutation. La mutation nous permet de **diversifier** notre population, et, par le fait même, d'explorer un espace de solution plus vaste.

Tout comme son analogue biologique, la mutation est un phénomène qui se doit d'être assez rare, sinon nous aurions essentiellement une évolution complètement aléatoire de nos gènes. À l'inverse, un taux de mutation trop faible ne nous permettra pas d'explorer efficacement l'espace de recherche.

Nous permettrons donc à chaque allèle de chaque enfant de muter selon une probabilité ρ prédéfinie, tel que (pour un AG binaire) :

$$\boldsymbol{r} \leftarrow U[0,1]$$

$$y(k) \leftarrow \begin{cases} y(k), & r(k) \geq \rho \\ y(k)^{-1}, & r(k) < \rho \end{cases}, \tag{4.5}$$

où $\boldsymbol{r}$ est un vecteur contenant un nombre entre 0 et 1 tiré d'une distribution uniforme (ou tout autre type de distribution) pour chaque allèle,

ρ est la probabilité de mutation,

$y(k)^{-1}$ est l'inverse de $y(k)$: $0\ si\ y(k) = 1, 1\ sinon.$

La Figure 4.5 montre un exemple d'une mutation sur un chromosome dans le cas d'un AG binaire.

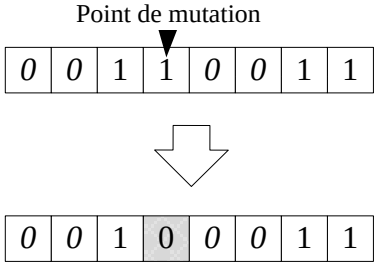


Figure 4.5 : Illustration du résultat de la mutation d'un allèle sur un chromosome dans le cas d'un AG binaire.

Dans le cas d'un AG continu, plusieurs options de mutation s'offrent à nous. Une des plus simples est la mutation uniforme centrée sur l'individu :

$$\boldsymbol{r}_i \leftarrow U[0,1]$$

$$y_i(k) \leftarrow \begin{cases} y_i(k), & \boldsymbol{r}_i(k) \geq \rho \\ U\left[y_i(k) - \alpha_i(k), y_i(k) + \alpha_i(k)\right], & \boldsymbol{r}_i(k) < \rho \end{cases} \quad \textbf{(4.6)}$$

$$\alpha_i(k) = \min\left(y_i(k) - y_{min}(k), y_{max}(k) - y_i(k)\right),$$

où $\boldsymbol{r}_i$ est un vecteur contenant un ensemble de nombres entre 0 et 1 tirés d'une distribution uniforme pour le $i$ème enfant de la population et où $i \in [1, N]$,

y_i est le $i$ème enfant de la population,

$y_{min}(k)$ et $y_{max}(k)$ sont respectivement les valeurs minimales et maximales du $k$ième allèle de la population.

α est alors choisi afin de maximiser l'exploration de l'espace de recherche sans, toutefois, dépasser les bornes du domaine de recherche.

4.2.6 Critère d'arrêt

Comme nous ne pouvons connaître d'avance la solution, il est important de définir un ou des critères d'arrêts afin d'éviter que l'algorithme **tourne à l'infini.** Parmi les choix usuels, nous retrouvons des critères d'arrêts basés sur le nombre de générations, la différence entre la solution ou un seuil que l'on considère comme adéquat.

4.2.7 Algorithme général de l'optimisation par algorithme génétique

Cela nous permet de définir un algorithme génétique général :

Initialiser la première génération, une population aléatoire de N individus $\boldsymbol{x} = \{x_i\}, i \in [1, N]$
Définir la fonction de sélection f_s
Définir la fonction de croisement f_c
Définir la fonction de mutation f_m
Définir le nombre d'élites $\phi_{\acute{e}}$
Définir le taux de croisement ϕ_c
Définir le taux de mutation ϕ_m
Évaluer la performance de chaque individu $p_i \leftarrow f(x_i), i \in [1, N]$
Tant que non (critère de terminaison)
    Pour chaque individu $x_i, i \in [1, N]$
        $p_i \leftarrow f(x_i)$
    Prochain individu
    Définir les $\phi_{\acute{e}}$ élites

$$\boldsymbol{E}_{\bullet} = \left\{p_{(1)}, \dots, p_{(\phi_{\acute{e}})}\right\} = \left\{\min_{i=1}^{\phi_{\acute{e}}} \boldsymbol{p}_{\bullet}, \max_{i=1}^{2} \boldsymbol{p}_{\bullet}, \dots, \max_{i=1}^{\phi_{\acute{e}}} \boldsymbol{p}_{\bullet}\right\}, i \in [1, \phi_{\acute{e}}]$$

    Ajouter les élites à la prochaine génération $\boldsymbol{y} \leftarrow \boldsymbol{E}_{\bullet}$
    Pour $k = 1{:}\, N * \phi_c$
        Sélectionner les candidats pour le croisement
        Si Étalon
            $x_{p1} = \boldsymbol{E}_{(1)}$
            Choisir deuxième parent $x_{p2} = f_s(\boldsymbol{x})$
        Sinon
            Choisir premier parent $x_{p1} = f_s(\boldsymbol{x})$
            Choisir deuxième parent $x_{p2} = f_s(\boldsymbol{x})$
        Fin Si
        Croiser les deux parents $\boldsymbol{y} \leftarrow c_k = f_c(x_{p1}, x_{p2})$ et ajouter à la prochaine génération
    Prochain k
    Pour $k = 1{:}\, N * \phi_m$
        Sélectionner aléatoirement un candidat pour la mutation $x_{k,m}$
        Muter l'individu et ajouter le mutant à la prochaine génération $\boldsymbol{y} \leftarrow f_m(x_{k,m})$
    Prochain k
    Les individus de la génération $\boldsymbol{y}$ deviennent les nouveaux $\boldsymbol{x}$, $\boldsymbol{x} = \boldsymbol{y}, \boldsymbol{y} = \{\emptyset\}$
Prochaine génération

Figure 4.6 : Algorithme général de l'optimisation par algorithme génétique.

Où $f(x)$ est la fonction objective utilisée pour évaluer la performance des individus.

$\boldsymbol{E}_{\bullet}$ est l'ensemble ordonné des élites de la génération courante.

$\boldsymbol{p}$ est le vecteur de la performance de chaque individu de la génération courante.

4.3 Optimisation par essaims particulaires

Les algorithmes d'optimisation par essaims particulaires, ou *Particle Swarm Optimization* (PSO) en anglais sont des algorithmes basés sur l'intelligence collective des colonies ou groupes d'individus. L'inspiration de ces algorithmes vient de l'observation des essaims d'oiseaux [110] et des bancs de poissons où les individus semblent démontrer une forme d'intelligence collective soit pour échapper aux prédateurs, soit pour accroitre leur succès alimentaire ou encore pour minimiser leur dépense énergétique [111] pour n'en nommer que quelques-uns.

Chaque individu, ou particule, représente une solution candidate au problème à optimiser. Différents facteurs feront en sorte que chacune de ces particules parcours, généralement de façon linéaire, l'espace paramétrique multidimensionnel à la recherche de la meilleure solution possible. La vitesse et la direction dans lesquelles l'individu parcours l'espace sera influencé par divers paramètres internes (individuels) et externes (influence du groupe).

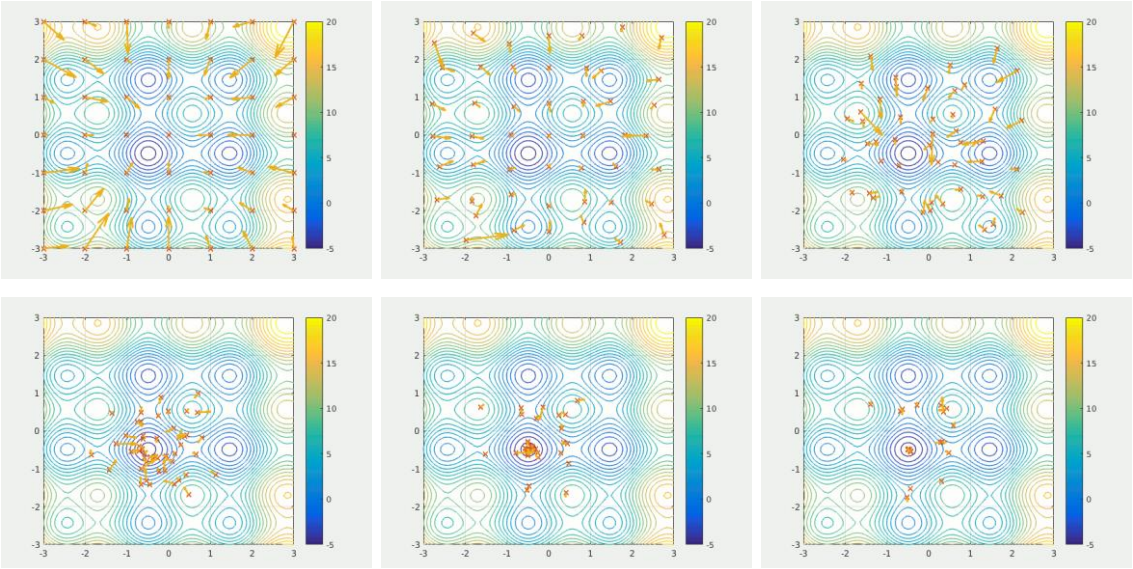


Figure 4.7 : Exemple de convergence d'un essaims particulaires cherchant un minimum global d'une fonction..

Source : Par Ephramac — Travail personnel, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=54975083

4.3.1 Inertie

L'inertie, phénomène à la base physique et posée par Kepler [112] se définit comme étant la résistance d'un objet au changement de son état. Les individus ont tendance à résister au changement, de faire les choses comme avant, l'influence de la société ou du groupe pourra

« encourager » un changement d'attitude. Le principe d'inertie, représenté par ω, tel que définit pour l'optimisation par essaims particulaires a initialement été défini par Shi et Eberhart [113]. Afin d'éviter que les particules ne « bougent » sans arrêt dans l'espace de solution sans converger, la valeur d'inertie sera généralement initialisée à une valeur maximale qui sera par la suite progressivement diminuée jusqu'à 0 : plus la solution est bonne, moins l'individu aura tendance à vouloir bouger, ce qui aide l'algorithme à garantir une convergence éventuelle.

Une autre façon de voir l'impact de l'inertie sur le comportement de l'OEP est que lorsque ω est élevé ($\omega > 1{,}2$), l'OEP se comporte plus comme un algorithme de recherche global, et lorsqu'il est petit $(\omega < 0{,}8)$, l'OEP se comporte plutôt comme un algorithme de recherche local. De cette façon, le résultat final est moins dépendant de l'état de notre population initiale puisque durant les premières générations les individus auront tendance à explorer une grande partie de l'espace de solution possible et au fur et à mesure que l'algorithme progresse, la population aura naturellement tendance à se diriger vers la meilleure solution possible.

4.3.2 Apprentissage individuel

Le principe de l'apprentissage individuel est représenté par le taux d'apprentissage cognitif ϕ_1 qui est basé sur le renforcement positif, c'est-à-dire que l'on apprend de nos succès. Si la solution trouvée à la génération actuelle est meilleure que la meilleure solution trouvée à date par l'individu, celui-ci continuera d'explorer l'espace dans la direction actuelle. Sinon, l'individu ralentira sa cadence pour éventuellement changer de direction si la solution continue de ne pas s'améliorer.

4.3.3 Influence par le groupe

Le principe de l'influence par le groupe est le même que celui de l'influence de la société, ci-dessous, sauf que la sphère d'influence est généralement plus limitée, mais plus forte. Les individus vont donc avoir tendance à suivre le groupe. Dans l'optimisation par essaims particulaires, ce paramètre est défini comme étant le taux d'apprentissage social ϕ_2 : on apprend par le succès de ses pairs. Ce paramètre fera en sorte qu'un individu tende à se diriger vers le voisin (voir section 4.3.5) ayant trouvé la meilleure solution à date, ou faire en sorte que le groupe tende à se diriger vers l'individu ayant trouvé la meilleure solution à date.

4.3.4 Influence de la société

Les individus vont regarder ce que les individus réussissent et vont combattre leur inertie afin d'émuler leur succès. Dans l'optimisation par essaims particulaires, ce paramètre est défini comme étant le taux d'apprentissage global ϕ_3 et s'apparente au concept d'élitisme vu dans les algorithmes génétiques.

Le concept est donc similaire à celui de l'influence par le groupe, mais à l'échelle globale.

4.3.5 Voisinage

Le voisinage définit à la fois la topologie de la relation entre les individus d'un groupe ainsi que la taille de celui-ci. Plusieurs formes de topologies s'offrent à nous et reflètent le type de topologie que l'on pourrait retrouver dans un réseau informatique : anneau, maillage complet, étoile, carré, etc.

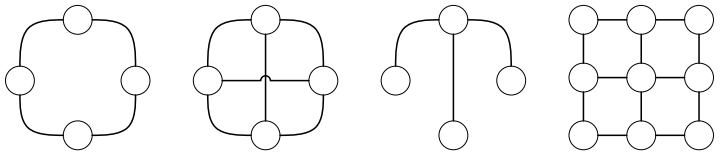


Figure 4.8 : Exemple de topologies de voisinage.

Par ailleurs, ce voisinage peut être défini de façon soit statique : le voisinage est défini au début de l'algorithme et demeure fixe tout au long de son exécution ; soit dynamique : le voisinage va évoluer au fur et à mesure que le l'algorithme progresse et que les individus de déplacent dans l'espace de recherche. Une autre approche populaire consiste en l'initialisation d'un voisinage et, lorsque le taux de convergence semble stagner, on réinitialise le voisinage de façon aléatoire afin de stimuler une plus grande couverture de l'espace de recherche.

4.3.6 Taux d'apprentissage

En plus de l'apprentissage par renforcement positif (ϕ_1, ϕ_2, ϕ_3), les individus peuvent aussi apprendre de leurs propres erreurs ϕ_4 ainsi que des erreurs du groupe ϕ_5 et de la société ϕ_6 : l'apprentissage par le renforcement négatif. Les individus, et la société auront donc tendance à éviter les mauvaises stratégies.

Cela nous permet finalement de généraliser l'équation de la mise à jour de la vitesse telle que :

$$v_i \leftarrow \omega v_i + \phi_1(b_i - x_i) + \phi_2(h_i - x_i) + \phi_3(g - x_i) \\ -\phi_4(\bar{b}_i - x_i) - \phi_5(\bar{h}_i - x_i) - \phi_6(\bar{g} - x_i), \quad \textbf{(4.7)}$$

où : x_i est le $i$ème individu (ou la $i$ème solution candidate)

v_i est le vecteur de vélocité du $i$ème individu (la vitesse de déplacement de l'individu dans les n dimensions de la solution).

ω est le vecteur n-dimensionnel d'inertie. Shi et Eberhart [113] proposent de faire varier, de façon décroissante, l'inertie des particules à chaque génération afin de ralentir les particules, ce qui améliore la convergence.

ϕ_j sont les taux d'apprentissage : ϕ_1 le taux d'apprentissage cognitif, ϕ_2 le taux d'apprentissage social, ϕ_3 le taux d'apprentissage global, ϕ_4, le taux d'apprentissage cognitif d'évitement, ϕ_5, le taux d'évitement social, ϕ_6, le taux d'évitement global.

g et $\bar{g}$ sont, respectivement, la meilleure et la pire solution actuelle de toute la population.

h_i et $\bar{h}_i$ sont, respectivement, la meilleure et la pire solution du voisinage, de taille σ, du $i$ème individu.

b_i et $\bar{b}_i$ sont, respectivement, la meilleure et la pire solution à date du $i$ème individu.

4.3.7 Algorithme général de l'optimisation par essaims particulaires

Finalement, dans la Figure 4.9, nous définissons un algorithme général d'optimisation par essaims particulaires.

Initialiser une population aléatoire de N individus $\boldsymbol{x} = \{x_i\}, i \in [1, N]$
Initialiser le vecteur n-dimensionnel de vélocité de chaque individu $v_i, i \in [1, N]$
Définir la valeur maximale d'inertie ω_{max}
Initialiser la meilleure position à date de chaque individu $b_i \leftarrow x_i, i \in [1, N]$
    Définir la meilleure position à date de toute la société
    $g \leftarrow \text{argmin}\{f(B)\}$
Initialiser la pire position à date de chaque individu $\bar{b}_i \leftarrow x_i, i \in [1, N]$
    Définir la pire position à date de toute la société $\bar{g} \leftarrow \text{argmax}\{f(\overline{B})\}$
Définir la taille du voisinage $\sigma < N$
Définir les valeurs maximales d'influence $\phi_{1,max}, \phi_{2,max}, \phi_{3,max}, \phi_{4,max}, \phi_{5,max}, \phi_{6,max}$
Initialiser le vecteur n-dimensionnel de vélocité maximale $v_{i,max}, i \in [1, n]$
Initialiser le vecteur n-dimensionnel d'inertie $\omega_{i,max}, i \in [1, n]$
Tant que non (critère de terminaison)
    Pour chaque individu $x_i, i \in [1, N]$
        $H_i \leftarrow \{\sigma \text{ voisins les plus près d}'x_i\}$
        $h_i \leftarrow \text{argmin}_x\{f(x): x \in H_i\}$
        $\bar{h}_i \leftarrow \text{argmax}_x\{f(x): x \in H_i\}$
        Pour chaque valeur d'influence $\phi_j, j \in [1,6]$
            Générer un vecteur aléatoire ϕ_j avec $\phi_j(k) \sim U[0, \phi_{j,max}]$
            pour $k \in [1, n]$
        Prochaine valeur d'influence
        $v_i \leftarrow \omega v_i + \phi_1(b_i - x_i) + \phi_2(h_i - x_i) + \phi_3(g - x_i)$
            $-\phi_4(\bar{b}_i - x_i) - \phi_5(\bar{h}_i - x_i) - \phi_6(\bar{g} - x_i)$
        Si $|v_i| > v_{max}$ alors
            $v_i \leftarrow v_i v_{max}/|v_i|$
        Fin si
        $x_i \leftarrow x_i + v_i$
        $b_i \leftarrow \text{argmin}\{f(x_i), f(b_i)\}$
        $\bar{b}_i \leftarrow \text{argmax}\{f(x_i), f(\bar{b}_i)\}$
        $g \leftarrow \text{argmin}\{f(x_i), f(g)\}$
        $\bar{g} \leftarrow \text{argmax}\{f(x_i), f(\bar{g})\}$
    Prochain individu
    Mettre à jour le vecteur d'inertie $\omega = \omega - \delta(t)$
Prochaine génération

Figure 4.9 : Algorithme général de l'optimisation par essaim particulaire.

où $f(x)$ est la fonction objective utilisée pour calculer la performance de l'individu x,

B est l'ensemble des meilleures positions,

$\overline{B}$ est l'ensemble des pires positions,

H_i est le voisinage de l'individu x_i et

$\delta(t)$ est un facteur d'ajustement de l'inertie et où l'inertie diminuera généralement de génération t en génération $t + 1$.

4.4 Évaluation et comparaison de la performance d'algorithmes évolutionnaires pour l'optimisation des poids d'interpolation

Dans cette section nous évaluons la capacité des AE à trouver des solutions optimales pour l'interpolation de nos modèles de langage et les comparerons aux algorithmes classiques principalement utilisés dans le domaine : la méthode de Powell [101] et l'optimisation par l'algorithme espérance-maximisation. Les questions auxquelles nous souhaitons répondre sont les suivantes :

- Est-ce que les algorithmes évolutionnaires peuvent obtenir de meilleurs résultats que les algorithmes classiques ?
- Est-ce que les algorithmes évolutionnaires peuvent obtenir une solution plus rapidement que les algorithmes classiques ?
- Est-ce que les algorithmes évolutionnaires peuvent réduire dynamiquement le nombre de bacs qui seront interpolés avec le modèle de langage principal ?
- Quel est l'effet des différents paramètres sur la vitesse de convergence de l'algorithme génétique ainsi que sur la performance de la solution obtenue ?

Afin de répondre à ces questions, nous avons généré des bacs jusqu'à une distance ontologique de 7 nœuds comparativement à 5 nœuds dans nos travaux précédents [86], [87].

De surcroît, comme les algorithmes évolutionnaires sont de nature stochastique, nous comparons leurs performances respectives à l'aide de la méthode de Monte-Carlo à 55 tours. Cela nous permettra d'avoir un portrait plus juste de leur performance respective.

La procédure utilisée pour générer les modèles de langage de base est la même que celle utilisée pour nos travaux ultérieurs [86], [87] et est décrite plus en détail à la section 3.3.

Nous utilisons l'outil SRILM [93] pour obtenir les comptes bruts du corpus d'entrainement. Nous utilisons ensuite notre algorithme de lissage ontologique pour générer des comptes lissés pour

chaque bac de distance tel que défini à la section 3. Bien que ces comptes lissés comportent maintenant de nouveaux mots, ceux-ci sont éliminés lors de la génération du modèle de langage final. Nous utilisons l'ensemble d'entrainement de 76 000 phrases du corpus WSJ1 [92] pour un total de 1,2 M mots.

Un modèle de base (trigramme) lissé à l'aide de l'algorithme Witten-Bell est créé et servira de modèle de langage de base pour l'interpolation des bacs issus de notre algorithme de lissage basé sur les ontologies.

Les expériences ont été effectuées sur un ordinateur double processeur à 6 cœurs par processeurs avec la technologie Hyper-Threading d'Intel pour un total de 24 processeurs logiques et 48 Gio de mémoire vive.

4.4.1 L'analogie entre paramètres évolutionnaires et ceux du modèle de langage

Afin de faciliter la compréhension de la pertinence de l'approche utilisant les AE que nous proposons pour optimiser la modélisation du langage, nous explicitons dans ce qui suit la relation entre les paramètres évolutionnaires et ceux relatifs au modèle du langage.

Ainsi, les poids d'interpolation représentent les paramètres de la population et la perplexité du modèle de langage interpolé définit la fonction objective :

> Pour l'optimisation par algorithmes génétique, l'analogie est la suivante : chaque individu, défini par un chromosome $\boldsymbol{\alpha}_i$, représente les poids λ d'interpolation des bacs ontologiques, ainsi que les bacs qui seront interpolés. Chaque allèle $a_i(k)$ représente donc soit un poids λ_k d'interpolation, soit un bit d'activation δ_k.
>
> Pour l'optimisation par essaims particulaires, l'analogie est similaire : chaque individu $\boldsymbol{x}_i$ est représenté par sa position dans l'espace de recherche et où sa position est définie par les poids d'interpolation λ_k et les bits d'activation δ_k des bacs ontologiques.

Pour illustrer plus clairement le concept, admettons une distance ontologique $D = 3$ et où $h = w_{n-N+1}^{n-1}$. En reprenant l'équation 3.2 :

$$P'(w_n|h) = \lambda_0 P(w_n|h) + \sum_{d=1}^{3} \lambda_d P_{B_d}(w_n|h).$$

Afin de simplifier l'implémentation du problème d'optimisation, nous définissons

$$\lambda_0 = 1 - \sum_{j=1} \lambda_j$$

Si on développe celle-ci, on obtient alors :

- Un individu sera représenté par un vecteur

$$x_i = \{(\lambda_1, \lambda_2, \lambda_3), (\delta_1, \delta_2, \delta_3)\}$$

Où

$0 \leq \lambda_j \leq 1$ et $0 < \sum_j \lambda_j < 1$ sont les poids d'interpolation des bacs P_{B_j} à optimiser.

$0 \leq \delta_j \leq 1$ sont les marqueurs d'activation correspondants aux bacs P_{B_j} et où la désactivation s'effectue à $\delta_j < 0{,}5$. Les bacs désactivés ne seront donc pas interpolés au modèle de langage principal et leur poids sera redistribué à λ_0.

Admettons un individu $x_i = \{(0{,}1; 0{,}5; 0{,}5); (0{,}8; 0{,}8; 0{,}3)\}$, le modèle de langage pour cet individu sera alors égal à $P'_{x_i}(w_n|h) = 0{,}4P(w_n|h) + 0{,}1P_{B_{d_1}}(w_n|h) + 0{,}5P_{B_{d_2}}(w_n|h)$.

4.4.2 Initialisation et paramètres de la population

Comme pour les algorithmes conventionnels, le choix de topologie des individus peut avoir un impact significatif sur la performance des algorithmes évolutionnaires [108], [114]. En effet, si le choix de paramètres caractérisant le problème n'est pas judicieux, il se peut que l'algorithme converge vers une solution insatisfaisante ou nécessite un très grand nombre de générations pour converger.

Pour notre problème d'optimisation des poids d'interpolation de nos modèles de langage, nous avons opté pour une topologie mixte pour nos chromosomes/individus. Pour fin de simplification nous utiliserons la nomenclature des algorithmes génétiques.

Nos chromosomes sont composés de $K = 2k$ allèles réels et « binaires » simulés et où $k = d_{max}$ bacs potentiels à interpoler. Les k premiers allèles, de format réel, représentent les poids d'interpolation actifs, les k derniers allèles, de format binaire simulé, représentent un vecteur

d'activation des bacs. Cela nous permet à la fois d'optimiser les poids d'interpolation individuels et de désactiver (désactivation à $\delta_i < 0{,}5$) les bacs qui contribuent peu ou pas à la solution. Nous avons choisi comme valeur de désactivation 0,5 afin de simuler une valeur binaire :

$$\delta_i = \begin{cases} 1 & si \quad 0{,}5 \leq \delta_i \leq 1 \\ 0 & si \quad 0 \leq \delta_i < 0{,}5 \end{cases}.$$

Nous noterons ici que le choix de probabilités non équiprobables entre le 0 et 1 binaire a été choisie afin de statistiquement encourager le maintien du plus grand nombre de bacs. Ainsi, si l'algorithme tend vers une désactivation, ceci nous donne une certaine garantie par rapport à son élimination. Nous avons choisi cette topologie particulière afin de renforcer la robustesse de la méthode et d'optimiser le temps de calcul. En effet, lors de nos premières expériences, nous utilisions des chromosomes à valeurs réelles de taille $k = d_{max}$ avec une désactivation des bacs basée sur un seuil plancher ($\lambda_i < 1{,}0E^{-6}$). Cependant, la vitesse de convergence était très lente (approximativement une demi-journée par tentative sur un ordinateur à 24 processeurs logiques et 48 Gio de mémoire vive).

4.4.3 Évaluation de la mutation, de la sélection et du taux de croisement

Dans cette section, nous évaluons l'effet sur la vitesse de convergence de l'optimisation par algorithme génétique de la variation des taux de mutation et de croisement et du nombre de candidats utilisés pour l'algorithme de sélection. Tous les résultats présentés représentent la moyenne de 25 simulations Monte-Carlo.

Dans cette section nous présentons aussi les résultats de l'évaluation des modèles de langage obtenus pour chaque expérience. Comme à la section 4.4.7, les modèles de langage ont été évalués sur le corpus d'évaluation du *Wall Street Journal-based Continuous Speech Recognition (CSR) Corpus Phase II* (WSJ1) [92]. Les ensembles utilisés pour l'évaluation sont les suivants :

- Hub 1 (base de référence WSJ lue),
- Spoke 1 (adaptation des modèles de langage),
- Spoke 2 (indépendance du domaine)
- Spoke 9 (diction spontanée WSJ)

Cela nous donne un total de 23 000 mots dans 1285 phrases sans filtrage des phrases contenant des mots hors vocabulaire. Le vocabulaire (fermé) utilisé est le vocabulaire WSJ standard de 20 000 mots. Le taux de mots hors vocabulaire est de 5,64 %.

Tel que décrit à la section 3.3, l'ensemble de développement/validation, comportant 4 340 phrases et un total de 71 759 mots, est utilisé pour l'optimisation des poids d'interpolation des bacs.

4.4.3.1 Nombre de candidats, sélection par tournoi

Tel que vu à la section, la sélection est l'action de choisir des individus qui se reproduiront et/ou survivront à la prochaine génération. La sélection par tournoi choisit un nombre fixe d'individus de façon aléatoire et choisit le meilleur individu du groupe. Il est donc possible que seulement de mauvaises solutions soient choisies. Nous évaluons donc l'effet du nombre d'individus choisi afin de déterminer si un nombre de candidats plus élevé apportera de meilleurs résultats puisque plus de chance de choisir des individus performants.

Les paramètres fixes utilisés pour l'évaluation sont décrits dans le Tableau 4.1. Notons d'ailleurs qu'afin de légèrement accélérer la vitesse de convergence que nous avons augmenté la valeur *FunctionTolerance* (seuil à partir duquel on considère qu'une solution stable est obtenue), passant de 0,01 à 0,05.

La Figure 4.10 présente la vitesse de convergence moyenne de l'optimisation par algorithme génétique avec de 2 à 15 individus choisis aléatoirement pour le tournoi de sélection.

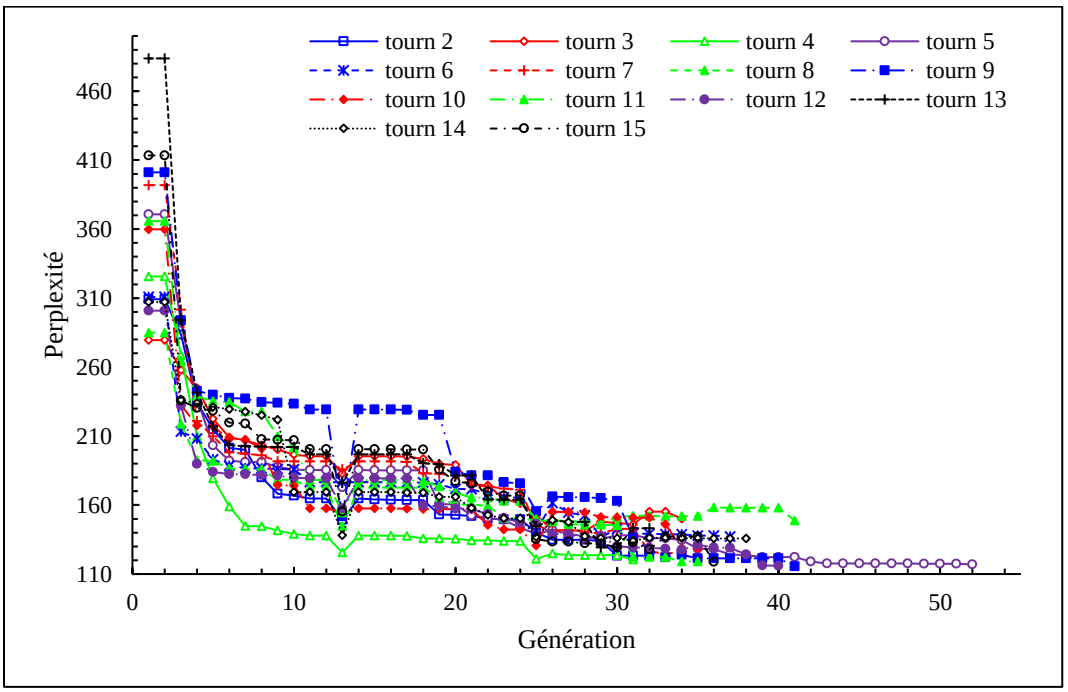
En analysant la Figure 4.10.a, optimisation sans amorçage, nous notons une convergence plus rapide avec 4 individus et, de façon surprenante, la plus lente avec 5 individus (17 générations de plus). La sélection à 8 individus semble avoir tendance à diverger vers les dernières générations. La sélection à 9 individus a un comportement aberrant, mais finit par converger plus rapidement qu'avec 5 individus.

Tableau 4.1 : Paramètres fixes utilisés pour l'évaluation du nombre d'individus choisis pour la fonction de sélection par tournoi.

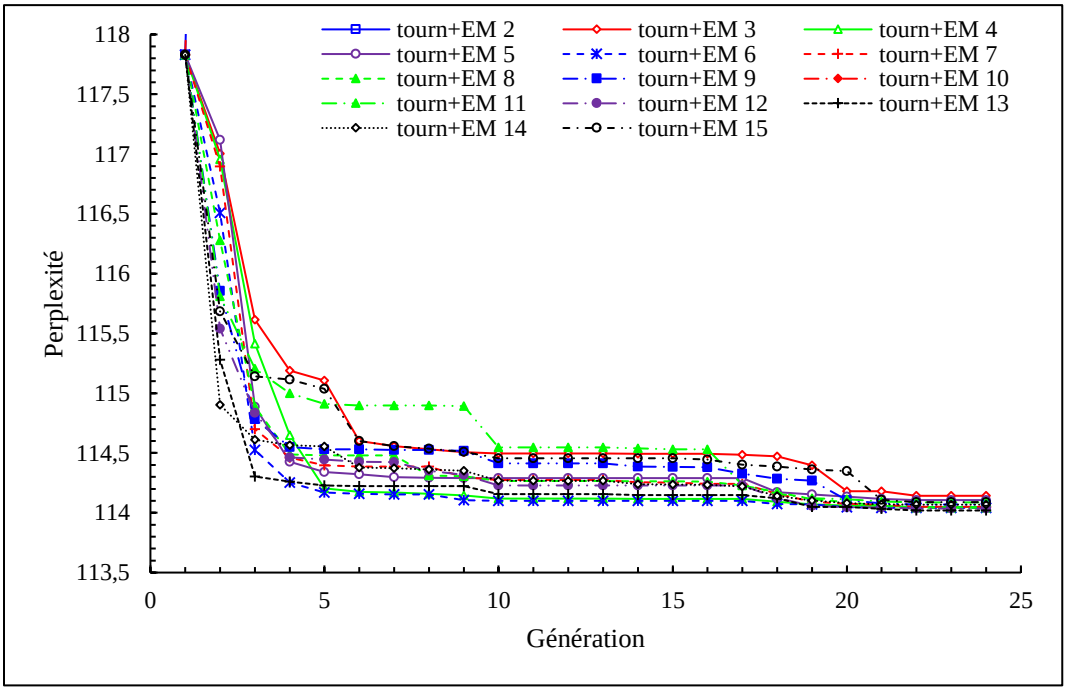
Paramètre	Description	Valeur
PopulationSize	Nombre d'individus	70
CreationFcn	Fonction de création d'individus	@gacreationuniform
MutationFcn	Fonction de mutation	@mutationadaptfeasible
CrossoverFcn	Fonction de croisement	@crossoverarithmetic
SelectionFcn	Fonction de sélection	@selectiontournament
CrossoverFraction	Pourcentage des individus qui sont le fruit d'un croisement	0,8
EliteCount	Nombre d'élites retenus	3
FitnessScalingFcn	Fonction de mise à l'échelle de la fonction de coût	@fitscalingrank
FunctionTolerance	Changement minimum constant considéré avant que l'algorithme termine.	0,05

Les résultats de l'optimisation avec amorçage sont beaucoup plus homogènes que ceux obtenus sans amorçage. Les expériences utilisant 3, 8 et 15 individus ressortent un peu du lot et prennent plus de temps à passer le cap du 114.5 que les autres. Le tournoi à 6 individus converge en moyenne vers une meilleure solution plus rapidement que les autres.

Les Figures 4.10 (a) et (b) présentent les résultats de l'évaluation de la perplexité des modèles de langage obtenus selon les différents paramètres. Les résultats obtenus ne nous permettent pas de prédire les résultats attendus : aucun motif apparent ne ressort. Cela nous porte à conclure que le comportement sera probablement très dépendant du problème à optimiser et que, pour la sélection par tournoi, le seul moyen de choisir le nombre optimal d'individus pour un problème donné est de façon empirique. Cela cadre d'ailleurs avec les applications du « monde réel » où l'on utilisera habituellement, comme dans notre cas, un ensemble de données de développement pour déterminer les paramètres optimaux propres à un problème d'optimisation.



(a)



(b)

Figure 4.10 : Convergence moyenne suite à 25 simulations Monte-Carlo de l'optimisation par AG : (a) représente la convergence moyenne pour un tournoi de 2 à 15 individus sans amorçage ; (b) représente la convergence moyenne pour un tournoi de 2 à 15 individus avec amorçage.

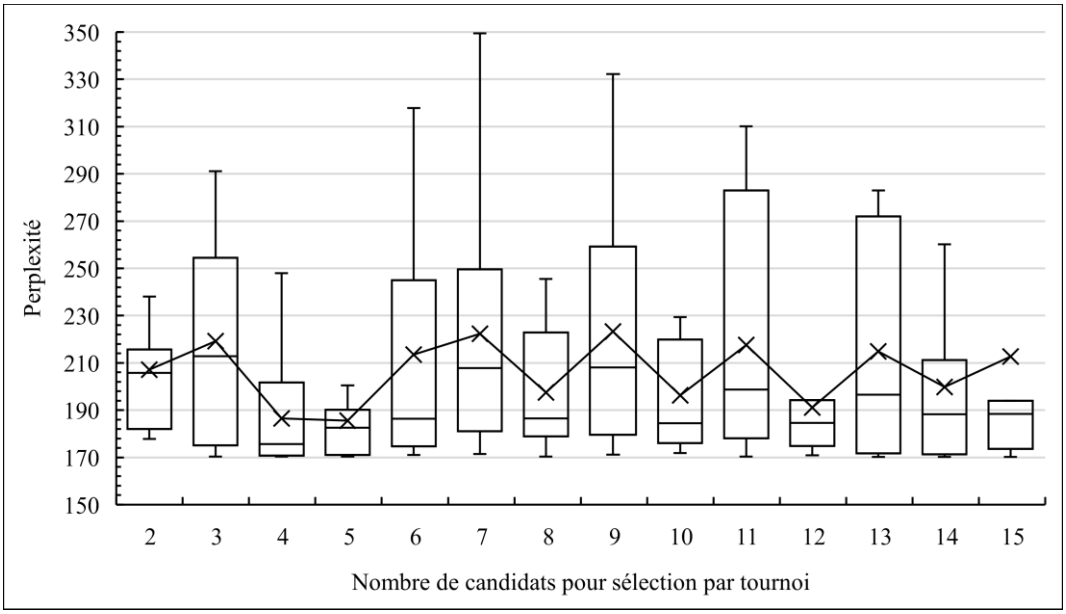
Cela dit, dans notre cas particulier, les meilleurs résultats sont obtenus avec des tournois à 5 individus pour l'optimisation sans amorçage et des tournois à 6 individus pour l'optimisation avec amorçage.

Finalement, le Tableau 4.2 présente un résumé des résultats obtenus pour chaque groupe d'expérience. Sans amorçage, les meilleurs résultats, en moyenne, sont obtenus avec des tournois de 5 individus, tandis que les pires en moyenne le sont avec des tournois de 9 individus. Pour ce qui est de l'optimisation avec amorçage, les meilleurs résultats, en moyenne, sont obtenus avec des tournois à 12 participants.

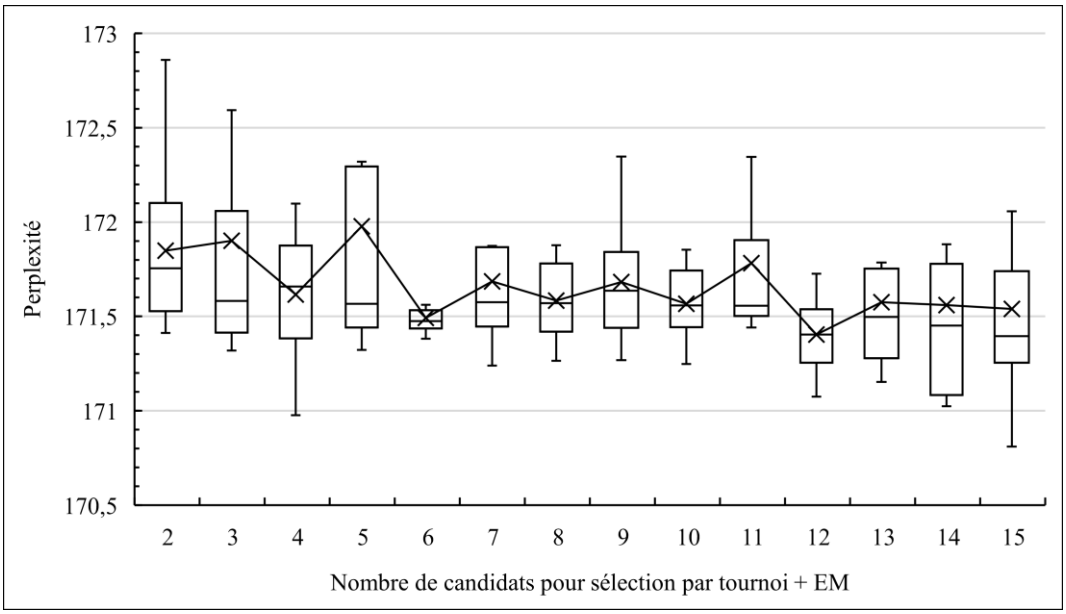
Ces résultats sont cohérents avec ceux obtenus dans la section 4.4.6 où nous avons utilisé des tournois à 6 individus.

Tableau 4.2 : Résumé des résultats obtenus dans le cadre de l'évaluation de l'influence de la taille des tournois sur la perplexité pour optimisation avec et sans amorçage.

Nombre d'individus	Perplexité (tournoi)			Perplexité (tournoi+EM)		
	Moyenne	**Min**	**Max**	**Moyenne**	**Min**	**Max**
2	207,143	177,793	268,193	171,8481	171,413	172,859
3	219,180	170,331	291,152	171,9017	171,319	174,064
4	186,512	170,339	247,984	171,6158	170,977	172,097
5	**185,515**	170,294	**228,911**	171,9779	171,322	174,674
6	213,531	171,008	317,868	171,4919	171,382	**171,682**
7	222,291	171,466	349,469	171,6857	171,239	172,827
8	197,488	170,289	245,475	171,5837	171,264	171,877
9	223,268	171,181	332,296	171,6814	171,269	172,347
10	196,194	171,859	229,375	171,5666	171,248	171,853
11	217,584	170,277	310,086	171,7819	171,441	173,029
12	191,062	170,950	247,366	**171,4039**	171,075	171,726
13	214,863	170,260	282,979	171,5757	171,153	172,523
14	199,774	170,191	260,211	171,5608	171,024	172,958
15	212,655	**170,188**	446,248	171,5395	**170,81**	172,816



(a)



(b)

Figure 4.11 : Résultats de l'évaluation des modèles de langages obtenus avec différent nombre d'individus par tournois. (a) présente les résultats de l'optimisation sans amorçage tandis que (b) présente les résultats de l'optimisation avec amorçage.

4.4.3.2 Taux de croisement

La reproduction des individus se fait par croisement, un processus analogue à l'enjambement des chromosomes. C'est cette étape qui est responsable de l'exploration de l'espace de recherche par le biais de la création de nouveaux individus qui intègrent les paramètres de leurs parents et qui nous permettent (idéalement) de **diversifier** la population.

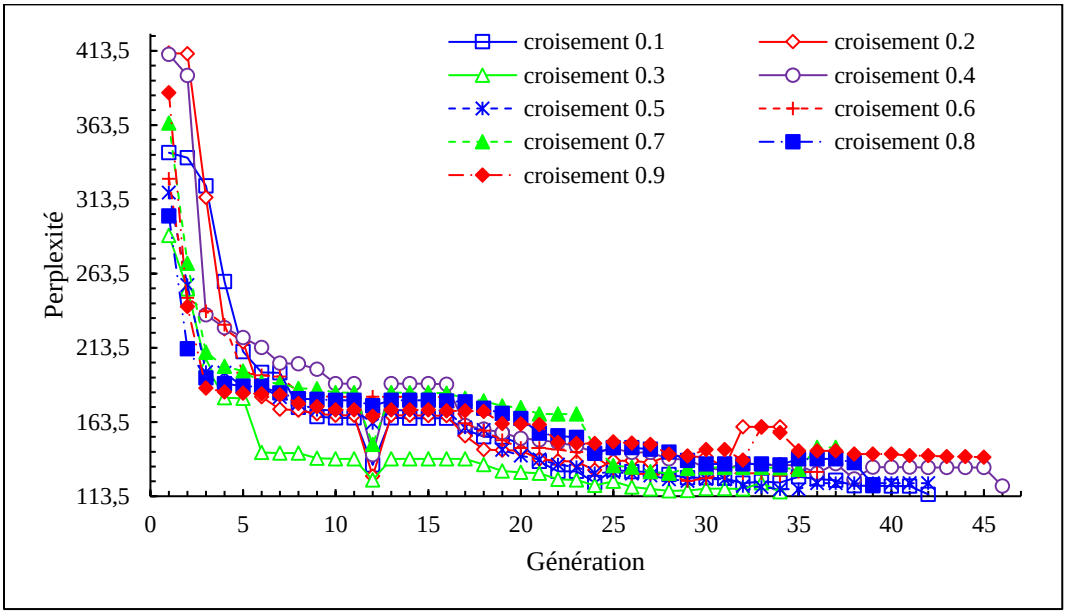
Dans cette section, nous évaluons l'effet du taux de croisement sur la performance des modèles de langage obtenus. Le taux de croisement représente la proportion d'individus qui seront le résultat d'une opération de croisement. La fonction de croisement utilisée pour cette évaluation est la fonction de croisement arithmétique. Le Tableau 4.3 résume les paramètres statiques utilisés pour cette évaluation.

La Figure 4.11 présente les résultats moyens de convergence obtenus avec les différents taux de croisement. Un taux de 30 % converge le plus rapidement et obtient les meilleurs résultats en termes de perplexité (voir Tableau 4.4) lors de l'optimisation sans amorçage. Cela est assez surprenant et semble indiquer qu'un faible taux de croisement est très bénéfique pour ce problème particulier d'optimisation.

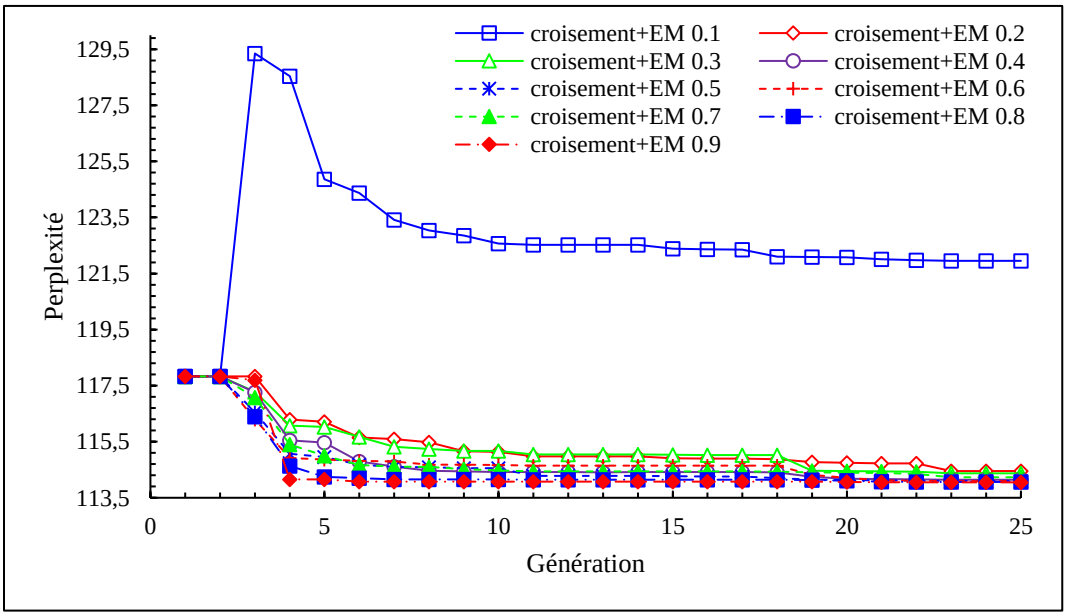
Tableau 4.3 : Paramètres fixes utilisés pour l'évaluation de l'effet du taux de croisement sur la performance finale du modèle de langage.

Paramètre	Description	Valeur
PopulationSize	Nombre d'individus	70
CreationFcn	Fonction de création d'individus	@gacreationuniform
MutationFcn	Fonction de mutation	@mutationadaptfeasible
CrossoverFcn	Fonction de croisement	@crossoverarithmetic
SelectionFcn	Fonction de sélection	@selectiontournament (tournoi à 6 individus)
Taux de mutation	Pourcentage des individus qui subiront une mutation	0,20
EliteCount	Nombre d'élites retenus	3
FitnessScalingFcn	Fonction de mise à l'échelle de la fonction de coût	@fitscalingrank
FunctionTolerance	Changement minimum constant considéré avant que l'algorithme termine.	0,005

Pour ce qui est de l'optimisation avec amorçage, les taux de croisement de 80 % et 90 % convergent le plus rapidement. Cela est assez cohérent avec le concept de l'amorçage où la population initiale reflète une solution adéquate, et un taux de croisement trop faible n'aura pas



(a)



(b)

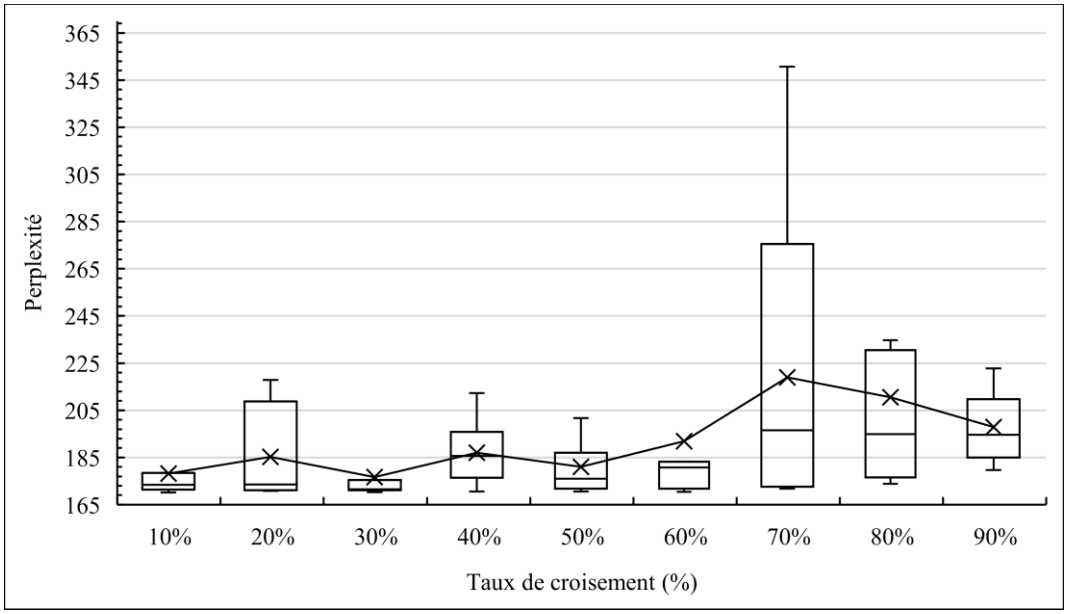
Figure 4.12 : Convergence moyenne suite à 25 simulations Monte-Carlo de l'optimisation par AG : (a) représente la convergence moyenne pour un taux de croisement de 10 % à 90 % sans amorçage. (b) représente la convergence moyenne pour un taux de croisement de 10 % à 90 % avec amorçage.

tendance à améliorer la performance des meilleurs individus, voir aura tendance à l'empirer. Cette observation se confirme avec un taux de croisement de 10 % où la solution obtenue est pire que la solution proposée lors de l'amorçage. Il est cependant important de noter, qu'avec amorçage, que la pire solution est obtenue avec un taux de croisement de 20 % (Figure 4.11 [b]) : la meilleure solution obtenue lors de l'optimisation ne donnera pas nécessairement lieu à la meilleure solution pratique. Pour ce qui est des autres taux de croisement, l'algorithme converge de façon beaucoup plus prévisible.

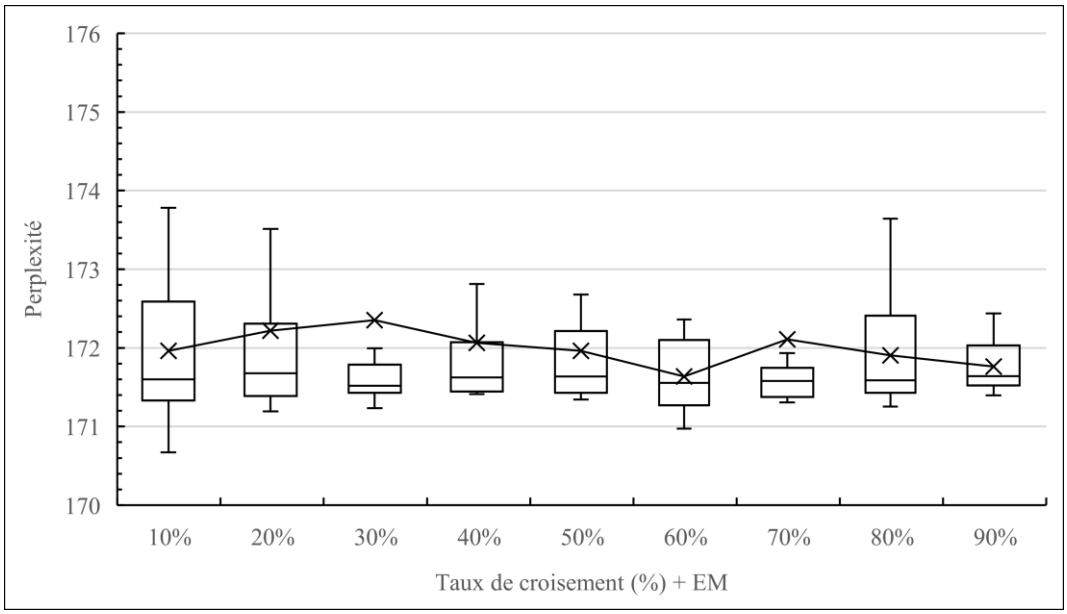
La Figure 4.13 et le Tableau 4.4 présentent les résultats de l'évaluation de la perplexité des modèles de langage obtenus selon les différents taux de croisements évalués. Comme ce fut le cas pour l'évaluation de la taille des groupes pour la sélection par tournoi, les résultats obtenus nous permettent difficilement de prédire les résultats attendus. Par contre, nous constatons tout de même une tendance un peu plus nette par rapport à l'évaluation de la fonction de sélection.

L'optimisation sans amorçage semble privilégier de faibles taux de croisement, avec une performance marquée à 10 % et 30 %. En effet, les résultats obtenus sont très stables avec un écart-type de 10,98, et 10,63 respectivement. Avec un taux de croisement de 70 %, nous observons un pic particulièrement difficile à expliquer étant donné que nous nous attendons à ce qu'un fort taux de croisement encourage la diversité et l'amélioration progressive de la population.

Les résultats obtenus avec amorçage ont un comportement inverse : les pires résultats sont obtenus avec de faibles taux de croisement et les meilleurs résultats sont observés à un taux de croisement de 60 %, 70 % et 90 %. Il est ici difficile de dire avec certitude lequel des taux de croisement est le plus favorable : un taux de 60 % produit les meilleurs résultats en moyenne, ainsi que le meilleur résultat absolu, tandis que les taux de 70 % et 90 % produisent des résultats beaucoup plus prévisibles.



(a)



(b)

Figure 4.13 : Résultats de l'évaluation des modèles de langage obtenus avec différent taux de croisement pour la fonction de croisement arithmétique. (a) présente les résultats de l'optimisation sans amorçage tandis que (b) présente les résultats de l'optimisation avec

Tableau 4.4 : Sommaire des résultats obtenus pour l'évaluation de l'effet du taux de croisement pour la fonction de croisement arithmétique avec et sans amorçage.

Taux de croisement	Perplexité (croisement)			Perplexité (croisement+EM)		
	Moyenne	Min	Max	Moyenne	Min	Max
10 %	178,154	**170,182**	200,4	172,003	170,994	174,667
20 %	185,256	170,802	217,835	173,048	171,194	181,42
30 %	**176,648**	170,261	**198,148**	172,669	171,233	181,547
40 %	187,018	170,543	212,299	172,065	171,413	175,21
50 %	181,061	170,505	201,754	171,962	171,342	174,114
60 %	191,941	170,414	294,583	**171,637**	**170,974**	**172,359**
70 %	219,010	171,794	350,714	172,107	171,305	177,217
80 %	210,518	173,85	331,942	171,904	171,254	173,645
90 %	197,845	179,712	222,759	171,761	171,395	172,44

4.4.3.3 Taux de mutation

Comme nous l'avons vu à la section 4.2.5, la mutation nous permet de diversifier notre population, et, par le fait même, d'explorer un espace de solution plus vaste. En principe, un taux de mutation trop faible risque de converger vers une solution locale, tandis qu'un taux de mutation trop élevé risque plutôt de nous empêcher de converger vers une solution convenable. C'est pourquoi il est important d'essayer d'atteindre un équilibre entre un taux trop faible ou un taux trop élevé de mutation, cela est particulièrement pertinent pour l'optimisation avec amorçage où la solution initiale proposée devrait être une solution optimale locale.

Le Tableau 4.5 présente un résumé des paramètres statiques utilisés pour évaluer l'influence du taux de mutation sur l'optimisation par algorithmes génétiques des poids d'interpolation du modèle de langage.

La fonction de mutation utilisée dans le cadre de ce travail est la fonction Matlab d'adaptation possible (*adapt feasible*). Cette fonction génère essentiellement un vecteur de directions aléatoires qui prennent en considération les limites et contraintes linéaires de la fonction à optimiser [109]. Elle vérifie par la suite si les valeurs résultantes donnent un résultat valide et satisfont aux limites

Tableau 4.5 : Paramètres statiques pour l'évaluation de la mutation.

Paramètre	Description	Valeur
PopulationSize	Nombre d'individus	70
CreationFcn	Fonction de création d'individus	@gacreationuniform
MutationFcn	Fonction de mutation	@mutationadaptfeasible
CrossoverFcn	Fonction de croisement	@crossoverarithmetic
SelectionFcn	Fonction de sélection	@selectiontournament (tournoi à 6 individus)
CrossoverFraction	Pourcentage des individus qui sont le fruit d'un croisement	0,8
EliteCount	Nombre d'élites retenus	3
FitnessScalingFcn	Fonction de mise à l'échelle de la fonction de coût	@fitscalingrank
FunctionTolerance	Changement minimum constant considéré avant que l'algorithme termine.	0,005

et contraintes. Une fois ces valeurs validées, la fonction génère un nouvel individu qui vera sa performance évaluée.

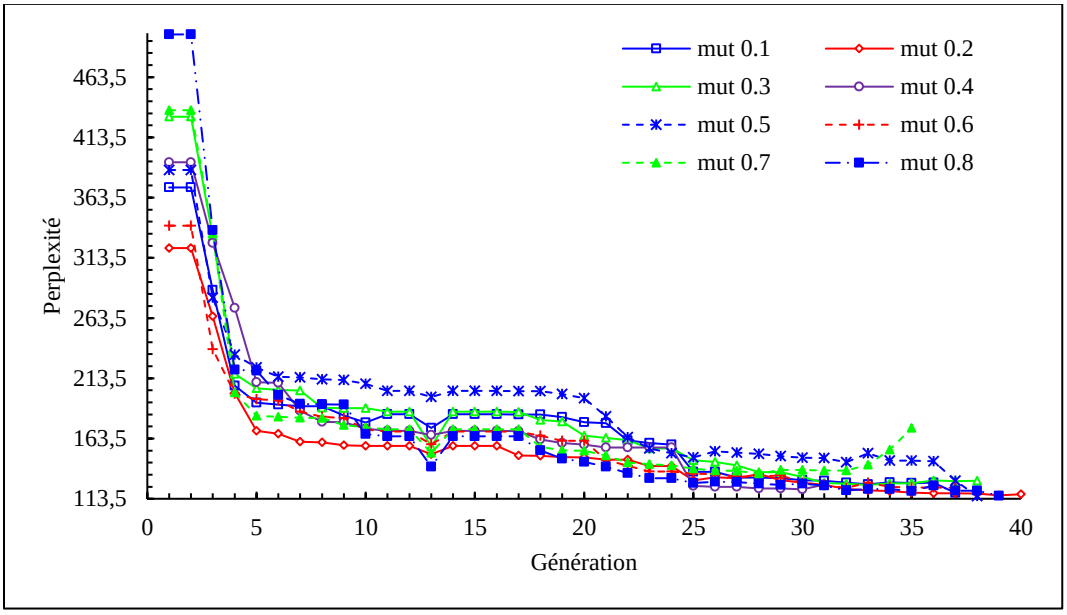
La Figure 4.13 présente les résultats moyens de convergence obtenus avec les différents taux de mutation. Outre les taux de mutation de 50 % et de 70 % qui ressortent du lot, la convergence s'effectue de façon assez similaire. Le taux de mutation de 50 % peine initialement à converger vers une solution optimale, mais débloque à partir de la 36^{e} génération. Par contre, le taux de mutation de 70 % semble problématique dans la mesure où il a tendance à diverger à partir de la 32^{e} génération. Ce comportement ne semble cependant pas se traduire à un taux de mutation de 80 % puisque celui-ci converge assez rapidement.

Pour l'optimisation avec amorçage, la convergence s'effectue de façon très similaire avec tous les taux de mutation, sauf 80 % où la solution moyenne proposée est supérieure à la moyenne.

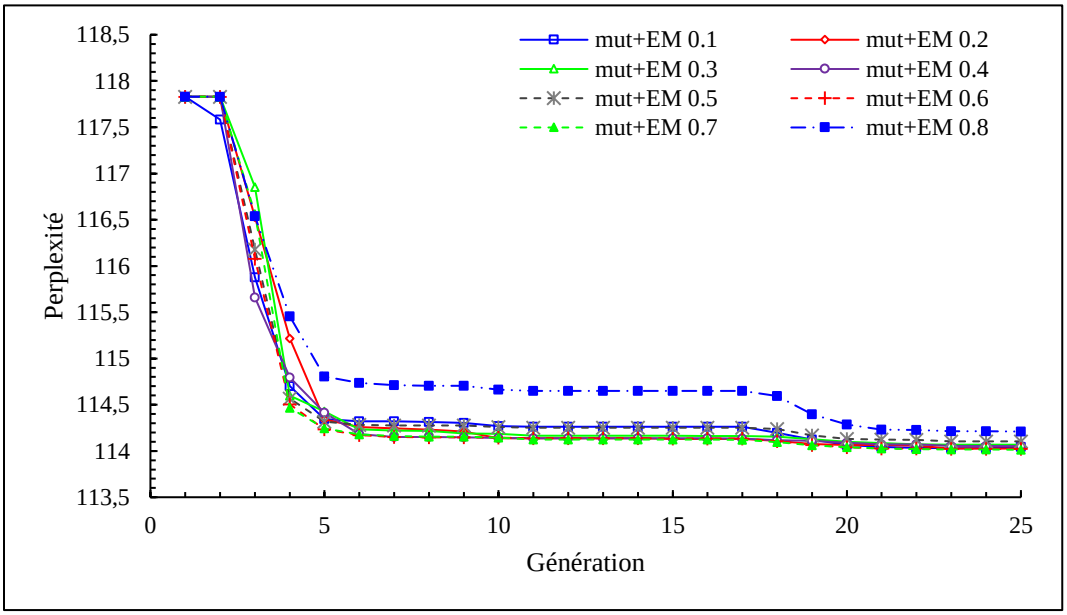
La Figure 4.14 présente les résultats de l'évaluation des modèles de langage sur le corpus d'évaluation. Pour le cas de l'optimisation sans amorçage, les résultats suivent à peu près la forme d'une cloche, avec des résultats qui s'améliorent à 20 %, mais se dégradent à 30 % pour ensuite

s'améliorer à partir de 60 %. Encore une fois, ces résultats nous semblent un peu contre-intuitifs : nous nous attendions à une dégradation de performance proportionnelle à l'augmentation du taux de mutation. Nous obtenons donc de meilleurs résultats entre 60 % et 80 %.

L'optimisation avec amorçage pour sa part produit des résultats plus ou moins stables avec une forte croissance à 80 % qui corroborent le comportement observé et ce à quoi nous nous attendions comme résultat.



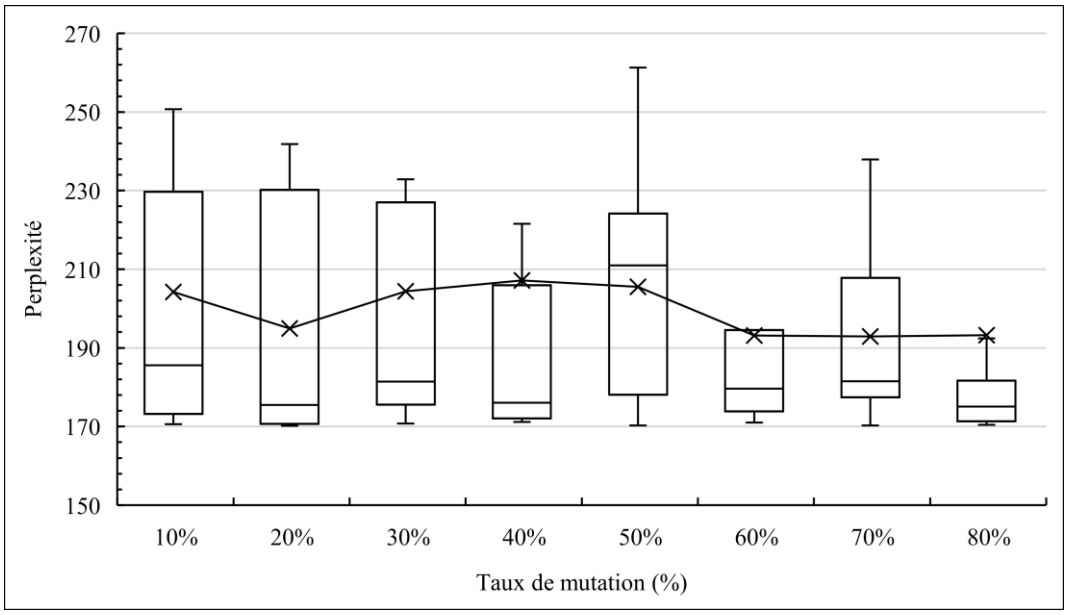
(a)



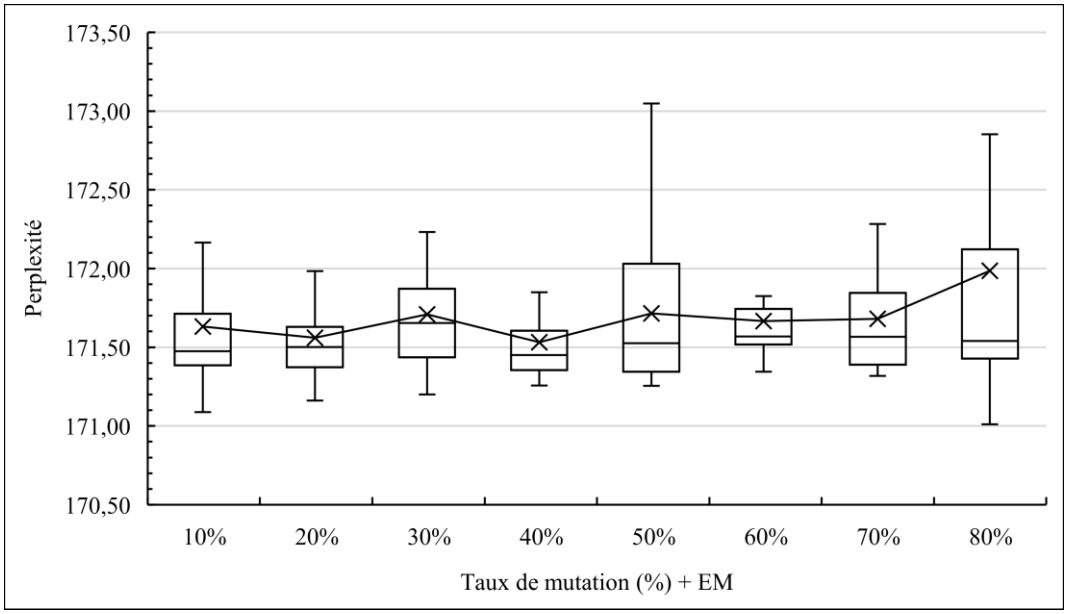
(b)

Figure 4.14 : Convergence moyenne suite à 25 simulations Monte-Carlo de l'optimisation par AG : (a) représente la convergence moyenne pour un taux de mutation de 10 % à 80 % sans amorçage. (b) représente la convergence moyenne pour un taux de mutation de 10 % à 80 % avec amorçage.

Finalement, le Tableau 4.6 présente un résumé des résultats obtenus. Les meilleurs résultats sont obtenus, en moyenne, avec un taux de mutation de 70 % pour l'optimisation sans amorçage, ce qui



(a)



(b)

Figure 4.15 : Résultats de l'évaluation des modèles de langage obtenus avec différent taux de croisement pour la fonction de croisement arithmétique.

correspond plus ou moins à ce que nous avons observé dans le Tableau 4.4. Les résultats de l'optimisation avec amorçage semble privilégier, en moyenne, un taux de mutation de 40 %.

Globalement, les résultats correspondent à ceux obtenus section 4.4.6 où nous avions utilisé un taux de mutation de 20 %.

Tableau 4.6 : Sommaire des résultats obtenus pour l'évaluation de l'effet du taux de mutation pour la fonction de mutation adaptative avec et sans amorçage.

Taux de mutation	Perplexité (mutation)			Perplexité (mutation+EM)		
	Moyenne	**Min**	**Max**	**Moyenne**	**Min**	**Max**
10 %	204,248	170,586	322,195	171,632	171,089	173,373
20 %	194,898	**170,195**	241,844	171,560	171,162	**172,31**
30 %	204,386	170,754	316,561	171,710	171,2	172,865
40 %	207,109	171,123	451,805	**171,532**	171,257	172,368
50 %	205,540	170,291	261,271	171,714	171,254	173,049
60 %	193,135	170,998	276,066	171,666	171,092	172,615
70 %	**192,919**	170,288	**237,890**	171,681	171,319	173,129
80 %	193,258	170,45	355,380	171,985	**171,01**	177,189

4.4.4 Paramètres de l'optimisation par algorithmes génétiques

Pour l'optimisation par algorithmes génétiques, nous utilisons l'utilitaire d'algorithmes génétiques de l'Optimization Toolbox de Matlab [115].

Les paramètres utilisés pour nos expériences ont été soit déduits empiriquement (fonction de sélection), soit en se basant sur la documentation fournie par MathWorks [109] et le guide disponible à la page 50 de Simon [108] et en utilisant notre expérience, paramètres qui sont très similaires à ceux que nous avons obtenus de façon empirique dans la section précédente.

Le Tableau 4.7 résume les paramètres que nous avons modifiés ainsi que leurs valeurs respectives.

Tableau 4.7 : Paramètres significatifs utilisés pour l'optimisation par algorithme génétique.

Paramètre	Description	Valeur
PopulationSize	Nombre d'individus	70
CreationFcn	Fonction de création d'individus	@gacreationuniform
MutationFcn	Fonction de mutation	@mutationadaptfeasible
CrossoverFcn	Fonction de croisement	@crossoverarithmetic
SelectionFcn	Fonction de sélection	@selectiontournament (tournoi à 6 individus)
CrossoverFraction	Pourcentage des individus qui sont le fruit d'un croisement	0,70
EliteCount	Nombre d'élites retenus	3
FitnessScalingFcn	Fonction de mise à l'échelle de la fonction de coût	@fitscalingrank
FunctionTolerance	Changement minimum constant considéré avant que l'algorithme termine.	0,001

4.4.5 Paramètres de l'optimisation par essaims particulaires

Pour l'optimisation par essaims particulaires, nous avons utilisé la librairie [116] et choisi nos paramètres (Tableau 4.8) selon la liste de choix de paramètres proposés dans le rapport technique de Pedersen [117]. Ceux-ci ont été obtenus de façon empirique et visent à permettre aux chercheurs de rapidement choisir des paramètres leur permettant d'obtenir rapidement de bons résultats.

Tableau 4.8 : Paramètres significatifs utilisés pour l'optimisation par essaims particulaires.

Paramètre	Valeur
AccelerationFcn	psoiterate
CognitiveAttraction ϕ_1	1,6319
SocialAttraction ϕ_2	0,6239
PopulationSize	70
StallGenLimit	10

4.4.6 Fonction objective

Nous utilisons comme fonction objective, pour les deux approches d'optimisation, AG et PSO, la mesure de perplexité du modèle de langage par rapport à un corpus de développement : un corpus, souvent tiré d'un plus large corpus, strictement utilisé pour l'optimisation des paramètres.

Rappelons que selon l'équation (2,58), la perplexité d'un texte $T = w_1, \ldots, w_N$ se mesure comme suit :

$$PP(T) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|h)}}. \tag{4.8}$$

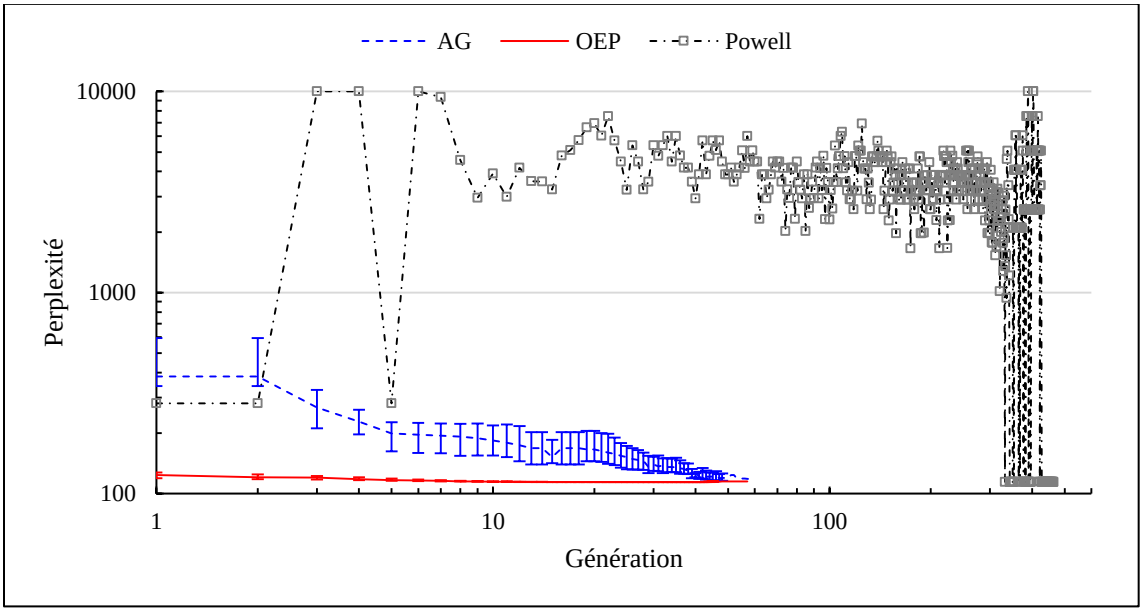
Le choix de la perplexité comme fonction objective est justifié par son faible coût computationnel comparativement à une évaluation complète utilisant le score obtenu par la RAP.

4.4.7 Résultats

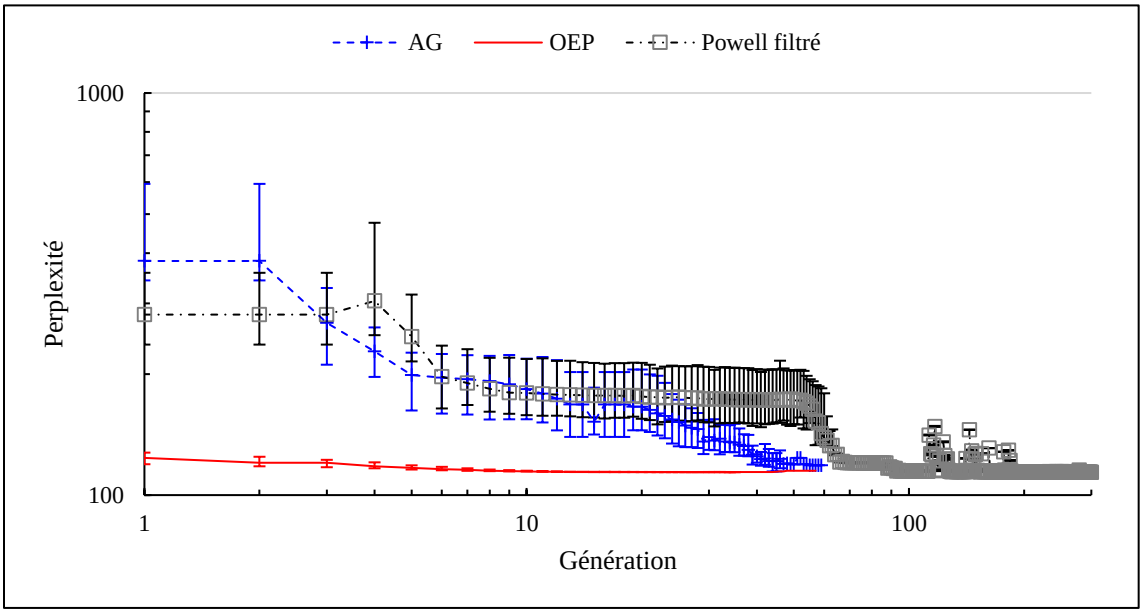
4.4.7.1 Optimisation sans amorçage

Nos résultats, illustrés par la Figure 4.16 et la Figure 4.17, démontrent que l'optimisation par essaims particulaires converge vers une solution potentielle beaucoup plus rapidement en moyenne que l'optimisation par algorithmes génétiques. Nous observons aussi que l'optimisation par la méthode de Powell explore l'espace de recherche de façon beaucoup plus erratique. Notons par ailleurs que nous avons filtré les figures 4.15 (b) et 4.16 (b) afin d'éliminer le bruit causé par les solutions hors bornes proposées par la méthode de Powell.

Plus spécifiquement, la Figure 4.17 nous permet de constater les grandes différences en termes de déviation standard entre les différentes méthodes d'optimisation. L'OEP converge en moyenne de façon beaucoup plus stable que l'optimisation par AG et par méthode de Powell. Par contre, bien que l'optimisation par AG produise une moins grande déviation standard en moyenne que la méthode de Powell lors de la phase d'optimisation, nous verrons plus tard, qu'en moyenne, que l'AG converge vers des solutions moins optimales que les deux autres algorithmes d'optimisation.



(a)

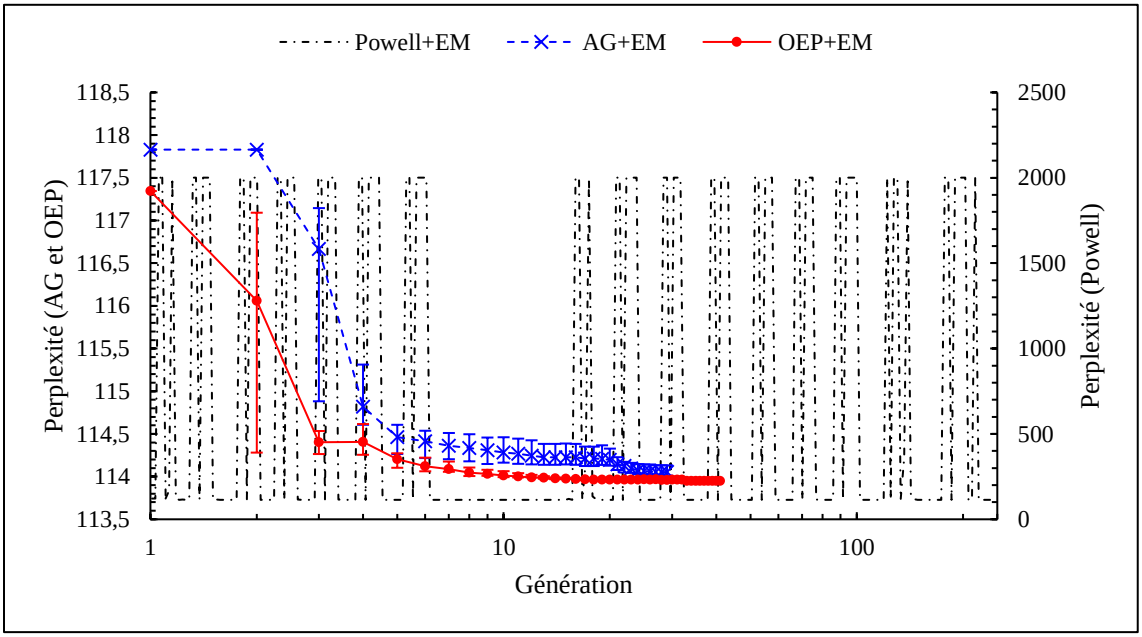


(b)

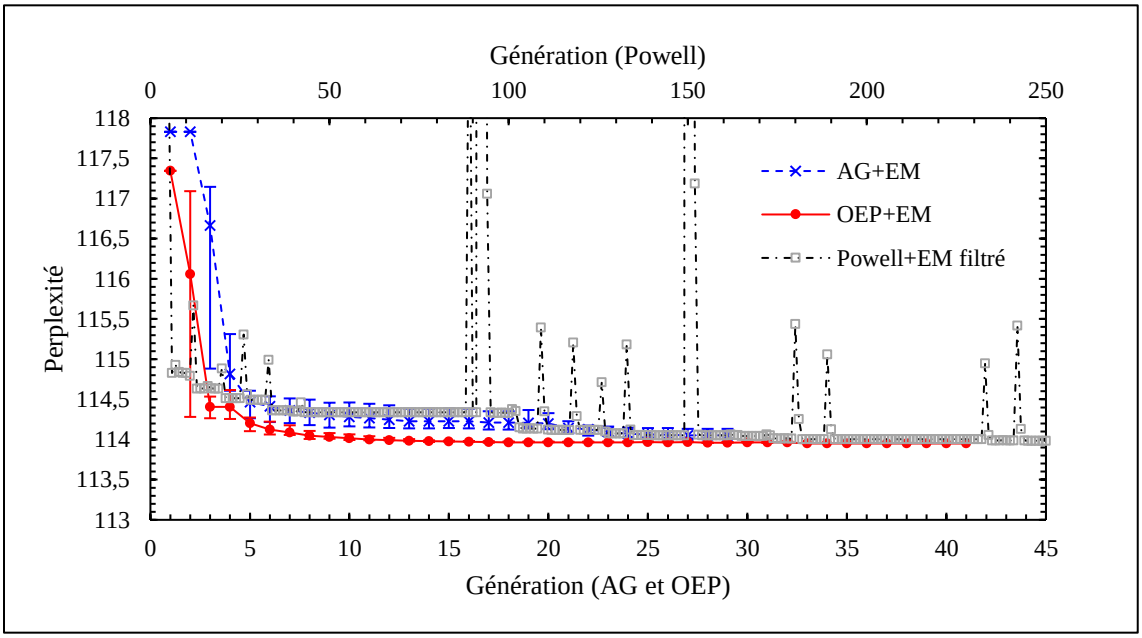
Figure 4.16 : Convergence moyenne sur 55 simulations de Monte-Carlo de la méthode de Powell vers une solution et où les barres verticales représentent la déviation standard. (a) illustre la convergence originale tandis que (b) filtre les solutions hors normes obtenues par l'algorithme de Powell.

La méthode de Powell quant à elle, mise en évidence dans la Figure 4.16, nécessite en moyenne 450 itérations avant de converger vers une solution. Nous constatons d'ailleurs que nous n'obtenons pas de solutions potentiellement correctes avant environ 350 itérations. La courbe très « bruitée » de l'optimisation moyenne de la méthode de Powell résulte de la façon dont

l'algorithme est mis en œuvre dans SciPy [118] et qui ne lui permet pas de prendre compte des bornes pour les poids d'interpolation ainsi que des contraintes. Nous pénalisons donc sévèrement



(a)



(b)

Figure 4.17 : Convergence moyenne sur 55 simulations de Monte-Carlo des différents algorithmes d'optimisations avec amorçage sur le corpus de développement. (a) illustre la convergence originale tandis que (b) filtre les solutions hors normes obtenues par l'algorithme de Powell. Les barres verticales indiquent la déviation standard.

toute solution proposée où les poids d'interpolation $\lambda_i < 0$ ou $\lambda_i > 1$ ou encore $\sum \lambda_i > 1$. Malgré cette limitation, nous avons tout de même opté pour l'outil SciPy en raison de sa très grande adoption par la communauté scientifique due à sa robustesse, sa flexibilité, ses fonctionnalités et sa rapidité. En effet, SciPy est une couche logicielle mince permettant de faire appel à plusieurs librairies scientifiques implémentées en Fortran et en C qui ont fait leurs preuves au fil du temps.

4.4.7.2 Optimisation avec amorçage par Maximisation de l'espérance

L'initialisation de la population peut avoir un impact majeur sur la capacité des algorithmes évolutionnaires à rapidement converger vers une solution et qu'elle soit optimale. Une technique populaire dans le contexte de l'optimisation de systèmes complexes est l'utilisation de connaissances expertes. Cette technique consiste à faire appel à un expert de domaine et utiliser ses connaissances du domaine pour initialiser la population. Nous simulerons cet expert en générant un individu qui représente une solution convenable obtenue par l'algorithme d'optimisation espérance-maximisation.

Nous avons utilisé la suite logicielle SRILM [93] avec sa commande `compute-best-mix` pour obtenir un point de départ informatif. Cette commande utilise l'algorithme espérance-maximisation pour calculer les poids d'interpolation optimaux des modèles de langage. L'avantage principal de l'algorithme espérance-maximisation est sa rapidité d'exécution, mais au prix d'une solution généralement sous-optimale.

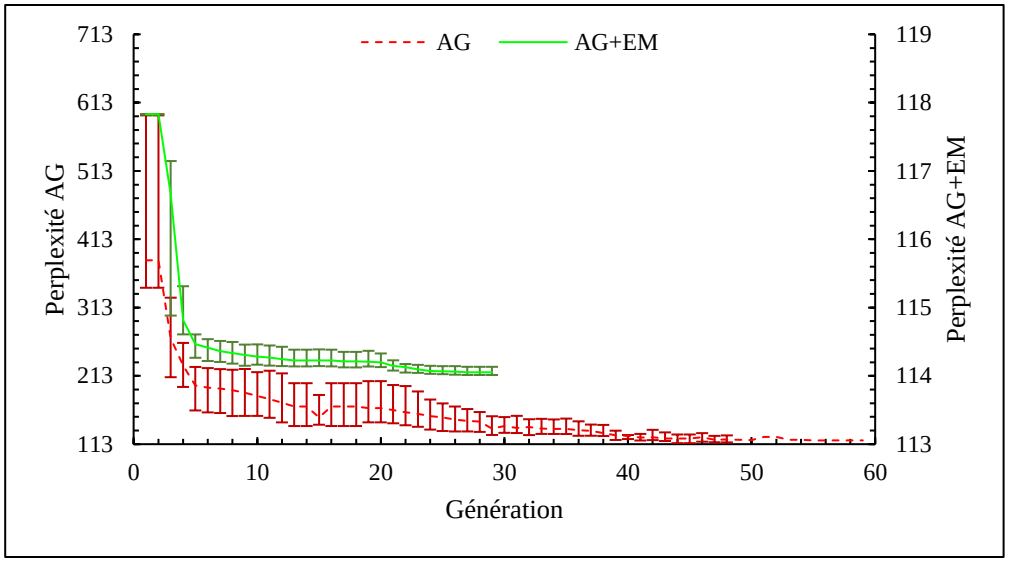
Nous avons donc procédé à une première optimisation des poids d'interpolation à l'aide de l'algorithme espérance-maximisation et utilisé les poids obtenus comme point de départ (amorçage) pour nos trois algorithmes d'optimisation.

La Figure 4.17 présente les résultats moyens obtenus avec amorçage pour chaque algorithme d'optimisation sur 55 simulations de Monte-Carlo, à l'exception de la méthode de Powell qui elle n'a été exécutée qu'une seule fois dû à sa nature heuristique.

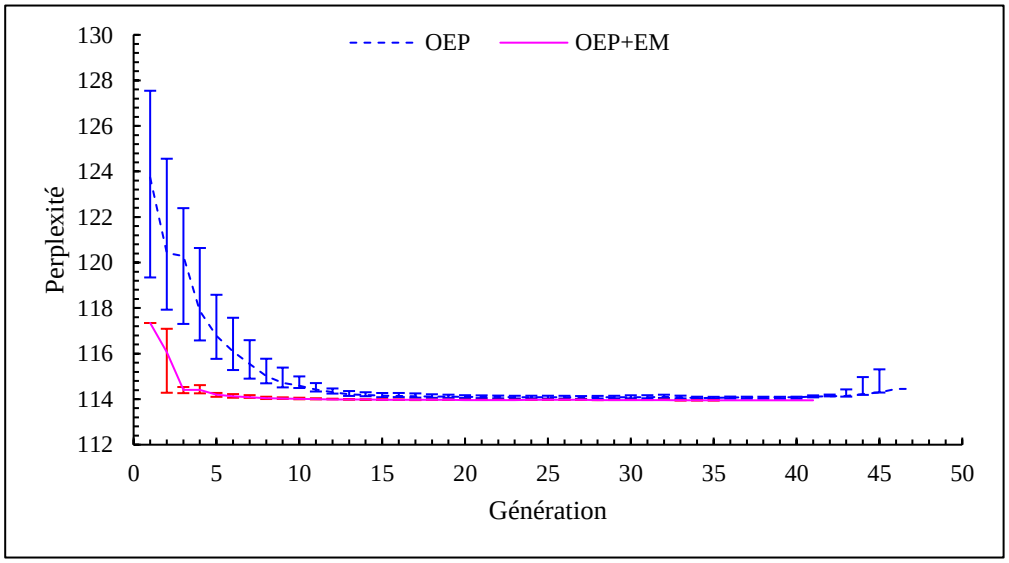
La Figure 4.17 quant à elle offre un portrait comparatif de l'optimisation des algorithmes évolutionnaires avec et sans amorçage.

La méthode de Powell voit elle aussi sa vitesse de convergence grandement améliorée. Nous pouvons maintenant obtenir une solution en approximativement 100 générations. Par ailleurs,

comme la méthode de Powell réalise des minimisations unidimensionnelles en suivant des directions conjuguées, nous aurons toujours le même résultat à partir d'un même point de départ puisque la tangente calculée sera toujours la même. Cela est dû au fait que l'on varie seulement une variable à la fois et que l'on garde seulement la valeur qui a eu le plus d'impact positif sur la solution. Cela explique aussi pourquoi nous n'avons pas à effectuer de simulations de Monte-Carlo



(a)



(b)

Figure 4.18 : Comparaison de la vitesse de convergence entre (a) l'optimisation par algorithme génétique avec et sans amorçage (b) l'optimisation par essaims particulaires avec et sans amorçage.

pour la méthode de Powell avec amorçage. Nous avons tout de même choisi de l'inclure à la Figure 4.17 comme référence.

L'algorithme génétique converge lui aussi beaucoup plus rapidement : en moyenne en 30 générations contre 59 en moyenne sans amorçage. Nous verrons aussi au Tableau 4.9 que la perplexité de la solution moyenne des AG est grandement réduite en termes de perplexité et d'écart-type.

Finalement, comme nous avons pu le constater par les Figures 4.16 et 4.17, l'algorithme d'OEP converge aussi plus rapidement : 41 générations en moyenne comparativement à 47 générations en moyenne sans amorçage.

Nous croyons que la stratégie de recherche de l'optimisation par essaims particulaires lui permet de garantir un niveau de robustesse beaucoup plus élevé par rapport aux variations du point de départ que l'algorithme génétique qui privilégie essentiellement une exploration (large) de l'espace de recherche basé sur les mutations.

4.4.7.3 Évaluation de la perplexité des modèles de langage

Dans cette section nous présentons les résultats de l'évaluation des modèles de langage obtenus par chaque algorithme. Les modèles de langage ont été évalués sur le corpus d'évaluation du *Wall Street Journal-based Continuous Speech Recognition (CSR) Corpus Phase II* (WSJ1) [92]. Les ensembles utilisés pour l'évaluation sont les suivants :

- Hub 1 (base de référence WSJ lue),
- Spoke 1 (adaptation des modèles de langage),
- Spoke 2 (indépendance du domaine)
- Spoke 9 (diction spontanée WSJ)

Cela nous donne un total de 23 000 mots dans 1285 phrases sans filtrage des phrases contenant des mots hors vocabulaire. Le vocabulaire (fermé) utilisé est le vocabulaire WSJ standard de 20 000 mots. Le taux de mots hors vocabulaire est de 5,64 %.

Comme la Figure 4.19 et le Tableau 4.9 l'illustrent, l'algorithme d'optimisation qui performe le moins bien en moyenne est l'algorithme génétique sans amorçage. Par contre l'AG obtient le meilleur score, soit une perplexité de 170.141 sur le corpus d'évaluation. En contrepartie, c'est

Tableau 4.9 : Sommaire des résultats de l'évaluation des modèles de langage optimisés selon les différents algorithmes d'optimisation et évalués sur l'ensemble de test WSJ1 Hub 1 Spoke 1, 2 et 9.

Algorithme d'optimisation	Perplexité		
	Moyenne	Min	Max
WB	323,26	-	-
KN	232,047	-	-
AG	194,587	**170,141**	303,549
AG + EM	171,650	170,904	175,271
OEP	171,596	171,331	173,536
OEP + EM	171,411	171,296	171,852
Powell	171,767	171,279	175,153
Powell + EM	**171,399**	171,399	**171,399**

aussi l'algorithme génétique qui nous donne la solution la moins performante : une perplexité maximale de 303,549. Lorsque l'algorithme génétique est combiné avec une phase d'amorçage, nous obtenons en moyenne des solutions équivalentes aux autres algorithmes d'optimisation.

Outre l'algorithme génétique sans amorçage qui sort de l'ordinaire, les autres algorithmes d'optimisation convergent en moyenne vers des solutions très similaires, avec ou sans amorçage.

L'avantage principal de l'implémentation de l'amorçage basé sur l'algorithme espérance-maximisation est une réduction considérable du temps moyen de convergence de chaque algorithme d'optimisation ainsi qu'une réduction de la variation des solutions proposées.

Ces résultats sont cohérents avec ceux de la littérature : les algorithmes (stochastiques) d'optimisation auront tendance à converger vers une solution optimale plus rapidement. Par ailleurs, lorsque l'on intègre des connaissances spécifiques aux problèmes dans nos algorithmes de recherche, nous pouvons nous attendre à obtenir de meilleurs résultats en moyenne. Finalement, comme nous cherchons non seulement à obtenir une solution optimale, mais aussi une solution optimale rapidement, l'utilisation de l'amorçage devient une étape cruciale.

Nous calculons la vitesse de convergence des différents algorithmes d'optimisation en nous basant sur le nombre total d'appels de la fonction objective. Cette méthode de calcul a été privilégiée suite

au profilage des différents algorithmes d'optimisation et où nous avons observé qu'environ 99% du temps d'exécution est représenté par le calcul de la fonction objective. La méthode de Powell sans amorçage avec une moyenne de 3500 appels (500 itérations * 7 appels par itération) avant l'atteinte de notre condition d'arrêt est la plus lente. L'algorithme d'OEP demande en moyenne au moins 2870 appels, ce qui le rend près de 20 % plus rapide que la méthode de Powell. Par ailleurs, contrairement à la méthode de Powell qui est plus séquentielle par sa nature, l'OEP, et l'AG peuvent faire appel à un parallélisme plus fin. Le plancher s'établit à un processus par individu versus 1 processus par variable par itération. Pour notre ordinateur d'évaluation, qui compte 24 cœurs logiques, cela revient à environ 120 appels totaux par processeur pour l'optimisation par essaims particulaires. Nous obtenons alors une solution beaucoup plus rapidement avec l'optimisation par essaims particulaires. L'AG avec amorçage demandera quant à lui en moyenne un total de 88 appels par processeur (environ 2100 appels de fonction), ce qui est encore plus rapide que la méthode de Powell puisque nous pouvons obtenir un niveau supérieur de parallélisme avec l'AG[32].

Un des principaux avantages des algorithmes évolutionnaires d'optimisation est qu'ils permettent de réduire le nombre de bacs à interpoler parallèlement à la recherche d'une solution optimale (réduction de la perplexité). La Figure 4.19 présente un portrait global de la capacité de chaque algorithme à réduire le nombre de bacs.

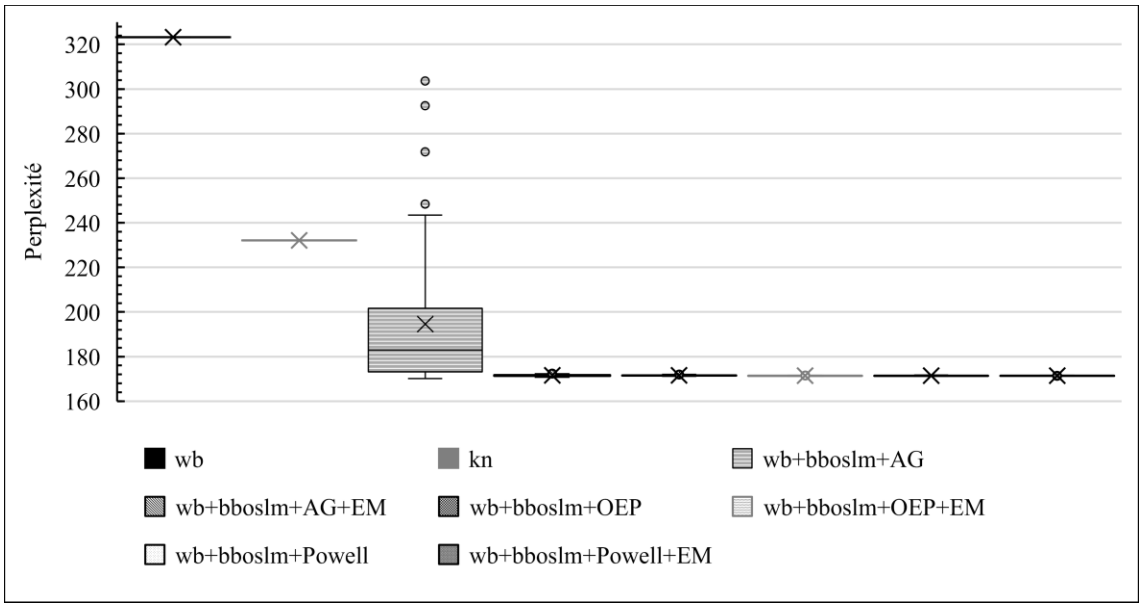
On remarque que l'AG propose une plus grande variété de solutions potentielles et qui réduisent le plus le nombre de bacs en moyenne (5 bacs en moyenne). C'est aussi l'algorithme qui propose le plus petit nombre de bacs à interpoler.

L'OEP propose en moyenne des solutions comptant entre 5 et 6 bacs à interpoler et certaines solutions proposées par l'algorithme d'optimisation peuvent réduire jusqu'à 4 le nombre de bacs à interpoler.

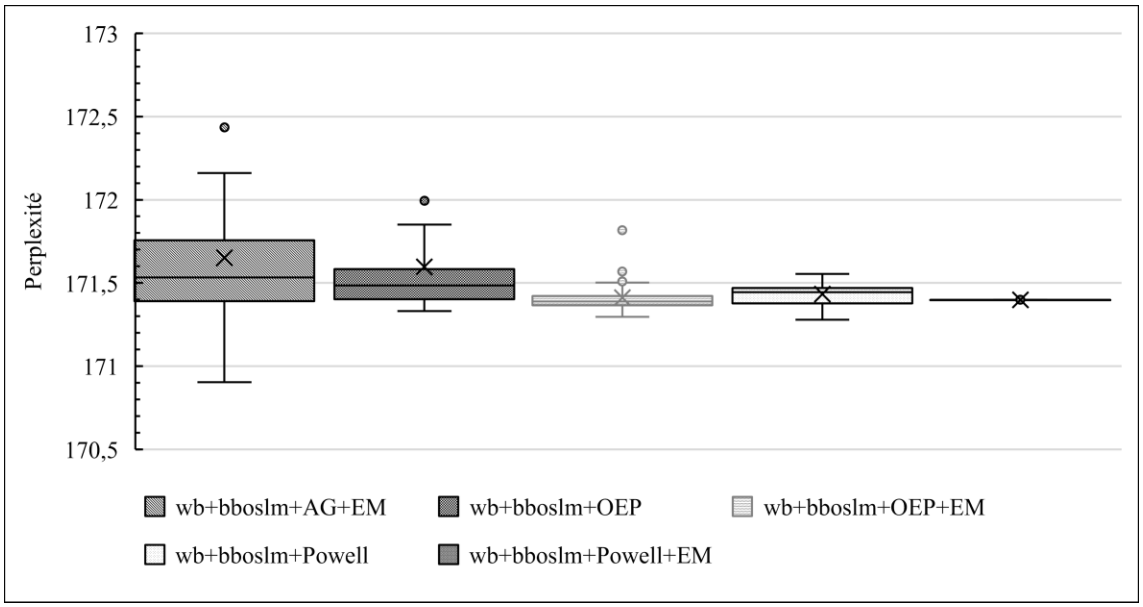
Lorsque la phase d'amorçage est implémentée, les deux algorithmes évolutionnaires obtiennent en moyenne la même réduction de bacs : entre 6 et 7 avec un minimum de 4 bacs pour l'optimisation

[32] Notons que le nombre maximal de fils concurrents pour la méthode de Powell correspond au nombre de paramètres à optimiser. Pour les algorithmes évolutionnaires, le nombre maximal de fils concurrents (outre le parallélisme attribuable à la fonction objective choisie) correspond au nombre d'individus dans la population.

par essaims particulaires. Nous pensons que cela s'explique par le fait que point de départ soit beaucoup plus près de la solution optimale, ce qui réduit l'espace de recherche exploré.



(a)



(b)

Figure 4.19 : Résultats de l'évaluation des modèles de langage optimisés selon les différents algorithmes d'optimisation et évalués sur l'ensemble de tests WSJ1 Hub 1 Spoke 1, 2 et 9. La ligne représente la médiane, le marqueur (X) la moyenne et les barres verticales représentent la variabilité empirique des valeurs en dehors des quartiles. Les points représentent les valeurs hors-norme.

La Figure 4.20 présente la relation observée entre le nombre de bacs interpolés avec le modèle de langage principal et la perplexité du modèle de langage optimisé.

Nous constatons encore une très grande variance pour l'AG. L'algorithme génétique est néanmoins en mesure de proposer une solution presque optimale avec quatre bacs (perplexité de 173.147). Les meilleurs résultats en moyenne sont obtenus avec 5 bacs. Cette figure nous permet d'ailleurs de constater que, contrairement aux autres algorithmes d'optimisation, l'AG converge de façon très variable et que l'algorithme produit des résultats qui sont faiblement et inversement corrélés au nombre de bacs (coefficient de corrélation $r = -0.196\,53$).

L'optimisation par essaims particulaires converge de façon beaucoup moins variable. Nous constatons une plus grande variation (proportionnellement) avec 4 bacs. Les résultats obtenus sont aussi plus (inversement) corrélés au nombre de bacs avec $r = -0.294\,41$ et un coefficient de détermination $R^2 = 0.221\,05$. Nous avons donc une faible corrélation qui est inverse au nombre de bacs : un nombre de bacs croissant aura un faible impact positif sur la perplexité du modèle de langage.

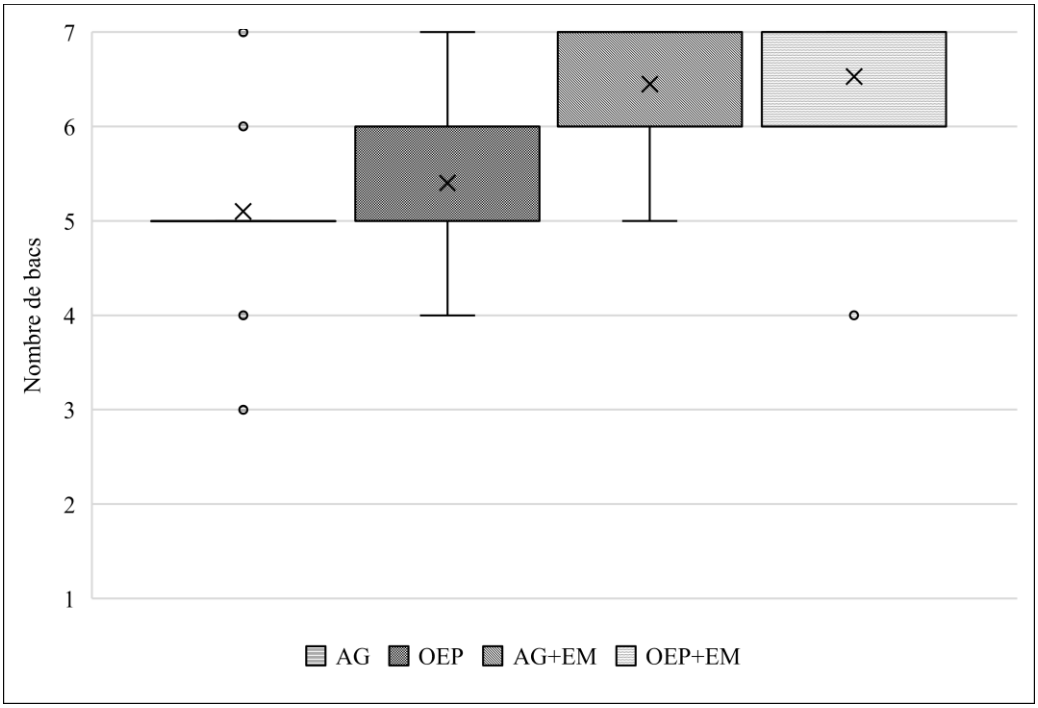


Figure 4.20 : Sommaire du nombre de bacs à interpoler obtenus par les différents algorithmes d'optimisation.

Tableau 4.10 : Taux d'observation des bacs retenus pour les solutions obtenues par les différents algorithmes d'optimisation.

	d1	d2	d3	d4	d5	d6	d7
GA	1	0,864	0,661	0,186	0,729	0,814	0,847
GA+EM	1	1	0,964	0,554	1,000	0,964	0,964
OEP	0,947	0,947	0,789	0,263	0,789	0,842	0,825
OEP+EM	1	0,982	0,964	0,655	0,982	0,982	0,964

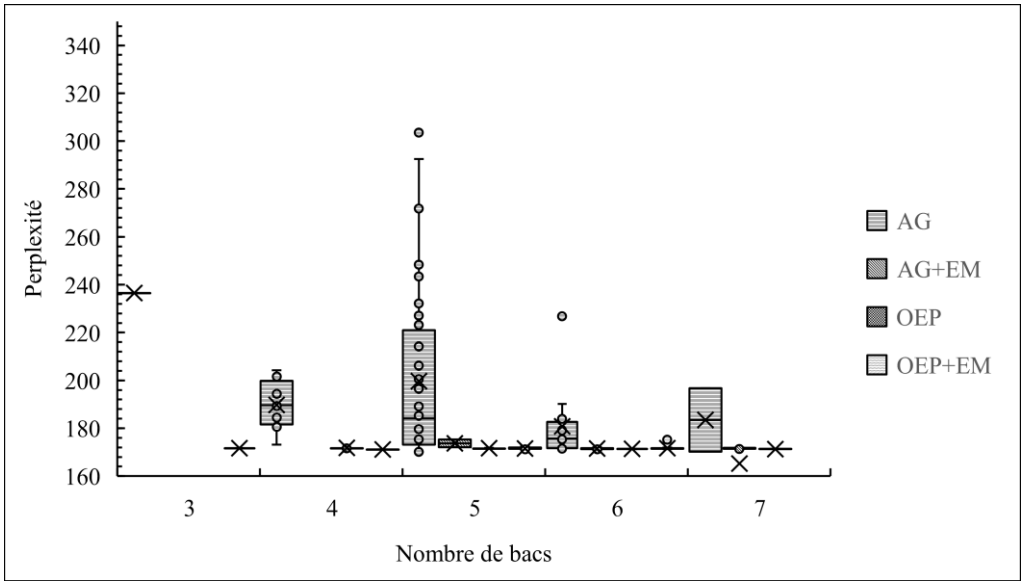
Par ailleurs, globalement, les bacs les plus retenus (voir Tableau 4.10) sont d1 98,7 %, d2 94,8 %, d6 90 %, d7 90 %, d5 87,5 %, d3 84,5 % et d4 41,4 %. Il apparait donc que les bacs d'une distance ontologique supérieure à 5 contribuent de façon significative à la performance obtenue du modèle de langage final.

Tel qu'observé à la Figure 4.19, l'amorçage réduit de façon très significative la variance de l'algorithme génétique. Son coefficient de corrélation passe à $r = -0{,}297\,38$. Le nombre de bacs a un impact légèrement plus important, mais il demeure néanmoins négligeable.

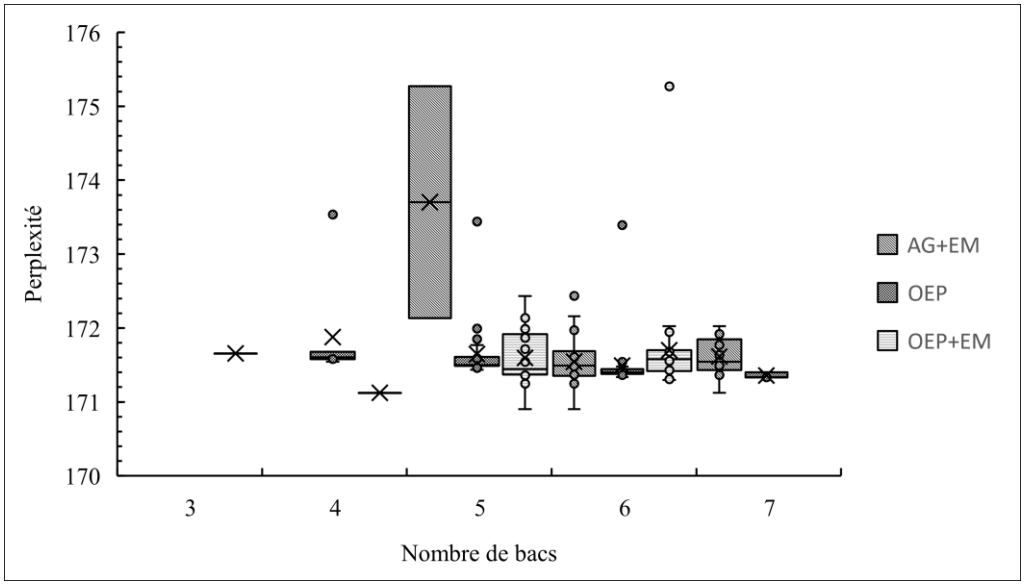
L'OEP voit principalement son coefficient de corrélation passer à $r = 0{,}047\,74$. L'amorçage fait donc en sorte que le nombre de bacs influence encore moins la solution finale obtenue.

Finalement, la Figure 4.21 nous permet d'observer pourquoi l'algorithme génétique sans amorçage obtient de moins bons résultats que les autres algorithmes d'optimisations. C'est l'algorithme qui couvre la plus grande surface dans l'espace de solution. Les résultats sont très fortement inversement corrélés au poids d'interpolation du modèle de langage primaire λ_0 : $r = -0{,}883\,34$ et $R^2 = 0{,}7803$. L'AG obtient de meilleurs résultats lorsque la solution proposée converge vers $0{,}6 < \lambda_0 < 0{,}7$, ce qui est consistant avec les résultats obtenus par les autres algorithmes d'optimisation et qui sont représentés par les points sur l'axe horizontal supérieur.

L'amorçage permet à l'AG d'intégrer des connaissances spécifiques aux problèmes dès la première génération, ce qui a pour effet de considérablement réduire sa dispersion dans l'espace de recherche. La tendance de l'AG à couvrir un plus grand espace de recherche lui permet d'ailleurs d'obtenir des solutions légèrement supérieures aux autres algorithmes d'optimisation. Nous observons maintenant une forte corrélation entre la valeur λ_0 et la perplexité du modèle de langage avec un coefficient $r = 0{,}895\,45$. De plus, la variation de perplexité pour des poids λ_0 très



(a)



(b)

Figure 4.21 : Relation entre le nombre de bacs à interpoler obtenus par chaque algorithme et la perplexité telle qu'évaluée sur le corpus d'évaluation. (a) montre les résultats complets tandis que (b) montre les résultats sans l'algorithme génétique afin de mieux observer le comportement des autres algorithmes d'optimisation.

similaires illustre bien que la perplexité soit aussi influencée par les poids d'interpolation des autres bacs ainsi que les bacs qui sont retenus par l'algorithme d'optimisation.

L'optimisation par essaims particulaires obtient des résultats qui sont beaucoup plus centrés autour de $\lambda_0 = 0{,}658$. Comme les solutions obtenues par l'algorithme génétique, les solutions de l'OEP

sont aussi fortement corrélées à λ_0 avec un coefficient de corrélation $r = 0{,}801\,16$. Nous notons cependant les données aberrantes aux points $(0{,}665\,812\,;\,173{,}859)$ et $(0{,}667\,268\,;\,173{,}558)$ qui illustrent la corrélation linéaire moyenne entre λ_0 et la perplexité du modèle de langage obtenu.

L'amorçage ressert encore plus les résultats obtenus par l'OEP. Un autre effet de l'amorçage est la réduction de l'influence du poids d'interpolation λ_0 sur la perplexité des modèles de langage obtenus. Le coefficient de corrélation passe à $r = 0{,}568\,85$, une faible corrélation linéaire positive. La performance de la solution obtenue est donc plus dépendante ($R^2_{\lambda_0} = 0{,}323\,59$) des autres facteurs : λ_{1+}, choix des bacs.

Globalement, les meilleurs résultats sont obtenus entre $0{,}63 < \lambda_0 < 0{,}675$ avec une tendance croissante de la perplexité à partir de $\lambda_0 > 0{,}65$. Notons d'ailleurs que le meilleur résultat obtenu, une perplexité de 170,149, à l'aide de l'algorithme génétique a été obtenu avec $\lambda_0 = 0{,}574\,19$. Cela semble indiquer que d'autres variations des différents algorithmes, ou des paramètres différents, seraient potentiellement en mesure d'obtenir, en moyenne, des résultats supérieurs à ceux que nous avons obtenus.

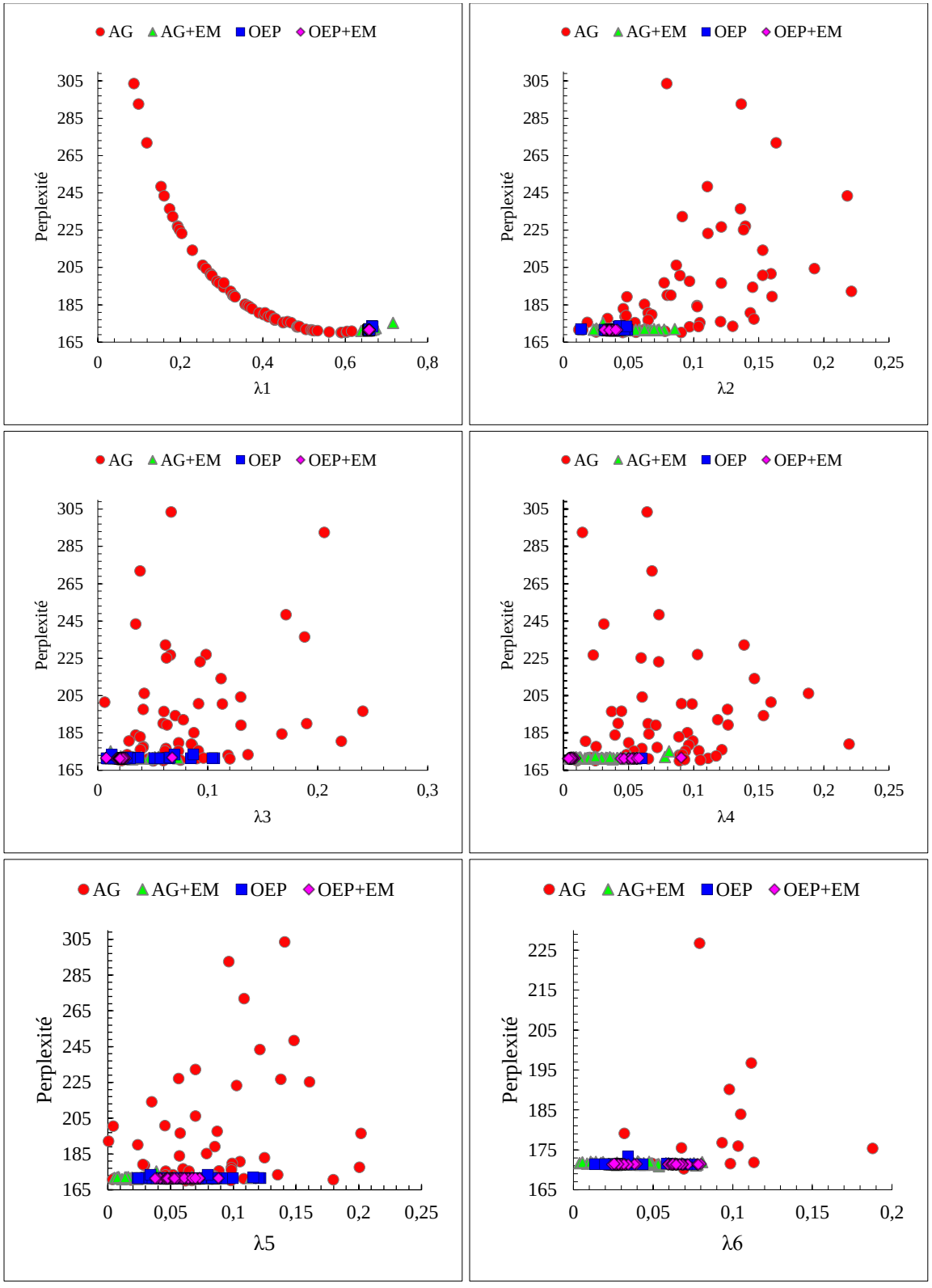


Figure 4.22 : Relation entre le poids d'interpolation λ et la perplexité des modèles de langage obtenus lors de l'optimisation.

4.5 Évaluation du taux d'erreur des mots

Tel que décrit à la section 4.4.7.3 ci-dessus, le corpus utilisé pour l'entrainement des modèles acoustiques et des modèles de langage et pour l'évaluation est le corpus *Wall Street Journal-based Continuous Speech Recognition Phase II*.

Le même dictionnaire que celui utilisé lors de l'évaluation de la perplexité, soit le dictionnaire WSJ à vocabulaire clos de 20 000 mots (taux de mots hors vocabulaire de 5,64 %), est utilisé pour l'évaluation.

La version 3.4.1 du logiciel HTK [119] a été utilisée pour la partie reconnaissance automatique de la parole.

Le taux d'erreur des mots (TEM), calculé selon l'équation (4.9), est utilisé pour comparer la performance des différents modèles de langage.

$$TEM = \frac{O + S + I}{N} \tag{4.9}$$

où O est le nombre d'omissions,
S est le nombre de substitutions,
I est le nombre d'insertions, et
N est le nombre de mots dans la phrase originale.

L'entrainement des modèles acoustiques a été réalisé en suivant le protocole d'apprentissage proposé par Vertanen [120] avec une phase d'entrainement discriminatif additionnelle (critère *Minimum Phone Error*). Nos modèles acoustiques sont entrainés sur l'ensemble complet d'entrainement du corpus WSJ1 et utilisent l'ensemble de 40 phonèmes du dictionnaire CMU [121]. Les modèles obtenus pour le décodage acoustique sont des triphones « extra-mot » (*cross-word*) à état liés (*tied-state*) à 32 gaussiennes par état et 64 pour le modèle de silence. Ces modèles sont représentés par un vecteur 39-dimensionnel de Coefficients Cepstraux à Fréquence de Mel (*Mel Frequency Cepstrum Coefficient*) avec l'énergie, éq. (4.10), du signal, le delta, éq. (4.11), (« dynamique » du signal) et l'accélération, éq. (4.12) :

$$E = \log \sum_{n=1}^{N} s_n^2 \tag{4.10}$$

$$d_t = \frac{\sum_{\theta=1}^{\Theta}(c_{t+\theta} - c_{t-\theta})}{2\sum_{\theta=1}^{\Theta}\theta^2} \tag{4.11}$$

$$a_t = \frac{\sum_{\theta=1}^{\Theta}(d_{t+\theta} - d_{t-\theta})}{2\sum_{\theta=1}^{\Theta}\theta^2} \tag{4.12}$$

où $\{s_n, n = 1, N\}$ représente le *n*ième échantillon de la parole,

c_t représente un coefficient statique de la parole à l'instant t,

d_t est le coefficient delta à l'instant t,

a_t est le coefficient d'accélération à l'instant t,

et Θ est la taille de la fenêtre et où $\Theta = 3$ dans notre cas.

L'outil HDecode [119] est utilisé pour l'évaluation de la RAP avec les paramètres suivants pour la largeur du faisceau d'élagage (*pruning beam width*), de la pénalité d'insertion de mot et de l'échelle du modèle de langage[33] : 220, -4 et 15, sont utilisés pour l'évaluation de la RAP.

Comme l'évaluation par décodage complet de la RAP est une opération particulièrement lente, nous avons d'abord procédé à un « rescoring[34] » (réassignation des poids de transition, ou renotation) d'un réseau des 100 énoncés les plus probables générés à l'aide d'un bigramme lissé à l'aide de l'algorithme de Witten-Bell afin d'identifier les meilleurs modèles obtenus par les différents algorithmes d'optimisation.

Comme l'illustre le Tableau 4.11, les meilleurs résultats (préliminaires) obtenus par *rescoring* sont, en moyenne, obtenus à l'aide de l'optimisation par l'algorithme génétique. Pour leur part, les autres algorithmes d'optimisation produisent en moyenne des résultats équivalents et légèrement supérieurs à ceux obtenus par le modèle de langage lissé à l'aide de la méthode Kneser-Ney. Nous soulignerons aussi le fort taux d'erreur des phrases, frôlant le 100 %, qui résulte du haut taux

[33] Le détail des paramètres et de leur signification sont documentés dans [122].

[34] Nous privilégierons l'utilisation du terme *rescoring* dans ce document puisque le terme est plus succinct que sa traduction et capture mieux, à notre avis, le concept.

Tableau 4.11 : Moyennes des résultats, exprimés en pourcentages, obtenus lors de l'évaluation de la performance de la reconnaissance automatique de la parole des modèles de langage par « rescoring » pour les divers algorithmes d'optimisation comparés aux trigrammes de bases lissées à l'aide des algorithmes Witten-Bell et Kneser-Ney.

Modèle	Corr.	Sub.	Supp.	Ins.	Err	Err Ph.
WB	49,90	40,70	9,40	6,10	56,30	99,70
KN	**50,80**	39,40	9,70	5,80	55,00	99,70
Powell	50,28	39,55	10,19	5,40	55,11	99,70
Powell + EM	50,28	39,55	10,19	5,40	55,11	99,70
AG	49,95	**38,89**	**11,15**	**4,65**	**54,69**	**99,58**
AG + EM	50,25	39,54	10,20	5,40	55,13	99,70
OEP	50,24	39,54	10,23	5,38	55,13	99,70
OEP + EM	50,21	39,60	10,20	5,40	55,16	99,70

d'erreur des mots obtenus et qui fait en sorte que presque toutes les phrases reconnues contiennent au moins une erreur.

Par ailleurs, comme observée dans la section précédente et illustrée par la Figure 4.23, l'optimisation par algorithme génétique produit les résultats ayant la plus grande variance, tandis que les algorithmes d'optimisation par essaims particulaires et par l'algorithme de Powell obtiennent des solutions beaucoup plus stables.

Nous avons par la suite procédé à une évaluation de la corrélation entre la perplexité du modèle de langage et sa performance en termes de taux d'erreur de mots. La Figure 4.24 présente la relation entre la perplexité des modèles de langage obtenus par les différents algorithmes d'optimisation et le TEM. Puisque nous avions précédemment procédé à l'évaluation de l'effet des divers paramètres d'optimisation pour les algorithmes génétiques, nous avons choisi d'inclure tous les résultats dans l'analyse. Cela nous permet de bien caractériser la relation potentielle entre la perplexité et le TEM.

Comme nous pouvons l'observer pour l'algorithme génétique, une augmentation de la perplexité du modèle de langage influence faiblement (coefficient de corrélation de 0,037) de façon négative le taux d'erreur des mots. La tendance est relativement plus marquée pour l'algorithme génétique avec amorçage avec un coefficient de corrélation de 0,288. La tendance s'inverse pour

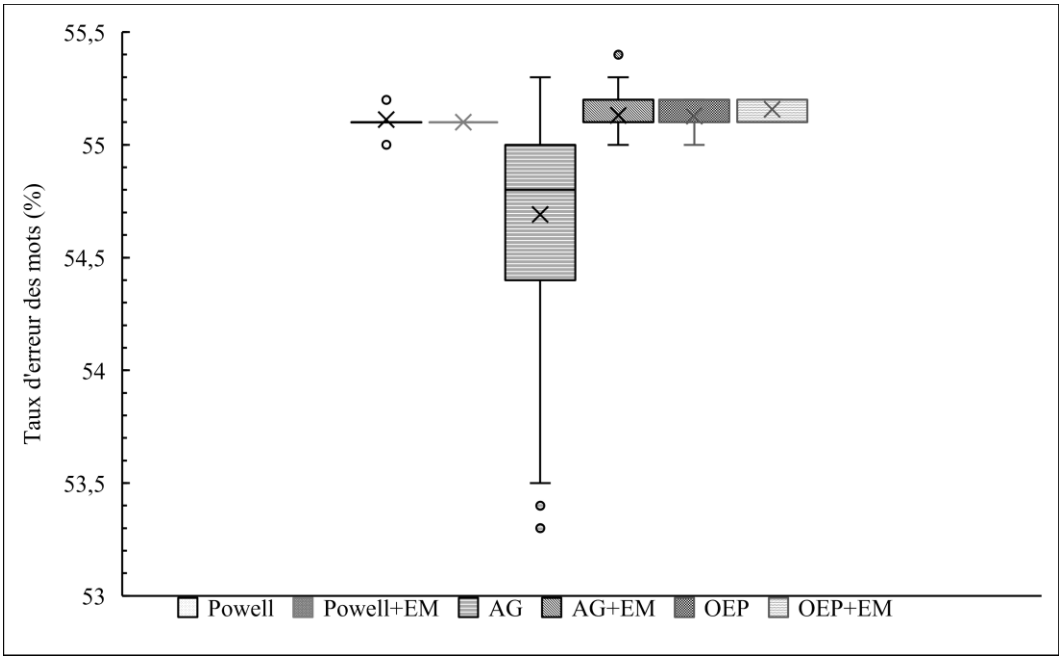


Figure 4.23. Résultats de l'évaluation des modèles de langage optimisés selon les différents algorithmes d'optimisation et évalués sur l'ensemble de test WSJ1 Hub 1, Spoke 1, 2 et 9. La ligne représente la médiane, le marqueur (X) la moyenne et les barres verticales représentent la variabilité empirique des valeurs en dehors des quartiles. Les points représentent les valeurs hors norme.

l'optimisation par essaims particulaires où nous constatons plutôt une tendance à l'amélioration du taux d'erreur lorsque la perplexité augmente (coefficient de corrélation de -0,296). Nous devons toutefois être prudent dans notre interprétation de ces résultats puisque l'AG avec amorçage et l'OEP sont beaucoup plus circoncis dans leur variation de perplexité que l'algorithme génétique sans amorçage. Le portrait est similaire pour le cas de l'OEP avec amorçage (coefficient de corrélation de -0,318 9) où la variation de perplexité est encore plus bornée. L'optimisation par algorithme de Powell poursuit la même tendance avec un coefficient de corrélation de -0,198. Finalement, lorsque nous analysons le portrait global, nous pouvons tout de même constater une faible tendance, un peu contre-intuitive, de l'amélioration du TEM proportionnelle à une augmentation de la perplexité (coefficient de corrélation de -0,264). Nous sommes d'avis que ce résultat un peu contre-intuitif est potentiellement dû au fait que nous avons procédé à une évaluation de la reconnaissance automatique de la parole par *rescoring* du réseau des N-meilleurs résultats plutôt que par décodage complet.

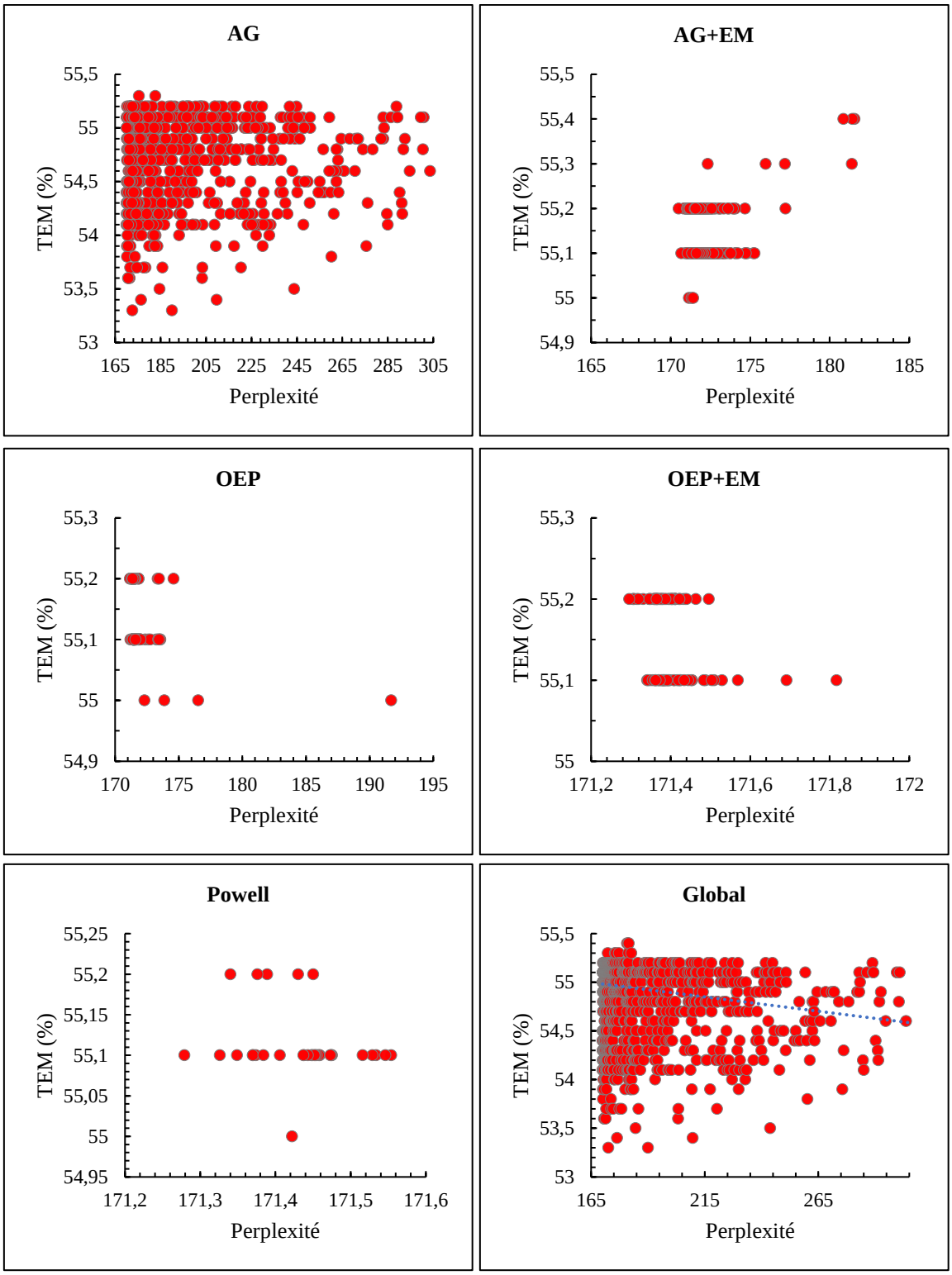


Figure 4.24 : Taux d'erreur des mots versus perplexité. Notons que nous avons inclus, dans la sous-figure globale, la ligne de tendance linéaire du système.

Après avoir identifié les meilleures solutions proposées par chaque algorithme, nous avons procédé à la reconnaissance automatique de la parole complète (*full decoding*). Les résultats obtenus sont résumés dans le Tableau 4.12 : . Nous pouvons constater que les résultats obtenus par les différents algorithmes d'optimisation s'équivalent en termes de performance. Cependant, comme nous l'avons expliqué précédemment, l'avantage des algorithmes évolutionnaires réside dans leur rapidité (parallélisme) et leur capacité à réduire le nombre de bacs pour l'interpolation (diminution de la complexité) tout en maintenant une précision équivalente à l'algorithme de Powell. Nous notons une diminution relative moyenne d'environ 14 % du taux d'erreur des mots ainsi qu'une augmentation relative d'environ 4,2 % du taux de phrases correctement reconnues.

Afin de déterminer si les améliorations obtenues sont statistiquement significatives, nous avons procédé à l'analyse de la variance des différents modèles de langage obtenus. Ces résultats sont résumés dans le Tableau 4.13 et ont été obtenus grâce à l'outil SCTK [123]. Comme nous pouvons le constater, tous les algorithmes d'optimisation ont produit des modèles de langage avec lissage ontologique ayant une performance supérieure ($p < 0.001$) aux modèles de base lissés à l'aide des algorithmes de Witten-Bell et Kneser-Ney respectivement. Nous pouvons donc conclure que pour ce problème particulier d'optimisation qu'il n'y a pas de différence significative de performance entre les solutions proposées par l'algorithme de Powell et les algorithmes évolutionnaires. Cependant, les algorithmes évolutionnaires nous permettent plus facilement de réduire la complexité de la solution proposée que l'algorithme de Powell.

Tableau 4.12 : Résumé des résultats obtenus par les divers modèles de langage suite au décodage complet.

Modèle	Corr.	Sub.	Supp.	Ins.	Err	Err Ph.
WB	75,4	19,4	5,2	3,1	27,7	77,3
KN	74,2	19,3	6,5	2,5	28,3	77,5
Powell	**78,6**	**16,6**	**4,8**	2,5	23,9	**76,2**
Powell + EM	**78,6**	**16,6**	**4,8**	2,5	23,9	**76,2**
AG	78,4	16,7	**4,8**	**2,4**	24	76,3
AG + EM	**78,6**	**16,6**	**4,8**	2,5	**23,8**	**76,2**
OEP	**78,6**	**16,6**	**4,8**	**2,4**	**23,8**	76,3
OEP + EM	**78,6**	**16,6**	**4,8**	**2,4**	**23,8**	76,3

Tableau 4.13 : Comparaison statistique selon l'analyse de la variance de la performance des différents modèles de langage. Le symbole ~ indique que les modèles s'équivalent statistiquement.

Modèle	KN	AG	OEP	AG+EM	OEP+EM	POW	POW+EM
WB	~ $p < 0{,}001$	AG $p < 0{,}001$	OEP $p < 0{,}001$	AG + EM $p < 0{,}001$	OEP + EM $p < 0{,}001$	POW $p < 0{,}001$	POW $p < 0{,}001$
KN		AG $p < 0{,}001$	OEP $p < 0{,}001$	AG + EM $p < 0{,}001$	OEP + EM $p < 0{,}001$	POW $p < 0{,}001$	POW $p < 0{,}001$
AG			~ $p < 0{,}001$	~ $p < 0{,}001$	~ $p < 0{,}001$	~ $p < 0{,}001$	~ $p < 0{,}001$
OEP				~ $p < 0{,}001$	~ $p < 0{,}001$	~ $p < 0{,}001$	~ $p < 0{,}001$
AG+EM					~ $p < 0{,}001$	~ $p < 0{,}001$	~ $p < 0{,}001$
OEP+EM						~ $p < 0{,}001$	~ $p < 0{,}001$
POW							~ $p < 0{,}001$

Le Tableau 4.15 présente un résumé de la perplexité des modèles de langage qui ont obtenu les meilleurs résultats lors de l'évaluation de la RAP. Nous pouvons constater que tous les modèles de langage ontologiques optimisés ont une perplexité inférieure aux modèles de références.

Finalement, le Tableau 4.14 présente le taux de réduction du TEM observé pour une diminution d'un bit de l'entropie du modèle de langage par rapport au modèle de base lissé à l'aide de l'algorithme Witten-Bell. Nous obtenons donc une réduction moyenne du taux d'erreur des mots de 3,458 % par bit d'entropie croisée en moins du modèle de langage comparativement au modèle de base lissé par l'algorithme Witten-Bell et utilisé pour l'interpolation des bacs ontologiques. Ces résultats, bien que légèrement inférieurs à ceux obtenus par Goodman [78] (réduction d'environ 4,9 % par diminution de bit d'entropie croisée) demeurent tout de même significatifs, mais illustrent bien les limites et difficultés de la modélisation du langage pour l'amélioration de la reconnaissance automatique de la parole : puisque nous ne pouvons avoir des modèles de langage ayant seulement 1 bit d'entropie (ils seraient seulement valables pour le corpus de test ou

Tableau 4.15 : Récapitulatif des meilleurs résultats obtenus pour la reconnaissance automatique de la parole et la relation de ces résultats avec la perplexité et l'entropie croisée du modèle de langage obtenu.

	WB	KN	Powell	Powell + EM	AG	AG + EM	OEP	OEP + EM
TEM (%)	27,7	28,3	23,9	23,9	24	23,8	23,8	23,8
PPL	323,26	232,05	171,42	171,4	181,28	171,32	171,41	171,53
Entropie (bits)	8,34	7,86	7,42	7,42	7,50	7,42	7,42	7,42

Tableau 4.14 : Diminution obtenue du taux d'erreur des mots en % par réduction d'un bit d'entropie du modèle de langage.

	Powell	Powell + EM	AG	AG + EM	OEP	OEP + EM
Réduction TEM (%) par bit d'entropie croisée	3,478 %/bit	3,478 %/bit	3,088 %/bit	3,572 %/bit	3,569 %/bit	3,566 %/bit

d'entrainement), il est éventuellement nécessaire d'utiliser d'autres techniques pour réduire le taux d'erreur absolu (d'un signal acoustique « parfait ») comme la modélisation sémantique, pragmatique, logique, voire philosophique.

4.6 Discussion

Dans ce chapitre, nous avons comparé la performance de différents algorithmes dans le contexte de l'optimisation des poids d'interpolation des modèles de langage pour les *Bin-Based Ontologically Smoothed langage models* [86] et avons démontré que les algorithmes évolutionnaires sont en mesure de trouver des solutions optimales plus rapidement, tout en préservant la même précision, que la méthode de Powell qui est couramment utilisée pour l'optimisation des modèles de langage.

Nous avons démontré que la méthode de Powell, largement utilisée pour l'optimisation des modèles de langage, est à la fois en mesure d'obtenir des solutions satisfaisantes et est robuste face à l'initialisation aléatoire. L'amorçage, par l'initialisation de la « population » en utilisation des poids d'interpolation obtenus à l'aide de l'algorithme d'espérance-maximisation, lui a permis de converger plus rapidement vers une solution. Cependant, étant de nature séquentielle, cet algorithme n'est pas aussi rapide que les algorithmes évolutionnaires que nous pouvons agressivement paralléliser.

Nous avons aussi montré que l'optimisation par algorithme génétique, avec les paramètres de base utilisés, converge de façon imprévisible. Cependant, lorsque nous avons inclus l'amorçage, nous avons observé une diminution de 0,22 bit de l'entropie croisée moyenne. Cela a aussi permis à l'AG de converger beaucoup plus rapidement vers une solution optimale. C'est aussi l'algorithme qui a été en mesure d'obtenir les meilleures solutions lors de l'évaluation sur le corpus WSJ1.

Nous avons par ailleurs évalué l'effet des différents paramètres de l'algorithme d'optimisation par algorithmes génétiques. Nos résultats ont montré que les paramètres que nous avions choisis pour nos expériences étaient près des paramètres optimaux. Nous avons aussi observé que la modification du taux de mutation n'avait pas nécessairement le comportement auquel nous nous attendions, c'est-à-dire une dégradation des résultats proportionnelle à l'augmentation du taux de mutation. Finalement, nous avons pu observer que c'est le taux de croisement qui a le plus d'effet sur la performance globale de l'optimisation par algorithmes génétiques, avec le plus grand écart-type, autant pour l'optimisation sans amorçage que pour l'optimisation avec amorçage.

Nous avons aussi montré que l'optimisation par essaims particulaires est l'algorithme d'optimisation qui converge de façon la plus stable. C'est l'algorithme qui obtient les meilleurs résultats en moyenne et a une déviation standard inférieure à celle de l'algorithme génétique. L'OEP est aussi l'algorithme qui obtient une solution optimale le plus rapidement.

Par la suite nous avons procédé à une évaluation exhaustive de la performance des modèles de langage obtenus dans une tâche de reconnaissance automatique de la parole. Nous avons pu constater, dans notre cas particulier, que la perplexité ou l'entropie croisée du modèle de langage optimisé était faiblement corrélée au taux d'erreur des mots obtenue. Toutefois, les modèles les plus performants en reconnaissance automatique de la parole sont tout de même ceux qui diminuent de façon substantielle l'entropie du modèle de langage : diminution moyenne de 0,9 bit

d'entropie pour les modèles optimaux choisis par rapport au modèle de base lissé par l'algorithme de Witten-Bell.

Finalement, bien que lorsque l'on fournit un vecteur de solution résultant d'une optimisation par EM et que les algorithmes obtiennent en moyenne des solutions similaires, il est de notre avis, suite à nos observations, que l'optimisation par essaims particulaires est le plus performant des trois algorithmes d'optimisation évalués. L'OEP est plus rapide que la méthode de Powell et que l'AG, en particulier lorsque l'on introduit le parallélisme. De plus, contrairement à la méthode de Powell, les algorithmes évolutionnaires sont plus flexibles dans le sens où ils sont à la fois en mesure de minimiser le coût de la fonction et réduire le nombre de bacs utilisé, et ce, sans avoir recours à l'optimisation multiobjective.

Analyse sémantique pour systèmes de dialogue verbaux
Articles

5 REAL-LIFE SPEECH-ENABLED SYSTEM TO ENHANCE INTERACTION WITH RFID NETWORKS IN NOISY ENVIRONMENTS

Application d'un système basé sur la parole facilitant l'interaction avec des réseaux RFID dans des environnements bruités.

Y. Benahmed[1,2]*, S-A. Selouani*[2]*, D. O'Shaughnessy*[1] *et A. H. Abolhassani*[3]

[1]INRS-EMT, 800 de la Gauchetière O, H5A 1K6, Montréal, Qc, Canada
[2]LARIHS Lab. Université de Moncton, Campus de Shippagan, New Brunswick, Canada
[3]VMR Lab. McGill University, Montreal, Quebec, Qc, H3A 2A7, Canada

Article soumis à : *The 37th International Conference on Acoustics, Speech and Signal Processing*

Lieu de la conférence : Prague, République tchèque

Accepté et publié

Date de publication : Mai 2011

5.1 Contribution des auteurs :

Yacine Benahmed : (candidat)

Co-écriture de l'article, conception du logiciel de gestion par RFID. Conception et développement du gestionnaire de dialogue, expérimentation, analyse des données, préparation des graphes, des figures et des tableaux.

Douglas O'Shaughnessy : (directeur)

Conseils techniques et théoriques, conseils de rédaction, édition, coauteur, corrections.

Sid Ahmed Selouani : (codirecteur)

Conseils techniques et théoriques, édition, coauteur, corrections.

A. H. Abolhassani : (coauteur)

Co-écriture de l'article, conception et développement du module de débruitage (VRE-KLT), préparation et analyse des données résultats des expériences, préparation des tableaux.

5.2 Commentaires des évaluateurs

La majorité des commentaires des évaluateurs ont été pris en considération lors de la révision finale de la version de l'article présenté dans cette section.

5.2.1 Évaluateur no 1

General Comments to Authors

It would be good to highlight any novel ideas in the implementation of the AIML based interpreter, so the contribution in that area is more clear. Also, sufficiently distinguishing the new contribution in terms of the VRE criterion over the previous work (cited in paper) would help readers.

5.2.2 Évaluateur no 2

General Comments to Authors

This paper proposes to describe a speech-enabled RFID system that can operate in noisy environments. However, most of the effort was focused on their implementation of the KLT model, which was already discussed in ref. 7. So really the paper is about improving the recognition of the speech, rather than the entire system as indicated in the title/abstract.

5.2.3 Évaluateur no 3

General Comments to Authors

The speech-enabled system for RFID applications definitely has industry promise. The algorithm development seems heuristic and the proposal does not justify why the approach works better in factory environments. The system seems to work but no comparison to other approaches were provided to claim novelty.

5.3 Résumé

Cet article présente un système qui permet à l'utilisateur travaillant dans un environnement très bruyant d'interagir par la parole avec un réseau d'identification par radiofréquence (RFID). Un nouveau cadre de dialogue est proposé afin de donner aux opérateurs humains la capacité de communiquer avec le système d'une façon plus naturelle. Un des problèmes les plus courants attribués aux interfaces opérant sur le modèle de « commande et contrôle » est leur manque de flexibilité par rapport aux énoncés reconnaissables. Afin de rendre notre système plus flexible, nous avons choisi d'implémenter un gestionnaire de dialogue basé sur le langage AIML (« *Artificial Intelligence Markup Language* ») qui, à la base, a été conçu pour le développement d'agents conversationnels (visant le clavardage). Ce langage permet au système de répondre à des requêtes formulées en langage naturel par l'intermédiaire de détection de motifs (« *pattern matching* »). Cela a aussi comme avantage de rendre le système beaucoup plus robuste par rapport aux erreurs de reconnaissance automatique de la parole, ce qui est particulièrement problématique dans des environnements très bruyants. De plus, afin de faire face à des environnements très bruyants et changeants, une technique en ligne d'amélioration du sous-espace vectoriel du signal basé sur la variance de l'erreur de reconstruction de Karhunen-Loève Transform (VER-KLT) est proposée. Finalement, le caractère naturel du dialogue ainsi que l'efficacité de l'amélioration du signal de la parole sont évalués dans des situations de la vie réelle en utilisant le framework Microsoft Speech API. Nos résultats démontrent à la fois que l'algorithme VER-KLT apporte une amélioration de l'évaluation perceptuelle de la qualité de la parole pour les signaux très bruités par rapport aux algorithmes de références et que le système devient effectivement plus robuste face aux erreurs de reconnaissance de la parole par rapport à une grammaire statique.

Le texte qui suit est une copie exacte de l'acte de conférence référencé ici haut[35].

[35] La numérotation des sections, équations, figures, graphiques, tableaux et références ont été adaptés pour correspondre à l'ordre retrouvé dans ce document.

5.4 Abstract

This paper presents a system that allows the user to interact by speech with a Radio-Frequency IDentification (RFID) network working in a highly noisy environment. A new dialog framework is proposed in order to give the human operators the ability to communicate with the system in a more natural fashion. This is achieved by the implementation of the Artificial Intelligence Markup Language which, allows the system to respond to close-to natural language queries by means of pattern matching. Besides this, and in order to deal with highly noisy and changing environments, an online signal subspace enhancement technique based on the Variance of the Reconstruction Error of Karhunen-Loève Transform (VRE-KLT) is proposed. Both of the dialog naturalness and speech enhancement efficiency are evaluated in real-life situations by using the Microsoft Speech API framework.

Index Terms— Natural dialog, RFID, speech recognition, speech enhancement, KLT, AIML

5.5 Introduction

Speech-enabled interfaces that include both speech recognition and synthesis are sufficiently advanced to a point where it becomes feasible to design viable and desirable solutions in various data collection applications. One of these innovative applications consists of providing the speech modality to allow the human operators of a Radio Frequency IDentification (RFID) Network to communicate naturally with related devices and information systems. The advantages of speech hands-free and eyes-free systems combined with the RFID technology are expected to improve speed, ergonomics and safety of numerous operations such as order picking, quality management, tracking and monitoring of assets, and shipping operations. In order to maintain an acceptable level of accuracy, the use of speech recognition is however limited in widely used applications to a fairly short list of input keywords for control and commands. For instance in Microsoft's approach [124], numeric characters for voice data inputs are used to select the command. Increasingly, and in all types of tasks, users are now requiring more naturalness in their verbal communications with machines. Therefore, making the speech-enabled interfaces more natural is one of crucial issues for their deployment in real-life applications such RFID.

Most interfaces incorporating speech interaction fall into three broad categories. The first category includes Command and Control (C&C) interfaces that rely on a fixed task dependant grammar to provide user interaction [125]. Their main advantage is their ease of implementation and high command recognition rate. However, their downside is the high cognitive load they induce when users interact with the system because of its lack of flexibility and lack of uniform command sets. The second category is based on interactive voice response (IVR) that guides users by the means of prompts in order to validate the utterance at every step [126]. This style of interaction is mostly used in menu navigation such as that found with phone and cable companies. Its relative lack of efficiency for fast interaction makes it a poor choice for everyday use. Finally, the third category uses natural language processing to parse the user's utterance and to determine the goal of the request. This can be done through multiple ways such as semantic and language processing and [127]. Hence, to be effective, due to "limitless" vocabulary, this type of interface needs an accurate recognizer.

In this paper, we present an effective framework for interactive spoken dialog systems belonging to the third category, using a dynamic pattern matching technique based on Artificial Intelligence Markup Language (AIML) [128]. This system will be used by RFID operators in very noisy environments. To improve the performance of the speech recognition in severely degraded environments, an online signal subspace-based approach is used to dynamically optimizing the choice of the noise-free Karhunen-Love Transform (KLT) principal components.

The organization of the paper is as follows. Section 5.6 presents the overall System that combines RFID and speech technologies to improve enterprise operations and also describes the AIML-based dialog framework used in our application. Section 5.7 presents the proposed subspace approach in enhancement of the noisy signal. The performance evaluation is reported in section 5.8, and in section 5.9 the paper is concluded.

5.6 Speech-enabled RFID system

Typically, an RFID-based process involves a human operator to perform multiple and simultaneous tasks in order to effectively use a number of devices and input devices to update an information system. The workflow may include other additional tasks such as inspecting items received for damages, confirming the quantity of a given product, etc. To reduce the cognition load of those

operators and to improve their comfort, the sensor capabilities of RFID is nowadays a very promising solution. As illustrated in Figure 5.1, the proposed system enables the orchestration of RFID events and inputs into synchronized operations in order to provide more automation thanks to the speech modality. Data collection is performed through the RFID devices. The readers and tag communication is managed by the speech-enabled system. Operator can speak in order to modify power, select antenna, launch the Read/Write tags, check inventory, etc.

5.6.1 System details

The application is composed of two major components:

The first component is the speech input layer. Here speech utterances are captured in a buffer and passed either directly to the automatic speech recognition (ASR) system or to the noisy speech enhancer according to the user preferences. The N-best ASR recognition outputs are then fed to a dialog manager which, parses them and returns either a user-friendly message to the user to indicate that it did not understand the meaning of the utterance or the most likely command with its parameters to the business logic end. For example, if the system recognizes the user input “Show me

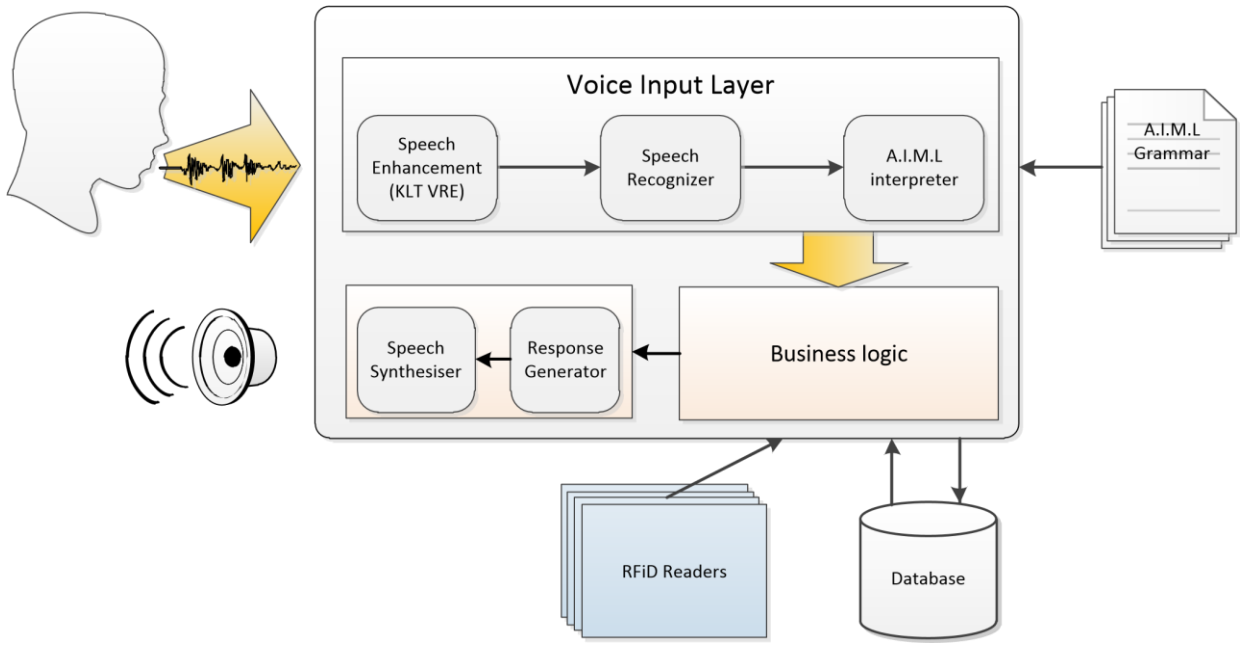


Figure 5.1 : The overall speech-enabled system interacting with RFID Network.

the RFID Reader Configuration", the system would return the RFID set up form. Feeding the N-best ASR outputs gives a certain error tolerance to the system since the most likely recognized sentence is not always the best one.

The second component is the business layer. It is constantly listening for user input as well as listening for RFID device events such as unauthorized inventory check out.

5.6.2 Dialog interpreter

The dialog interpreter used in this particular application is an Artificial Intelligence Markup Language (AIML) parser. AIML is an XML compliant language designed to create chat bots that can fool users into thinking that they are chatting with a person. Its primary design feature is minimalism in the sense that it can reduce complex user utterances into simpler atomic components. As such, it is essentially a pattern matching system that maps well with Case-Based Reasoning [128]. AIML consists of categories that encapsulate a pattern, the user input, and a template, the possible answer. The AIML parser then tries to match what the user said to the most likely pattern and outputs the corresponding answer. Additionally, patterns can include wildcards which, are especially useful for our application. It also supports recursion which, enables the system to answer based on previous inputs.

5.7 VRE speech enhancement technique

A key issue in developing a KLT-based speech enhancement model is to choose the optimal number of principal components (PCs). If fewer PCs are selected than required, a poor model will be obtained and an incomplete representation of the process results. On the contrary, if more PCs than necessary are selected, the model will be over-parameterized and will include noise. Different approaches had been proposed in the past to select the optimal number of PCs among which, we can cite the Minimum Description Length (MDL) [129] that used in our evaluation tests. Generally speaking, enhancement is obtained by removing the noise subspace and optimally weighting the signal subspace to remove noise energies from this subspace. The motivation to choose KLT is its optimality in compression of information, while the DFT and the DCT are suboptimal.

In [81] we introduced a novel approach for the optimal subspace partitioning using the Variance of the Reconstruction Error (VRE) criterion. This criterion provides consistent parameter estimates and allows us to implement an automatic noise reduction algorithm that can be simply applied to the observed data. In this paper we propose a new VRE criterion and we use it to improve the performance of real-time speech recognition that is used for interacting with the RFID network. The main advantage of our method is that it is a wave-in-wave-out method and does not require the separation of noise information. This approach is well adapted to the kind of application we target.

5.7.1 Online VRE-KLT speech enhancement

Let's consider stochastic processes $X(t)$, $N(t)$, and $S(t)$, that have generated $x(t)$, $n(t)$, and $s(t)$, respectively, are wide sense ergodic. We define a real-valued observation vector $x(t) \in \mathbb{R}^K$ to be the sum of the signal vector $s(t) \in \mathbb{R}^K$ and a noise vector $n(t) \in \mathbb{R}^K$, i.e.,

$$x(t) = s(t) + n(t). \tag{5.1}$$

We arrange a *K*-dimensional observation vector in a $M \times N$ Hankelstructured observation matrix $X_{M\times N}(t)$ where $K = M + N - 1$, i.e.,

$$X_{\{M\times N\}}(t) = \begin{pmatrix} x_t & x_{t-1} & \cdots & x_{t-N+1} \\ x_{t-1} & x_{t-2} & \cdots & x_{t-N} \\ \vdots & \vdots & \ddots & \vdots \\ x_{t-M+1} & x_{t-M} & \cdots & x_{t-M-N+2} \end{pmatrix}. \tag{5.2}$$

Due to the ergodicity assumption, we can estimate the covariance matrix R_{xx} using the zero-mean-scaled version of (5.2) as:

$$\widehat{R}_{xx} = \frac{1}{M-1} X^T X \in \mathbb{R}^{N\times N}. \tag{5.3}$$

Let $q_1, q_2, \ldots, q_N$ be eigenvectors corresponding to the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_N$ of the correlation matrix $R_{xx} \in \mathbb{R}^{N\times N}$. By defining the Q matrix as :

$$Q = [q_1 \ q_2 \dots \ q_N] \in \mathbb{R}^{N \times N}, \tag{5.4}$$

where the eigenvectors are orthonormal due to the symmetry in R_{xx} and ordering the eigenvalues in decreasing order in a diagonal matrix:

$$\Lambda \ = \ diag(\lambda_1, \lambda_2, \dots, \lambda_N) \in \mathbb{R}^{N \times N}$$

$$\text{where } \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N \geq 0.$$

For positive-definite matrices, we can decompose the original matrix R_{xx} into its eigenvalue decomposition (EVD),

$$R_{xx} = Q \Lambda Q^T. \tag{5.5}$$

The separation of eigenvectors and principal components matrices gives:

$$Q = \left[\hat{Q}_l \middle| \tilde{Q}_{N-l}\right] \quad Y = \left[\hat{Y}_l \middle| \tilde{Y}_{N-l}\right] \tag{5.6}$$

$$X = \hat{Y}_l \hat{Q}_l^T + \tilde{Y}_{N-l} \tilde{Q}_{N-l}^T = \hat{X} + N \tag{5.7}$$

with

$$\hat{X} = X \tilde{C}_l \quad \text{and} \quad N = X \tilde{C}_{N-l} \tag{5.8}$$

where the matrices $\hat{C}_l = \hat{Q}_l \hat{Q}_l^T$ and $\tilde{C}_{N-l} = I_N - \hat{C}_l$ constitute the KLT model. The variable reconstruction approach is presented. Imagine that our signal is corrupted with a noise n_j along a direction $\xi_j \in \mathbb{R}^N /$:

$$x = s + n_j \xi_j, \tag{5.9}$$

where $\|\xi_j\| = 1$. The task of signal reconstruction is to find an estimate for s along the direction of upset ξ_j to best correct the effect of the upset. In other words, we try to find $\hat{n}_j$ such that:

$$\hat{s} = x - \hat{n}_j \xi_j \tag{5.10}$$

has minimum model error, i.e.,

$$n_j = \arg\min_{n_j} \|s - \hat{s}\| = \arg\min_{n_j} \|\tilde{s}\|^2 = \arg\min_{n_j} \left\|\tilde{x} - \tilde{n}_j \tilde{\xi}_j\right\|^2, \tag{5.11}$$

where $\tilde{s}$ and $\hat{s}$ sign show the clean signal portion in the RS and PCS respectively. The solution to (5.12) can be easily found using least squares,

$$\hat{n}_j = \frac{\hat{\xi}_j^T \tilde{x}}{\hat{\xi}_j^T \hat{\xi}_j} = \frac{\hat{\xi}_j^T x}{\hat{\xi}_j^T \hat{\xi}_j}. \tag{5.12}$$

Substituting the above $\hat{n}_j$ into equation (5.11) we obtain the best signal reconstruction as follows.

$$\hat{s} = \left(I - \frac{\tilde{\xi}_j \tilde{\xi}_j^T}{\tilde{\xi}_j^T \tilde{\xi}_j}\right) x = \left(-\tilde{\xi}_j^o \tilde{\xi}_j^{oT}\right) x, \tag{5.13}$$

where $\tilde{\xi}_j^o \equiv \tilde{\xi}_j / \left\|\tilde{\xi}_j\right\|$. Substituting equation (5.10) into equation (5.11), we obtain,

$$s - \hat{s} = (\hat{n}_j - n_j)\xi_j. \tag{5.14}$$

Substituting equation (5.10) into equation (5.13) we get:

$$\hat{n}_j - n_j = \frac{\tilde{\xi}_j^T s}{\tilde{\xi}_j^T \tilde{\xi}_j}. \tag{5.15}$$

Therefore, the reconstruction error, is

$$s - \hat{s} = (\hat{n}_j - n_j)\xi_j = \frac{\tilde{\xi}_j^T s}{\tilde{\xi}_j^T \tilde{\xi}_j} \xi_j. \tag{5.16}$$

and

$$\|s-\hat{s}\| = \frac{\left|\tilde{\xi}_j^T s\right|}{\tilde{\xi}_j^T \tilde{\xi}_j}. \tag{5.17}$$

From equations (5.16) and (5.17) we observe that the variance of $(s-\hat{s})$ occurs only in the reconstruction direction ξ_j, which, is the same as the variance of $n_j - \hat{n}_j$ and, the reconstruction error $(s-\hat{s})$ depends on the number of PCs retained in the PCA model. The number of optimal PCs is then obtained by achieving the minimum reconstruction error. Further, if the reconstruction error is minimized for a particular magnitude n_j, it is minimized for all magnitudes. Therefore, we can simply determine the number of PCs for the case of $n_j = 0$ to achieve the best reconstruction. Assuming $n_j = 0$, the variance of the reconstruction error (VRE) in the direction ξ_j can be calculated as follows,

$$u_j \equiv \mathrm{var}\{\xi_j^T(x-\hat{s})\} = \mathrm{var}\{\hat{n}_j\} = \frac{\tilde{\xi}_j^T R_{xx} \tilde{\xi}_j}{\left(\tilde{\xi}_j^T \tilde{\xi}_j\right)^2}, \tag{5.18}$$

where R is the covariance (correlation) matrix defined in equation (5.3). In Equation (5.19), u_j is the variance of the reconstruction error in the estimation of s by using $\hat{s}$.

In order to find the number of PCs, we have to minimize u_j with respect to the number of PCs. Considering different noise directions, we propose the VRE to be minimized as

$$VRE(l) = \sum_{j=1}^{N} \frac{u_j(l)}{\xi_j^T R \xi_j}. \tag{5.19}$$

In this equation we calculate the VRE by summing u_j in all dimensions. In order to equalize the importance of each variable, variance-based weighting factors are applied. The criterion that we propose to define the optimal order of reconstruction is given by the following equation:

$$l_{opt} = \min\left(\arg\min_{l}[VRE_{t-\alpha}(l)]\right), \tag{5.20}$$

where t is the frame index and α is the number of past frames that are used to determine the optimal number of components. Equation (5.20) gives more robustness to the process of determining the order of reconstruction and prevents from rapid fluctuations of that optimal order that could be due to artifacts. In our case $\alpha = 3$.

5.8 Experiments and results

5.8.1 Evaluation of the dialog framework

In order to evaluate if the flexibility of the AIML dialog scheme in the context of a RFID application, we proceeded with informal dialog testing. We accomplish this by testing different ways of achieving the task of going to the inventory screen and then having the system read out which items are checked out. Table 5.1[36] shows possible valid scenarios. These examples show how the system can cope with a certain measure of speech recognition inaccuracy as well as its ability to allow the user to express him/her more fluently.

5.8.2 Evaluation of the KLT-VRE enhancement method

To evaluate the performance of the VRE enhancement technique, we carried out extensive objective quality tests with the NOIZEUS database. We use all 30 sentences of the NOIZEUS database [130]. Noisy signals were generated by adding a factory noise from the Noisex-92 database [132] to the clean signals of NOIZEUS database, at 0 dB, 5dB and 10 dB overall SNR. The frame sizes were chosen to be 30 ms long with 40% overlap. An 8 kHz sampling frequency (downsampled from the original) and a Hamming window were applied. The methods to be compared with VRE are the Minimum description length (MDL) [129], Wiener and the Spectral Subtraction (SS) methods [130]. The objective measures used for the evaluation are the Weighted Spectral Slope (WSS) distance (smaller reflects less distortion) and the Perceptual Evaluation of Speech Quality (PESQ). These measures are chosen because ASR gives a trend corresponding to that of subjective intelligibility that is strongly correlated to these measures [133]. For the definitions of these measures see [130]. In

[36] Le calcul du PSEQ utilisé dans le tableau 5.1 utilise l'algorithme de Loizou [130] qui a été supplanté par la norme UIT862.1[131] permettant de normaliser les scores PSEQ.

Table 5.1 : Objective evaluation under a factory noise degradation.

Objective Measures	Input SNR(dB)	Weiner	SS	MDL	VRE
WSS	0	126.89	142.21	141.14	**126.48**
	5	95.47	102.34	104.28	**93.88**
	10	**65.32**	75.25	82.56	68.21
PESQ|UIT862.1	0	1.62 | 1.38	1.28 | 1.24	0.95 | 1.15	**1.74 | 1.45**
	5	1.96 | 1.60	1.92 | 1.57	1.58 | 1.36	**2.14 | 1.75**
	10	**2.74 | 2.45**	2.42 | 2.04	1.78 | 1.48	2.72 | 2.42

Table 5.2 : Possible valid dialogs of Speech-enabled system.

Example with perfect speech recognition	
User : Show me the inventory please. ***System****: tab inventory* **User** : Tell me which items are checked out please. ***System****: read chkout inventory*	**User** : Go to inventory tab. ***System****: tab inventory* **User** : Are there any checked out products? ***System****: read chkout inventory*
Example with speech recognition errors	
User : Show better inventory peace. ***System****: tab inventory* **User** : The witch items or checked out please. ***System****: read chkout inventory*	**User** : Got inventory tab. ***System****: tab inventory* **User** : Arthur Anny checked out products? ***System****: read chkout inventory*

VRE and MDL, $N = 21$ (KLT dimension). In Wiener, α (the smoothing factor for the Decision-Directed method for estimation of A Priori SNR) equals 0.99, and the smoothing factor for the noise updating is 9. In SS, c (the scaling factor in silence periods) is set to 0.03. The results given in Table 5.1 show that VRE outperforms the other methods.

5.9 Conclusion and future work

Automatic speech recognition and text-to-speech have become sufficiently mature technologies to allow their effective inclusion in RFID networks. In this paper we presented a viable and realistic solution which, takes into account the difficulties faced by both the RFID and speech technologies.

We have incorporated an effective enhancement technique and a flexible dialog framework in a Microsoft platform. As expected, the results show that the implementation of AIML and VRE allows the RFID operators to interact more naturally with the system in noisy environments. Thus, this solution could constitute a promising answer to the growing need for ubiquitous access to information that is driving today's technological advances in many real-life applications using RFID technology. In future work will look at modifying the AIML framework to support semantic and pragmatic interpretation instead of pure textual analysis. This should provide a more flexible and powerful framework to create robust dialog management.

6 ONTOLOGY-BASED PATTERN GENERATOR AND ROOT SEMANTIC ANALYSER FOR SPOKEN DIALOGUE SYSTEMS

Générateur de motif et analyseur sémantique pour systèmes de dialogue verbaux

Y. Benahmed[1,2]*, S-A. Selouani*[2] *et D. O'Shaughnessy*[1]

[1]INRS-EMT, 800 de la Gauchetière O, H5A 1K6, Montréal, Qc, Canada
[2]LARIHS Lab. Université de Moncton, Campus de Shippagan, New Brunswick, Canada

Article soumis à : *La 25e Conférence Canadienne de Génie Électrique et Génie Informatique*

Lieu de la conférence : Montréal, Canada

Accepté et publié

Date de publication : 29 avril 2012

6.1 Contribution des auteurs :

Yacine Benahmed: (candidat)

Écriture de l'article, développement des logiciels et programmes, expérimentation, analyse des données, préparation des graphes, des figures et des tableaux.

Douglas O'Shaughnessy: (directeur)

Conseils techniques et théoriques, conseils de rédaction, édition, coauteur, corrections.

Sid-Ahmed Selouani: (codirecteur)

Conseils techniques et théoriques, édition, coauteur, corrections.

6.2 Commentaires des évaluateurs

La majorité des commentaires des évaluateurs ont été pris en considération lors de la révision finale de la version de l'article présenté dans cette section.

6.2.1 Évaluateur no 1

Comments for the Authors:

This paper presents the dialogue interpreter subsystem which allows the user to interact by speech with a Radio-Frequency IDentification (RFID) network working in a highly noisy environment. Algorithms have been presented and the evaluation results are included. I feel that the results could be elaborated more. Overall, the paper has some merit.

6.2.2 Évaluateur no 2

Comments for the Authors:

General comments: 1. First part of Section 2: figure 1 ? Figure 1, also apply to all other cases. 2. Start Subsection 2.1 on the new page or move it up. 3. Caption of Fig.1: Network ? network. 4. In Section 5: You have 5.1 but no 5.2, … 5. Syntax error in the first sentence of Subsection 5.1. 6. Last sentence in the conclusion: English syntax error. 7. Inconsistency of writing references. See ref [2] and [3]. Also check conference formatting rules. 8. In Section 2.2: Where is Table 2.2 ??

6.2.3 Évaluateur no 3

Comments for the Authors:

The following small errors should need to be corrected: (VRE-KLT)[6]… (VRE-KLT) [6]… It was designed by Wallace[5]… It was designed by Wallace [5]… such as OWL 2[9] and SUO-KIF[10]. such as OWL 2 [9] and SUO-KIF [10]. WorndNet®[11]… WorndNet [11]… being a synonym[11]. being a synonym [11]. Zesch et al.[12] since… Zesch et al. [12] since… with TreeTagger[13]. with TreeTagger [13]. In order to evaluate if the flexibility… In order to evaluate the flexibility… [4] C. C. Fung G. O. Sing,… [4] C. C. Fung, G. O. Sing,… Reference [10] should be properly justified [12] Torsten Zesch, Christof Müller, and Iryna Gurevych,… [12] T. Zesch, C. Müller, and I. Gurevych,…

6.3 Résumé

Cet article propose une amélioration au gestionnaire de dialogue proposé dans [80]. Rappelons qu'il s'agit d'un système basé sur la parole facilitant l'interaction avec les réseaux RFID dans des environnements bruyants. Un des désavantages principaux que nous avons rencontrés avec le langage AIML est la nécessité de créer des motifs pour toute les variations d'une commande possible. Par exemple, un des problèmes est qu'une simple requête de recherche d'inventaire nécessite une multitude de motifs pour couvrir tous les lemmes, les conjugaisons et toutes autres variations possibles. La solution proposée vise à faciliter la conception de grammaire/motifs-type de domaine via la mise en œuvre d'un générateur de motif basé sur les ontologies. Ce nouveau cadre de dialogue est proposé afin de donner aux opérateurs humains une plus grande flexibilité de communication avec le système qui sera dorénavant en mesure de tenir compte des variations possibles de commandes-type. Nos résultats démontrent que le système est en mesure de générer de nouveaux motifs à partir des motifs de base tout en minimisant l'impact au niveau de la perplexité du modèle de langage généré.

Le texte qui suit est est une copie exacte de l'acte de conférence référencé ici haut[37].

[37] La numérotation des sections, équations, figures, graphiques, tableaux et références ont été adaptés pour correspondre à l'ordre retrouvé dans ce document.

6.4 Abstract

This paper presents improvements in the dialogue interpreter subsystem for an application that allows the user to interact by speech with a Radio-Frequency IDentification (RFID) network working in a highly noisy environment. A new dialog framework is proposed in order to give the human operators the ability to communicate with the system in a more natural fashion. This is achieved by the implementation of the Artificial Intelligence Markup Language combined with an ontology-based pattern generation and root semantical analysis algorithms, which allows the system to respond to close-to natural language queries by means of pattern matching.

IndexTerms— *Natural dialog, pattern generation, semantic analysis, natural language processing, AIML*

6.5 Introduction

Speech-enabled interfaces that include both speech recognition and synthesis are sufficiently advanced to a point where it becomes feasible to design viable and desirable solutions in various data collection applications. One of these innovative applications consists of providing the speech modality to allow the human operators of a Radio Frequency IDentification (RFID) Network to communicate naturally with related devices and information systems. The advantages of speech hands-free and eyes-free systems combined with the RFID technology are expected to improve the speed, ergonomics and safety of numerous operations such as order picking, quality management, tracking and monitoring of assets, and shipping operations. In order to maintain an acceptable level of accuracy, the use of speech recognition is however limited in widely used applications to a fairly short list of input keywords for control and commands. For instance in Microsoft's approach [124], numeric characters for voice data inputs are used to select the command. Increasingly, and in all types of tasks, users are now requiring more naturalness in their verbal communications with machines. Therefore, making the speechenabled interfaces more natural is one of the crucial issues for their deployment in real-life applications such RFID.

Most interfaces incorporating speech interaction fall into three broad categories. The first category includes Command and Control (C&C) interfaces that rely on a fixed task dependent grammar to

provide user interaction [125]. Their main advantage is their ease of implementation and high command recognition rate. However, their downside is the high cognitive load they induce when users interact with the system because of its lack of flexibility and lack of uniform command sets. The second category is based on interactive voice response (IVR) that guides users by the means of prompts in order to validate the utterance at every step [126]. This style of interaction is mostly used in menu navigation such as that found with phone and cable companies. Its relative lack of efficiency for fast interaction makes it a poor choice for everyday use. Finally, the third category uses natural language processing to parse the user's utterance and to determine the goal of the request. This can be done through multiple ways such as semantic and language processing and filtering [127]. Hence, to be effective, due to "limitless" vocabulary, this type of interface needs an accurate recognizer.

However one problem still remains with the implementation of such interfaces, in the sense that a lot of handcrafting of rules and knowledge is still required. This quickly becomes an issue as the complexity of the interface increases since it is extremely time consuming and ultimately inconceivable to manually create a complete set of speech understanding rules.

In this paper, we present an effective framework for generating patterns from root semantical patterns for use with an interactive spoken dialog system belonging to the third category, using a dynamic pattern matching technique based on the Artificial Intelligence Markup Language (AIML) [128]. That is, the goal is to allow developpers to quickly generate possible utterances for user interaction only by having to specify general utterance rules.

The organization of the paper is as follows. Section 6.6 presents the overall system that combines RFID and speech technologies to improve enterprise operations and also describes the AIML-based dialog framework used in our application. In Section 6.7 we briefly introduce the concept of ontologies and their pertinence in natural language processing. Section 6.8 introduces the ontology-based pattern generation and root semantic analysis algorithms. An evaluation of the system is reported in section 6.9, and in section 6.10 the paper is concluded.

6.6 Speech-enabled RFID system

Typically, an RFID-based process involves a human operator to perform multiple and simultaneous tasks in order to effectively use a number of devices and input devices to update an information system. The workflow may include other additional tasks such as inspecting items received for damages, confirming the quantity of a given product, etc. To reduce the cognitive load of those operators and to improve their comfort, the sensor capabilities of RFID nowadays provide a very promising solution. As illustrated in Figure 6.1, the proposed system enables the orchestration of RFID events and inputs into synchronized operations in order to provide more automation thanks to the speech modality. Data collection is performed through the RFID devices. The readers and tag communication are managed by the speech-enabled system. Operator can speak in order to modify power, select antenna, launch the Read/Write tags, check inventory, etc.

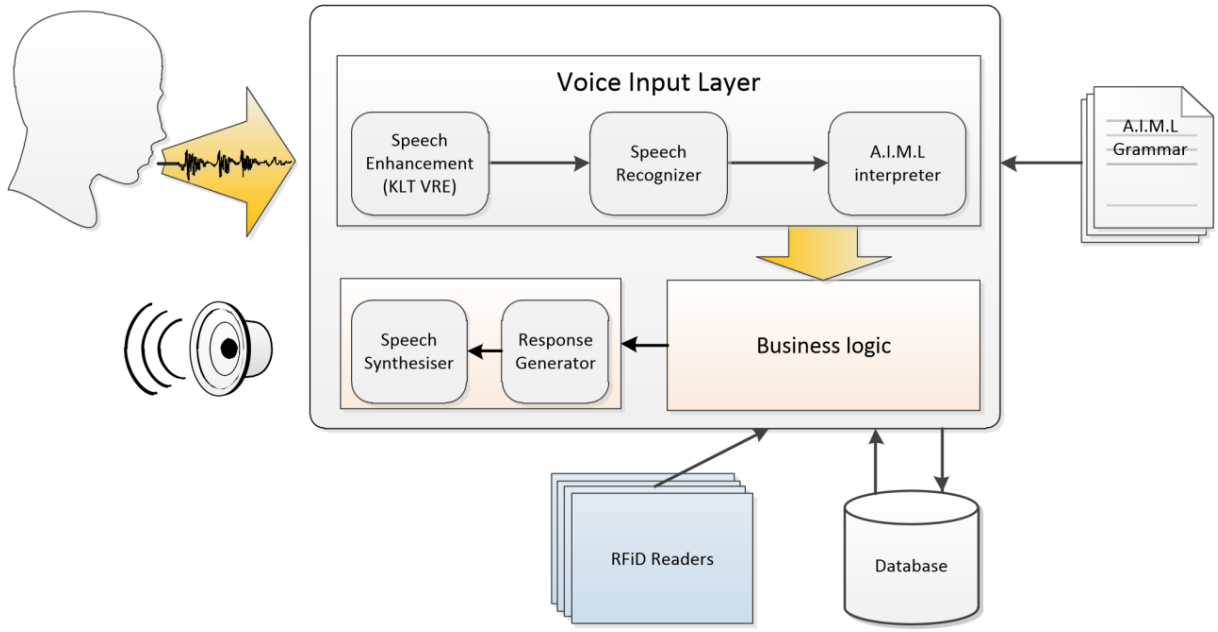


Figure 6.1 : The overall speech-enabled system interacting with the RFID Network.

6.6.1 System details

The application is composed of two major components:

1. The first component is the speech input layer. Here speech utterances are captured in a buffer and passed either directly to the automatic speech recognition (ASR) system or to

the noisy speech enhancer according to the user preferences. Noisy speech is enhanced by an online signal subspace enhancement technique based on the Variance of the Reconstruction Error of the Karhunen-Loeve Transform (VRE-KLT) [12] whose main advantage is that it is a wave-inwave-out method and does not require the separation of noise information. The N-best ASR recognition outputs are then fed to a dialog manager which, parses them and returns either a user-friendly message to the user to indicate that it did not understand the meaning of the utterance or the most likely command with its parameters to the business logic end. For example, if the system recognizes the user input "Show me the RFID Reader Configuration", the system would return the RFID set up form. Feeding the N-best ASR outputs gives a certain error tolerance to the system since the most likely recognized sentence is not always the best one.

2. The second component is the business layer. It is constantly listening for user input as well as listening for RFID device events such as unauthorized inventory check out.

6.6.2 AIML-based dialog interpreter

The dialog interpreter used in this application, like our previous work [9], [80], [134], relies on a pattern matching language to understand user's utterances. More specifically, the Artificial Intelligence Markup Language (AIML), an XML compliant language designed to create chat bots that can fool users into thinking that they are chatting with a person. It was designed by Wallace [128] to create chat bots rapidly and efficiently. Its primary design feature is minimalism, and reductionism by recursivity. That is, multiple inputs can be reduced to a general model. As such, it is essentially a pattern matching system that maps well with Case-Based Reasoning [128]. With AIML, botmasters effectively create categories that consist of a pattern, the user's input, and a template, the Bots' answer. The AIML parser then tries to match what the user said to the most likely pattern and outputs the corresponding answer. Additionally, patterns can include wildcards, which can be useful for catching irrelevant input or the target of requests. Moreover, the language supports recursion, which enables it to answer based on previous input. Table 6. presents an example of a simple grammar used to inquire about the quantity of a particular product that was shipped to a certain destination. The first category represents the general pattern to be identified (the actual search with its arguments). The star "*" found in the `pattern` tag represents a wildcard, and as such will match any input. In the `template` tag, we find the `star` tag. It points

Table 6.1 : Example of an AIML category.

```
<category>
      <pattern>
            HOW MANY * SHIPPED TO *
      </pattern>
      <template>
            N SHIPPED <star index="1" />
            TO <star index="2" />
      </template>
</category>

<category>
      <pattern>
            * HOW MANY * WERE SHIPPED TO *
      </pattern>
      <template>
            <srai>
                  HOW MANY <star index="2" />
                  SHIPPED TO <star index="3" />
            </srai>
      </template>
</category>
```

to the wildcard from the pattern and means that its value will be returned as output. For example the input "How many *radios* shipped to *Montreal*" would return "N SHIPPED *radios* TO *Montreal*". The second category uses the srai tag. It is used for reductionism, in this example, to the "HOW MANY * SHIPPED TO *" pattern. Furthermore, the index attribute from the star tags with value of 2 and 3 means that for the input "Tell me how many radios were shipped to Montreal" the template would ignore the text preceding "how many" and reduce it to the "N SHIPPED *radios* TO *Montreal*" output.

One of the main disadvantages that we encountered with the AIML language is that we needed to manually create patterns for all possible utterances covered by the application. That is, for a simple search request, multiple patterns were required to cover all lemmas, conjugation and variations thereof. What we will seek to accomplish is to leverage the ontologies to create our patterns. The designer will need only to specify one pattern per possible action/request and the system will initially be able to generate all possible variations from the ontological relations; this is addressed in section 6.8.

6.7 Ontologies in natural language processing

Ontologies are used to relate concepts in a hierachical and relational fashion that ultimately describe the meaning of terms. Logical and philosophical entities can be grouped contextually with the use of named relations. Figure 6.2 shows a portion of the entry for the ontology of a cat as it relates to rodents and birds. In this example, solid lines represent inheritance, dashed lines such as the ones connecting cat with bird and rodents represent functional relationships and the dotted line between vertebrate and spine represent inherited properties.

Ontologies can be described by numerous formal languages. It is therefore important to properly choose a language that fits the needs of the application, ideally choosing a standard-based language

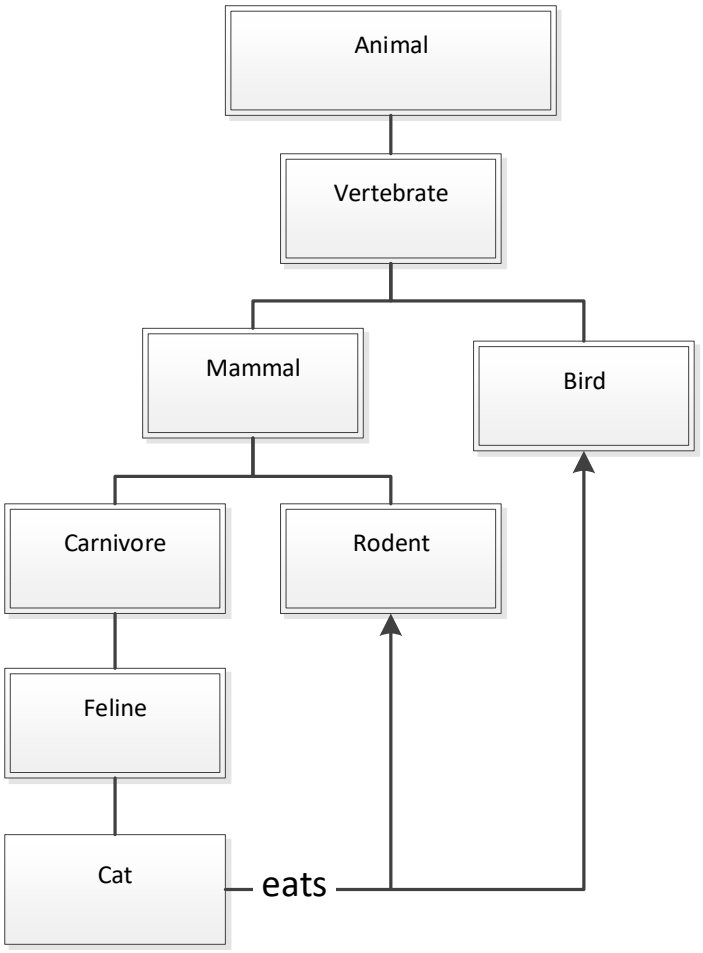


Figure 6.2 : Example of a simple ontology diagram representing animals and their relations and properties.

such as OWL 2 [135] and SUO-KIF [136]. This eventually ensures that the application will be

compatible with external ontologies and be able to quickly implement new knowledge with minimal involvement from the developper.

6.8 Ontology-based speech pattern generator

6.8.1 Ontology

In the current itteration of our application, the ontology used is WordNet [88] from Princeton University. It is essentially a large (15,5287 defined words) lexical database of English nouns, verbs, adjectives and adverbs. The main relation between defined words is synonymy, the state of being a synonym. We are also currently working to increase the ontological knowledge of WordNet to exploit data from Wiktionary in the spirit of Zesch et al [31] since all defined terms are tagged with their part of speech and can have relations to other terms defined.

6.8.2 Pattern generation and root semantic analysis algorithms

The pattern generation algorithm is described as follows and is inspired by the work in paraphrasing for automatic evaluation by Kauchak et al [85]. For an AIML grammar *G* composed of patterns P_i, first do part of speech tagging for every term T_j in P_i. That is, assign a part of speech tag, *n* for noun, *v* for verb, *adj* for adjective and *adv* for adverb to every $T_j \Rightarrow T_j^{pos}$. Then, for every term T_j^{pos} return a vector $\mathbf{T}_j^{pos}$ of the k^{th} ontologically related terms where *k* is the maximum relational distance between two ontological roots defined by the number of links separating them. In this case a simple depth-limited search is implemented with a complexity of $O(|M| + |E|)$ where $|M|$ is the number of vertices (terms) and |*E*| is the number of edges (links), with the knowledge that there is "no" solution and that the problem of optimality is not pertinent. That is we are not concerned with the possibility of finding a solution that is more expensive than a solution in another path. Finally generate a matrix of new patterns $\mathbf{P}_i$ by a cartesian product of every $\mathbf{T}_j^{pos}$ with a maximum complexity of $O(N * M)$ where N is the number of terms in the pattern and M the number of related terms for every term. Consequently, the overall approximate complexity of the algorithm is $O\big(LN(M + E)\big)$ where L is the number of patterns in the grammar.

Algorithm 6.1 : Ontological-based speech pattern algorithm where: G is the grammar, P the grammar's patterns and, T the pattern's terms.

for all $P_j \in G$ **do**
  for all $T_j \in P_i$ **do**
    do part-of-speech-tagging(T_j)
    return vector $\mathbf{T}_j^{pos}$ of related terms
  end for
  Combine $\mathbf{T}_j^{pos}$ for matrix of new pattern $\mathbf{P}_i$
end for

For the root semantic analysis algorithm, which allows novel utterances to be processed, the operation is simplified and works as follows. For each term T_j in the recognized utterance U do part of speech tagging to obtain T_j^{pos}. Then, for each T_j^{pos} return a vector $\mathbf{T}_j^{pos}$ of the k^{th} ontologically related terms. Generate a matrix of new utterances $\boldsymbol{U}$ by a cartesian product of every T_j^{pos} . Then, for each root pattern P_{root} composed of $T_{root,j}$ terms in the grammar G find the maximal similarity measure, in this case through a simple term frequency/inverse document frequency (tf-idf) measure. If the similarity is within 0.6, align the matching terms and push the unmatched terms to the wildcards.

Algorithm 6.2 : Ontological-based root semantical analysis algorithm where: G is the grammar, P_{root} the grammar's root patterns and, $T_{root,j}$ the root pattern's terms, U the user's recognized utterance and T_j U's terms.

for all $T_j \in U$ **do**
  do part-of-speech-tagging(T_j)
  return vector $\mathbf{T}_j^{pos}$ of related terms
end for
Combine $\mathbf{T}_j^{pos}$ for matrix of new pattern $\mathbf{P}_i$
return $\operatorname{argmax}\big(\text{tf-idf}(\mathbf{P}_i, P_{root})\big)$

In our application, the part of speech tagging operation is done with TreeTagger [137]. Since TreeTagger's output is based on the Penn-Treebank tagset, a simple mapping is done between the Penn-Treebank tagset and the WordNet tagset used for our classification process. For even easier processing we can envisage converting the terms in every pattern to their lemma. Since TreeTagger can output the lemmas this would ultimately reduce the ontological search and ultimately reduce the complexity of the proposed algorithm.

6.9 Experiments and results

6.9.1 Evaluation of the pattern generator

In order to evaluate if the flexibility of the ontology-based pattern generation algorightm, we proceeded with the generation of all possible variations of the root patterns of the speech-enabled RFID application described in section 6.6. The original grammar consists of 47 distinct patterns. Furthermore in the interest of evaluating the impact of complexity we measure the generated language model's complexity before and after generation with the use of the SRILM [93] tool against a sample of 417,000 sentences extracted from Wiktionary. The pattern-generation algorithm yielded 133 additional patterns which should certainly and shows how the system can now cope better with a certain measure of utterance variability. Table 6.2 shows a sample of the new patterns generated by the algorithm with relation to their root pattern. We see that the patterns are still semantically significant in terms of user interaction. As for the perplexity of the new Original patterns grammar, as expected it only increased by around 0.15 points as illustrated by Table 6.3.

Table 6.2 : Example of additional AIML categories generated by the pattern generation algorithm.

Original patterns	
how many * were shipped to *	* out inventory *
Generated patterns	
how many * were embarked to * how many * were sent to * how many * were transported to *	* out products * * out wares *

Table 6.3 : Perplexity (ppl) evaluation of original and generated grammars as measured by SRILM tool.

Sentence	**PPL**
Original	3.62633
Generated	3.77896

6.10 Conclusion and future work

In this paper we presented an improvement to a previously published system, which takes into account the difficulties faced by both the RFID and speech technologies. We have incorporated an effective and flexible dialog framework in a Microsoft platform through the implementation of a AIML pattern generation algorithm that leverages the knowledge contained within ontologies as well as the implementation of an algorithm to reduce novel utterances to their root semantical patterns without significantly altering the perplexity of the grammar. In future work will look at leveraging multiple domain ontologies through the use a distributed computing paradigm in order to more rapidly compute the semantic and pragmatic interpretations contained in user utterances.

7 A BIN-BASED ONTOLOGICAL FRAMEWORK FOR LOW-RESOURCE N-GRAM SMOOTHING IN LANGUAGE MODELLING

Un cadre basé sur des groupements ontologiques pour le lissage de *N*-grammes dans un contexte de faibles ressources pour la modélisation du langage

Y. Benahmed[1,2]*, S-A. Selouani*[2] *et D. O'Shaughnessy*[1]

[1]INRS-EMT, 800 de la Gauchetière O, H5A 1K6, Montréal, Qc, Canada
[2]LARIHS Lab. Université de Moncton, Campus de Shippagan, New Brunswick, Canada

Article soumis à : *The 39th International Conference on Acoustics, Speech and Signal Processing*

Lieu de la conférence : Florence, Italie

Accepté et publié

Date de publication : 4 mai 2014

7.1 Contribution des auteurs :

Yacine Benahmed: (candidat)

Écriture de l'article, développement des logiciels et programmes, expérimentation, analyse des données, préparation des graphes, des figures et des tableaux.

Douglas O'Shaughnessy: (directeur)

Conseils techniques et théoriques, conseils de rédaction, édition, coauteur, corrections.

Sid-Ahmed Selouani: (codirecteur)

Conseils techniques et théoriques, édition, coauteur, corrections.

7.2 Commentaires des évaluateurs

La majorité des commentaires des évaluateurs ont été pris en considération lors de la révision finale de la version de l'article présenté dans cette section.

7.2.1 Meta évaluation no 1

Overall Evaluation

Interesting idea with some preliminary results from small-scale experiments. One main issues is that the interpolation weights in Eq ([7].4) are tuned on test sets, rather than a hold out dev sets, which makes the perplexity results in Table 1 unconvincing. Further, my guts feeling is that these weights should be better tied to w_n rather than globally shared as the way the authors present it. Is there any theoretical or experimental justification for this?

The basic idea of integrating semantic relations have also been explored in other ways such as using word clustering, and integrate them into a max-ent (log-linear) LM. The authors should discuss the connections and comparisons.

7.2.2 Évaluateur régulier no 2

Comments to authors

The idea is interesting, but the experimental validation lacks a proper baseline comparison against simple class based LMs.

Also, working on an artificially weakened baseline makes the relevance of the work questionable: is there no actual low-resource task where the approach can be used to improve the state of the art?

7.2.3 Évaluateur régulier no 3

Comments to authors

Section 1 : This talks about a specific kind of language modeling - namely "statistical language modeling" but manages to convey the impression that all language modeling is probabilistic. It would be useful to maintain this context whenever categorical statements are made, e.g. "Formally, we compute the probability of word w = w_n given its history or context"

"number of times that we observed h,w" should be "number of times that we observed {h,w}" or some reasonable set/sequence notation.

Equations 1 and 2 are repetitions and should be consolidated.

Page 1, Section 1, last para: the reasoning is somewhat vague. Sensitivity to corpora and data sparseness are both related to one underlying cause - that word distributions in any language are observed to follow Zipf's law. Regardless of the size of the corpora, there will be a large number of N-grams that will not have sufficient counts. The patterns of N-grams differ in different text corpora, and Zipf's law compounds the differences to the extent that statistics obtained from one are largely inadequate to predict trends in the others.

Page 1, Section 1, 2nd column : The statement "if we train our model on text from Shakespeare, it is evident that the model will not be adequate for the prediction of astrophysical text." should be qualified more technically. The word "generalizability" would be more correct than the word "significance".

The statement "This paper combines Witten-Bell smoothing over Kneser-Ney smoothing is that our smoothing algorithm produces real numbered counts that are not supported by the current implementation of the SRILM Toolkit." does not make sense in its current form.

Section 2, first para: The significance of the length of the lines is not explained even in the caption of Fig. 1. If this is merely a stylistic reference the fact should be mentioned.

"by smoothing its probabilities with those of weighed n-grams of related terms" : "weighed" should be replaced with "weighted" here and elsewhere in the paper.

Section 3.1 : Section 3: While the approach is interesting in itself and obviously works, there still remains the matter of providing good explanations in the paper. In a world where a lamb bleats while a dog barks, existing techniques assign minimal probability to the unseen N-grams "lamb barks" and "dog bleats". If however the ontologies of the two are merged as proposed, this probability is likely to be higher unless further semantic constraints are placed. To give a specific example : why would one expect a word that is 3 distant on WordNet from a word w to have 1/3rd the probability of w in every context? Here's a sanity check that is puzzling: Let R=1, assume that (w_n | history) is not in the training text, but w_n^, which is 3-distant from w_n, is present in the training text. Then, according to Eq.3:

P(w_n | history) = 1/3 * P(w_n^ | history).

How can this be justified? Eq. 4 does average out these unexplained counts over many words, but it is not clear why this averaging is not the only reason why the technique does not fail. The premises on which this technique is built are not justified in the paper. E.g., why is WordNet distance any predictor for N-gram probabilities in the first place? Using the technique presented it should be possible to build a single language model from an ontology that would generalize to corpora containing all words present in the ontology, starting from a uniform language model. How would this model generalize in comparison to the models built from actual corpora that use standard smoothing techniques, or indeed no smoothing techniques?

Section 3.3 : Again, "unweighed graph" should be "unweighted graph" to the best of my knowledge. this applies to all instances of the word "unweighed" in the paper.

Section 4 : Experimental results: The LMs created are pre-smoothed using Witten-Bell and Kneser-Ney discounting. It is not clear why the proposed algorithm is used to smooth already smoothed N-gram counts. At least this is what comes across from the description in para 2, which is hard to interpret.

Section 4.1 : "Unfortunately, the SRILMtoolkit does not support float counts for Kneser-Ney modeling" : there are issues with this statement. Here, and elsewhere, "discounting" or "smoothing" are replaced with the word "modeling". That is not accepted convention. "float counts" could be "floating point counts", although I am not sure why counts should be floating point numbers (unless some normalization is implicit in the word "count")

Section 4.3 : "This experiment investigates the use impact" - seems to be an insertion here.

Figure 2 is indecipherable and not explained properly. What are we looking at? Why are there spikes in perplexity? What is the main message of this figure? The same comments hold for Figure 3. In fact the WER% spikes are unusual - good ASR systems do not behave this way under constrained optimization.Improvements in WER% in Table 2 do not seem to be statistically significant.

Section 4 in its entirety is poorly written, with multiple errors and unclear statements. Section 5 is also poorly written - it makes categorical statements such as "We showed that it reduced word error rate (WER) in a statistically significant manner compared with state-of-the-art LM techniques."

This conclusion is questionable given the contents of Table 2 and the lack of explanation thereof. Note also that unsupervised adaptation can actually reverse WER trends in comparable experiments, especially if the WERs variations are not statistically significant.

I suggest the authors simplify the graphs presented, show just one or two significant lines, and explain them properly.

7.3 Résumé

Dans cet article, nous introduisons une nouvelle méthode de lissage des modèles de langage basée sur les informations sémantiques trouvées dans les ontologies et qui est spécialement adaptée pour la modélisation du langage dans un contexte de ressources limitées. Nous exploitons la connaissance latente de la langue qui est profondément codée dans des ontologies. En tant que tel, cet article examine la possibilité d'utiliser les relations sémantiques et syntaxiques entre les « synsets » (mot+sens) couverts par l'ontologie WordNet (anglais). Ces propriétés seront alors exploitées afin de générer de nouveaux contextes plausibles pour les événements non vus dans le corpus d'entrainement dans le but de « simuler » un corpus plus vaste. Ces « nouveaux » *N*-grammes sont ensuite interpolés avec un modèle de langue de base lissé à l'aide de l'algorithme de Witten-Bell. Cela, dans le but de réduire, à la fois, la perplexité du modèle de langage et le taux d'erreur de mots dans une tâche de reconnaissance automatique de la parole. Nos résultats expérimentaux démontrent qu'il est possible d'obtenir, d'une part, une réduction significative de la perplexité du modèle (jusqu'à 9,85 % par rapport au modèle de base) et, d'autre part, de réduire le taux d'erreur de mots d'une manière statistiquement significative par rapport à la fois au modèle original lissé à l'aide de l'algorithme de Witten-Bell et d'un autre modèle de langue de base lissé à l'aide de l'algorithme de Kneser-Ney entrainé sur le corpus « Wall Street Journal-based Continuous Speech Recognition Phase II ».

Le texte qui suit est une copie exacte de l'acte de conférence référencé ici haut[38].

[38] La numérotation des sections, équations, figures, graphiques, tableaux et références a été adaptée pour correspondre à l'ordre retrouvé dans ce document.

7.4 Abstract

In this paper, we introduce a novel method of smoothing language models (LM) based on the semantic information found in ontologies that is especially adapted for limited-resources language modeling. We exploit the latent knowledge of language that is deeply encoded within ontologies. As such, this work examines the potential of using the semantic and syntactic relations between words from the WordNet ontology to generate new plausible contexts for unseen events to simulate a larger corpus. These unseen events are then mixed-up with a baseline Witten-Bell(WB) LM in order to improve its performance both in terms of language model perplexity and automatic speech recognition word error rates. Results indicate a significant reduction in the perplexity of the language model (up to 9.85% relative) all the while reducing word error rate in a statistically significant manner compared to both the original WB LM and baseline Kneser-Ney smoothed language model on the Wall Street Journal-based Continuous Speech Recognition Phase II corpus.

Keywords: Language modeling, context modeling, ontologies, low-resource speech recognition

7.5 Introduction

Language modeling is an integral part of the speech recognition task where the purpose is to try to reliably predict what the speaker is most likely to say next. Today, *n*-gram language modeling remains the dominant technique in automatic speech recognition applications. Formally, we compute the probability of word $w = w_n$ given its history or context $h : P(w|h)$ where $h = w_{n-1}, w_{n-2}, \ldots, w_{n-N}$ - is obtained from counts of very large textual corpus. However, this is not always convenient or practically achievable since new sentences are constantly being created [29]. For example it would be impossible to predict a novel sentence, since its count would be zero. In addition, it is easy to see that as the length of the sentence and/or vocabulary grows, the task of obtaining those counts becomes impractical. The goal of *n*-gram language models then is to simplify the prediction of the *n*-th word based on the $n-1$ previous words using maximum likelihood estimates (MLE):

$$P_{MLE}(w_n|h) = \frac{C(h, w_n)}{C(h)} \tag{7.1}$$

Two well-known drawbacks of language modeling with *n*-grams are its sensitivity to the corpus and data sparseness, i.e. if we train our model on text from Shakespeare, it is evident that the model will not be adequate for the prediction of astrophysical text, namely the count of the word "magnetohydrodynamics" in a Shakespearian work $C(magnetohydrodynamics_{astrophysics}|h_{Shakespeare})$ is surely equal to 0. That is why multiple approaches exist to either obtain better language models (LM) or smooth existing ones in order to improve their statistical accuracy and generalizability. Notable techniques include discounting, interpolation and various backoff schemes such as Good-Turing Discounting, Witten-Bell discounting [24], Modified Kneser-Ney smoothing [36] or Hierarchical Pitman-Yor language models [138] used to smooth out low-order counts. Variable length n-grams and class-based n-grams [139] also enabled better performance of language models. Other work such as [140]–[143] looks into exploiting the vast amount of text available on the Web and raw n-gram counts such as those provided by Google.

Although we have made significant leaps in the area there is still room for progress. This is particularly true in the case where we don't have access to training data that relates to the testing or real-world data or that such resources are limited and/or too expensive. It is for these reasons that this work mainly focuses on semantically expanding the information found in small corpus for smoothing language models.

In this work, we introduce a novel method of smoothing language models by exploiting the semantic information found in ontologies. It will be used in combination with Witten-Bell smoothing since the intuition behind this smoothing algorithm is to look at the diversity of contexts in which a particular history occurs: how likely is it to see a new n-gram given a particular history.

This is especially suited for our n-gram count smoothing algorithm since one of its goals is to generate a new set of probable n-gram events that enrich the contextual diversity of the language model.

The contribution of this work is the demonstration that ontological relations can reliably be used for smoothing existing ngram counts as well as producing novel *n*-gram contexts to enrich the

language model and increase its performance both in terms of perplexity and in an automatic speech recognition task.

The outline of this paper is as follows. Section 7.6 introduces ontologies and the WordNet ontology used in this work. Section 7.7 describes our proposed ontological n-gram count smoothing algorithm. Section 7.8 proceeds with the description of the experiment set-up and the evaluation of our algorithm. Finally, in section 7.9, we conclude and discuss our results.

7.6 Ontologies

Ontologies are used to relate concepts in a hierarchical and relational fashion that ultimately describe the meaning of terms. Logical and philosophical entities can be grouped contextually with the use of named relations. Figure 7.1 shows a portion of the entry for the ontology of a cat as it relates to rodents and birds. In this example, solid lines represent inheritance, dashed lines such as the ones connecting cat with bird and rodents represent functional relationships and the dotted lines between vertebrate and spine represent inherited properties.

7.6.1 The WordNet ontology

We used the popular WordNet [88] as our base ontology from Princeton University. It is essentially a large lexical database of English. The main relation between defined words is synonymy, the state of being a synonym [88]. WordNet lemmas are classified by part of speech: adjectives (21,479), adverbs (4,481), nouns (117,798) and verbs (11,529) for a total of 155,287 senses; however since many words are found in multiple parts-of-speech, a total of 147,278 unique indexed words are found in the ontology.

7.7 Ontology-based n-gram smoothing

In this section we introduce our novel technique to improve the performance of *n*-gram smoothing. One of the problems with current language modeling techniques, as previously stated, is that they do not necessarily generalize themselves to words of the same semantic class/sub-class. Counts for words such as cat, dog, mouse, etc., all being animals, could be proportionally smoothed by the *n*-gram entry for *lamb* from the example of section 2.2. The same could be said about relational

markers between objects where similar relations could be smoothed with the help of well established ones. As such, we propose a method of smoothing the *n*-gram counts by leveraging the latent information stored in ontologies. Ontologies are used to relate concepts in a hierarchical and relational fashion. As was stated in section 7.6, entities can be grouped logically and/or philosophically in a contextual fashion through the use of named and weighted relations. Figure 7.1 shows a portion of the WordNet ontology where the ontological distance from cat = 1.

Therefore, by studying the information contained within, we are able to smooth our *n*-gram language model by smoothing its probabilities with those of weighted *n*-grams of related terms, i.e., use the counts of related *n*-grams $C(w_r|h)$ for each word $w_r \in W_R$ related to w_n and its context h to smooth the counts $C(w_n|h)$, where W_R is a vector of R words related to w_n. Inversely,

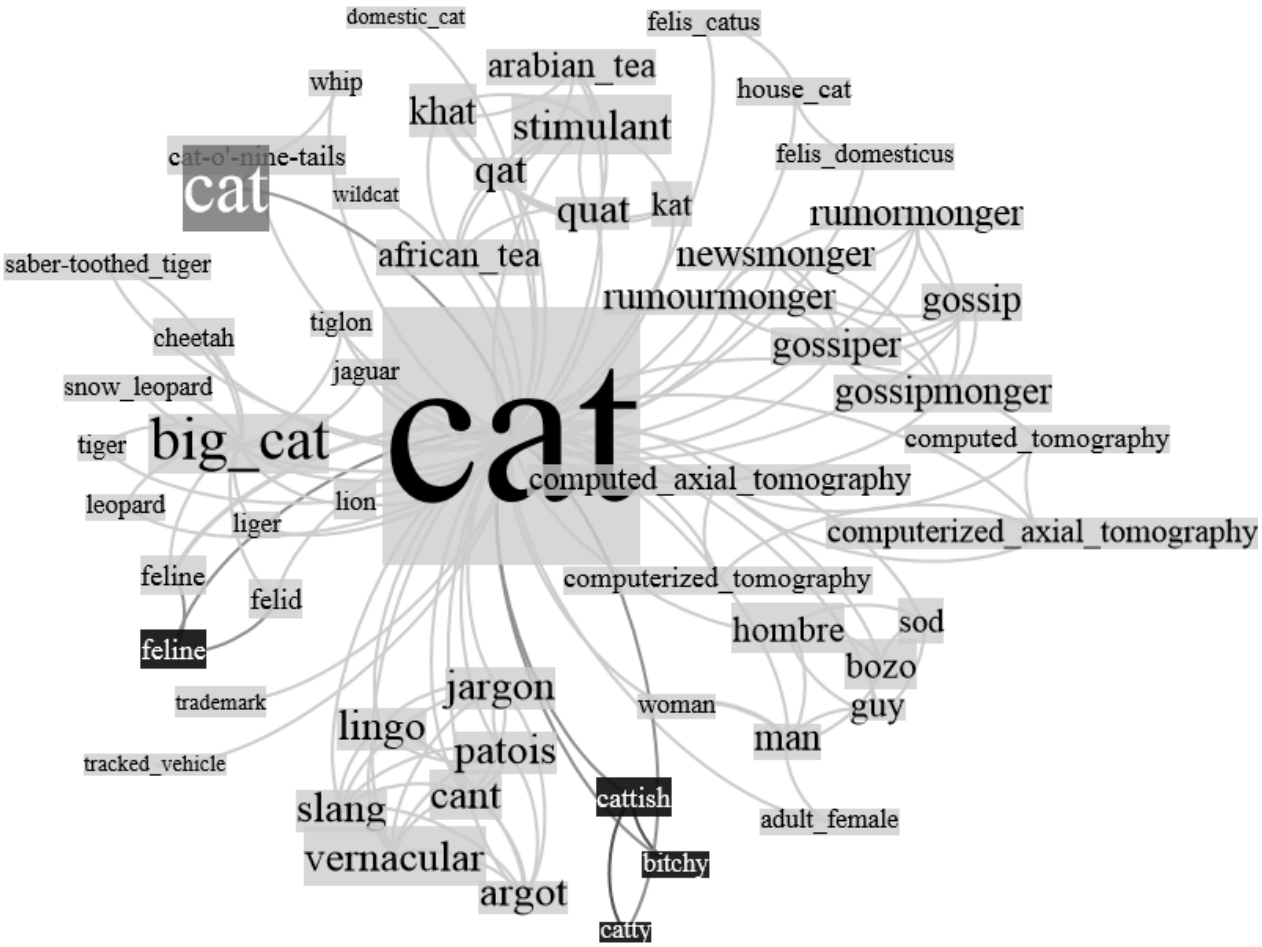


Figure 7.1 : Sample of the full ontology as related to a cat. The size of the nodes represent their degree of connectedness and their color their part of speech class: adjectives (black), adverbs (not shown, white), nouns (light gray) and verbs (gray). The colors of the edges represent the type of relation between nodes. Note that the length of the lines have no meaning other than to improve the clarity of the illustration.

to smooth out zero counts, we can use the information from known $C(w_n|h)$ counts to generate semantically probable unseen $C(w_r|h)$ smoothed counts.

We believe that this is a reasonable assumption, especially for spoken dialogue where speakers can interchange words for related terms that come to mind more quickly. Furthermore, since this work seeks to help reduce the data sparseness of counts that are used for smoothing we propose the following technique based on interpolation to smooth low-resource language models by intelligently adding information not seen before.

7.7.1 Bin-Based Ontological Smoothing

The Bin-Based Ontological Smoothing Language Model (BBOSLM) smoothing technique consists of creating bins B_d of smoothed counts taking into account all words found at a distance d, the shortest path (number of edges) in the ontology from words w_n and w_r, from the original *n*-gram counts and where $d \leq 1 \leq d_{max}$ and d_{max} is the maximal distance from w_n that is considered in the ontology. For example, in figure 1, the distance $d(\text{cat}; \text{tiger}) = 1$, $d(\text{feline}; \text{cattish}) = 2$, etc. More formally, for each $C(w_n|w_{n-N+1}^{n-1}) \in B_d$:

$$C(w_n|w_{n-N+1}^{n-1}) = \frac{1}{R}\sum_{r=1}^{R_d}\frac{C(w_r|w_{n-N+1}^{n-1})}{d} \tag{7.2}$$

where R is the total number of words related to w_n up to d_{max}. In practice, a vector of related words is pre-computed at the start of the program using a modified form of depth-limited iterative deepening depth-first search. As such, R is the number of linked terms for each word and $R_d \leq R$ is the number of related terms at distance d. The premise behind the use of the shortest path as a measure of conceptual or semantic distance is illustrated in [91], namely that "*...when is-a hierarchies are defined ... shortest path length can be used to measure conceptual distance between concepts*". The ontologically generated counts are then used to generate language models for each bin LM_{B_d}. Finally, these language models are interpolated into the final smoothed BBOSLM:

$$\begin{aligned} P'(w_n|w_{n-N+1}^{n-1}) = \lambda_1 P(w_n|w_{n-N+1}^{n-1}) \\ + \sum_{d=1}^{D}\lambda_{d+1}P_{B_d}(w_n|w_{n-N+1}^{n-1}) \end{aligned} \tag{7.3}$$

where $D = d_{max}$ is the maximum edge distance from the original in the ontology, $P_{B_d}(w_n|w_{n-N+1}^{n-1})$ are the probabilities obtained from the smoothed counts in each bin and where λ are the mixture weights and

$$\sum_{i=1}^{D} \lambda_i = 1$$

are obtained with the help of Powell optimization.

7.7.2 Ontological smoothing algorithm

The first step in the ontological smoothing process is to convert the ontology to an efficient graph format for processing. Given an ontology O (WordNet in this case) and part of speech classes *POS*, we convert the ontology to a directed graph G where each node N is a lexeme (lemma + forms). Each node can be from multiple classes, e.g. cat is both a noun and a verb and each edge E represents a relation between two nodes.

The second step is to tag our corpus's terms by their part-ofspeech. For this task, we used the Penn-State TreeTagger [137]. This is to preserve the meaning of the *n*-gram counts. For example, we potentially wouldn't want to smooth the *n*-gram entry of a verb with that of a noun. As per [91] we restrict the search space to synonymous terms for this work. Once the corpus is tagged, we proceed with the counting task using the SRILM toolkit [93].

Our algorithm operates in D passes : each pass creates a bin containing the smoothed counts for distance d using (7.2). Each pass also adds unseen $w_r|h$ events in order to reduce the effects of limited corpus. Since the number of new counts generated can be particularly high given a large distance metric, we parallelized our algorithm. This gives us a speed-up of up to 52% on four threads with $d_{max} = 5$ for the whole smoothing process.

7.7.3 Ontology search algorithm for R related terms

Since the search space of the ontology is considerable we needed an efficient algorithm to look for the related terms of each ngram entry. The fact that we use an unweighted graph enabled us to avoid the inherent challenges of searching weighted directed graphs. The algorithm is a depth-limited form of unweighted iterative deepening depth-first search. Basically, as we progress

through the graph each relation is marked gray; we then recursively explore each unmarked child node (we do not return to it since it has already been seen at a lower level), and thus the "shortest" path to it is already known. This greatly reduces the cost of exploring the ontology all the while guaranteeing optimal paths for each relation.

7.8 Experiments and results

7.8.1 Limited resources experimental setup

The experiments reported in this paper were performed using the Wall Street Journal-based Continuous Speech Recognition Phase II corpus (WSJ1). We train our language models on the 76,136 sentences provided by the standard WSJ1 training sets for a total of 1,243,340 words. For the evaluation we restricted ourselves to the Hub 1 and Spoke 1 tests (10,347 words in 582 sentences) without filtering out sentences with out-of-vocabulary words (OOV rate of 5.41%).

The SRILM toolkit and HTK toolkit [119] were used for our experiments for language modeling and automatic speech recognition respectively. We obtain raw training counts using the SRILM toolkit. These raw counts are then used to generate ontologically smoothed count bins for each distance group, note that, as per eq. (7.2), the majority of the counts are fractional counts. Then, WittenBell LMs were created for each bins. Finally, a mixture-based language model is created by mixing up each bin-based LM with the original Witten-Bell LM. Optimal mixup weights were obtained through the use of Powell's optimization algorithm [101] using the SciPy library [118] with the objective function defined as the word error rate of the held-out development set consisting of 4,340 sentences for a total of 71,759 words. Unfortunately, the SRILM toolkit does not support fractional counts for Kneser-Ney smoothing and as such, we cannot properly interpolate the baseline KN model with our bins.

Perplexity and word error rate (WER) are used to measure the performance of our language modeling technique. For the acoustic models, we use the well-known recipe from Vertanen [120]. Our model is trained by using all of the WSJ1 training data using the 40 phones set of the CMU dictionary. Cross-word tied-state triphones with 32 Gaussian mixtures per state and 64 Gaussian mixtures per silence state are eventually generated. The acoustic models are represented by Mel Frequency Cepstral Coefficients with energy, delta and acceleration (*MFCC_E_D_A*) for a total of

a 39-dimensional feature vector. The HDecode tool with parameters for the pruning beam width, word insertion penalty, and the language model scale factor of 220.0, -4.0, and 15.0 respectively were used for the automatic speech recognition test.

7.8.2 Perplexity evaluation

Perplexity evaluation was performed on the Hub 1 (read WSJ baseline) and Spoke 1 (language model adaptation) portion of the WSJ1 corpus which consist of 582 sentences and 10,347 words. The standard 20,000-word WSJ closed-vocabulary was used for the evaluation and backed-off trigram language models were generated using SRILM with the `-float-counts` option enabled.

The average path length of the ontology is 59.24 with a network diameter of 15 measured using the Gephi toolkit [144]. The network diameter is a measure of the maximum distance between any two pairs in the graph. Since it would be too computationally intensive to fully explore each node, we chose to study the effects of d_{max} from 1 to 5. From the $\sim$346,000 original raw counts we end up with $\sim$15,284,000 smoothed counts for B_5. Figure 7.2 shows the evolution of the language model perplexity throughout the Powell optimization algorithm for the Bin-based ontological smoothing against the baseline models trained using Witten-Bell (WB) smoothing and the original Kneser-Ney (UKN) smoothing. Note that we were not able to include results for Modified KneserNey (MKN) smoothing as the SRILM toolkit kept computing negative discount parameters for the trigrams. Results for both single and multiple bins mixup are provided. Figure 7.2 shows a summary of the best results obtained. As was expected, our technique is able to significantly reduce the perplexity of the baseline language models with the best model showing maximal perplexity reduction of 9.85% for the WB LM. We were able to get very close to the perplexity of Kneser-Ney discounting for Bins $1 \leq d \leq 5$ (Table 7.1). However this is to be expected as the training

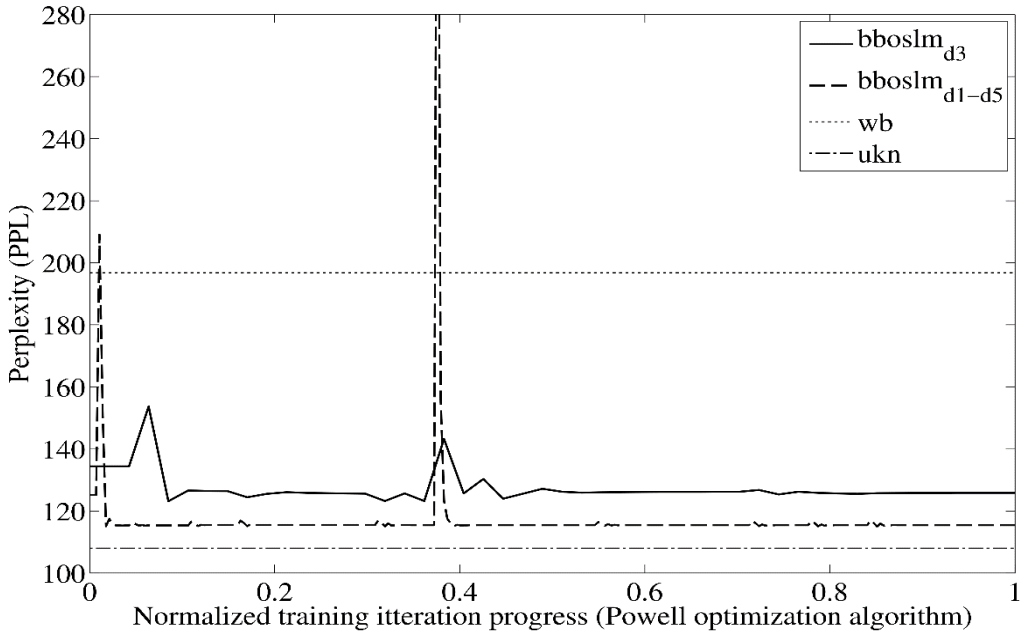


Figure 7.2 : Perplexity evolution during the Powell WER optimization process to determine the best BBOSLM bin mixup weights for 20K word trigram LMs trained on the WSJ1 training corpus and conditioned the WSJ1 development corpus. The spikes are due to the algorithm trying to see if it is in a local optimum.

process did not explicitly look at minimizing perplexity but rather minimizing Word Error Rates. This is consistent with the literature as it is well known that a reduction in perplexity will not always translate in improvements in automatic speech recognition tasks. The spikes in perplexity are due to the Powell algorithm exploring the bounded space to see if it is not currently in a local optimum. This is encouraging since it demonstrates that our technique can compete with state-of-theart language modeling techniques.

Table 7.1 : Comparison of perplexity on WSJ1 Hub 1 and Spoke 1 evaluation corpus using 20K word trigrams using best WER performing single and multiple bins mix-up.

Language Model	Perplexity
Witten-Bell	142.61
Kneser-Ney	117.15
BBOSLM	-
$d = 1$	137.46
$d = 2$	137.5
$d = 3$	137.92
$d = 4$	137.54
$d = 5$	137.53
$1 \leq d \leq 2$	137.14
$1 \leq d \leq 3$	128.55
$1 \leq d \leq 4$	129.45
$1 \leq d \leq 5$	**122.49**

7.8.3 Automatic speech recognition experiments

This experiment investigates the performance of the BBOSLM on word error rate in an automatic speech recognition task. Much like the perplexity evaluation, we provide the WER results for single bin mixup and multiple bins mixup in relation to the LM perplexity optimization process. The ptimization process was run three times with evenly distributed starting λ values, Figure 7.3 shows the optimization process for the best performing final models and starting with $\lambda_0 = 0.9$. Final λ_1 from eq. (7.3) values for the two best performing LMs, $BBOSLM_{d3}$ for single bin mixup and $BBOSLM_{d1-d5}$ for multiple bin mix-up, are 0.9962 and 0.8913 respectively. Globally, results show (Table 7.2) that the best performing LM are the ones mixing-up multiple smoothed counts bins. These results are confirmed by every statistical significance test available in the NIST Scoring Toolkit: a statistically significant reduction (mean $p = 0.001$) in word error rate of 6.8% relative for $BBOSLM_{d1-d5}$ and statistically significant reduction (mean $p = 0.01$) for $BBOSLM_{d4}$ with a

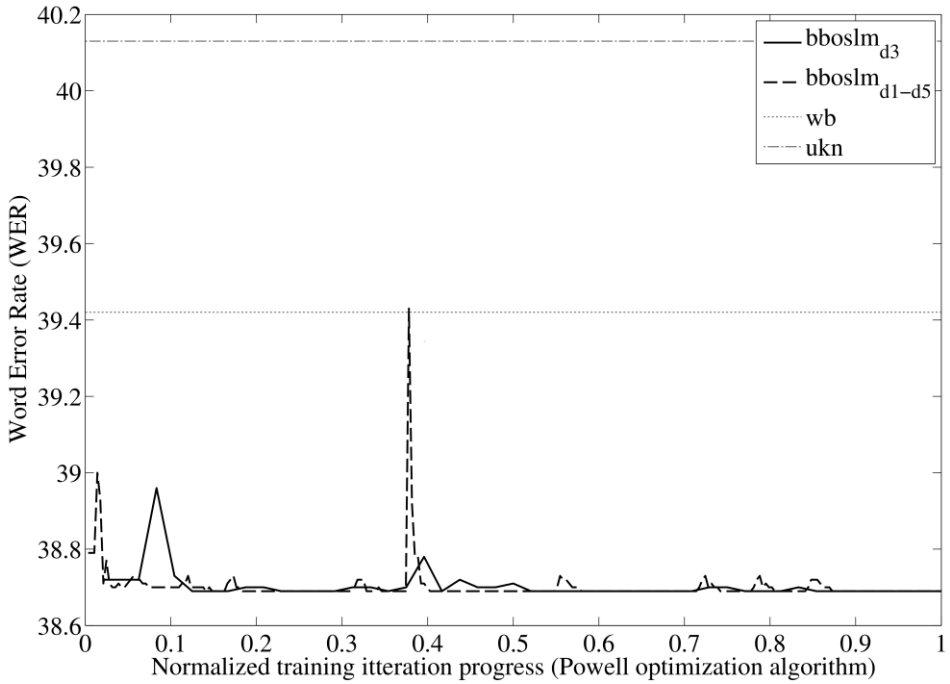


Figure 7.3 : WER (%) evolution of the BBOSLM during the WER optimization process on WSJ1 development corpus using 20K word trigrams. The "large" spike in the training process is due to the Powell algorithm jumping in λ values to see if it is in a local optimum.

Table 7.2 : ASR performance comparison on WSJ1 Hub 1 and Spoke 1 evaluation corpus using 20K word trigrams using single and multiple bins mix-up.

LMS	***PPL***	***SUB***	***DEL***	***INS***	***WER***
WBLM	142.61	22.43	5.89	2.64	30.96
KNLM	117.15	21.87	7.71	1.41	30.98
BBOSLM	-	-	-	-	-
$d = 1$	137.46	21.2	6.65	1.72	29.56
$d = 2$	137.5	21.13	6.67	1.77	29.57
$d = 3$	137.92	20.99	6.67	1.78	29.44
$d = 4$	137.54	21.08	6.73	1.74	29.55
$d = 5$	137.53	21.05	6.79	1.73	29.57
$1 \leq d \leq 2$	137.14	21.18	6.65	1.73	29.56
$1 \leq d \leq 3$	128.55	20.69	6.76	1.58	29.03
$1 \leq d \leq 4$	129.44	20.72	6.74	1.6	29.06
$1 \leq d \leq 5$	122.49	20.23	7.03	1.59	28.85

considerable reduction in insertion and substitution rates compared to the baseline Witten-Bell discounted LM.

7.9 Conclusion and discussion

In this paper, we introduced an original method of preprocessing and smoothing n-gram counts based on the semantic information found in ontologies for use in limited resources automatic speech recognition. It exploits the relations found in ontologies to create bins of new n-gram events related to existing n-gram events. These bins are transformed into language models (LM) and then mixed-up with the original LM. Experiments demonstrate that our smoothing technique reduced the perplexity of a given language model by up to 9.85% when considering bins of related wordsw ith a maximal ontological distance of up to 5. In addition, we evaluated our language models on the Hub 1 and Spoke 1 tasks of the Wall Street Journal-based Continuous Speech Recognition Phase II corpus. We showed that it reduced word error rate (WER) in a statistically significant manner compared to Kneser-Ney discounting ($p = 0.001$). The fact that we were able to obtain statistically significant performance increase indicate that it shows promise as a tool for language model smoothing, especially when textual resources are sparse. Future work will look to incorporate different ontological distance metrics such as those proposed by Jing *et al.*[145] as well as exploring information theory based schemes to weight the different relations between words described by the ontology.

8 EVALUATION OF GRAPH METRICS FOR OPTIMIZING BIN-BASED ONTOLOGICALLY SMOOTHED LANGUAGE MODELS

Évaluation de l'utilisation des mesures de graphes pour l'optimisation des modèles de langage basé sur le lissage par groupements ontologiques

Y. Benahmed[1,2]*, S-A. Selouani*[2] *et D. O'Shaughnessy*[1]

[1]INRS-EMT, 800 de la Gauchetière O, H5A 1K6, Montréal, Québec, Canada
[2]LARIHS Lab. Université de Moncton, campus de Shippagan, New-Brunswick, Canada

Article soumis à : *EUSIPCO 2016*

Lieu de la conférence : Budapest, Hongrie

Accepté

Date d'acceptation : 28 mai 2016

8.1 Contribution des auteurs :

Yacine Benahmed: (candidat)

Écriture de l'article, développement des logiciels et programmes, expérimentation, analyse des données, préparation des graphes, des figures et des tableaux.

Douglas O'Shaughnessy: (directeur)

Conseils techniques et théoriques, conseils de rédaction, édition, coauteur, corrections.

Sid-Ahmed Selouani: (codirecteur)

Conseils techniques et théoriques, édition, coauteur, corrections.

8.2 Commentaires des évaluateurs

La majorité des commentaires des évaluateurs ont été pris en considération lors de la révision finale de la version de l'article présenté dans cette section.

8.2.1 Évaluateur no 1

Comments and Recommended Changes:
In this work, the authors present a continuation of previous work regarding bin-based ontologically smoothed language models (BBOSLM), which have been first published at ICASSP 2014. The additions include the evaluation of various graph measures for distance calculations between words in BBOSLM, evaluating the entire powerset of bins for interpolation with the background LM, as well as testing the approach on more corpora/data.

While incorporating the various graph measures did lead to slight regressions over the unmodified BBOSLM, it is still an interesting approach. By evaluating the whole powerset of bins, it could be shown that certain bins do not contribute to gains. Overall, significant improvements in WER over the baseline model could be achieved.

It might be interesting to see the WER results of the different methods for the single evaluation corpora. Maybe a certain method/graph measure improves on a certain part of the test data?

To better asses the quality of the ASR baseline, it would be helpful to quote state-of-the-art results from other groups as well, if they are accessible.

8.2.2 Évaluateur no 2

Comments and Recommended Changes:
This work presents a set of methods for optimizing bin-based ontologically smoothed language models (BBOSLM). The main novelty in this work is the use of different graph measures (Weighted degree, HITS, PageRank, Modularity) to optimize ontology distances between words when estimating the BBOSLM.

Bin-based ontological smoothing is a relatively new approach within language modeling. Therefore it is nice that the authors devoted a section for explaining this method separately.

In terms of results, BBOSLM offers some improvements (WER,perplexity) compared to standard smoothing methods such as Witten-Bell and Kneser-Ney. The graph measures, which were the novelty in this work, did not give any additional improvements compared to the original BBOSLM approach.

How many bin configurations were evaluated in the ASR experiments? The difference in performance between the different BBOSLM models seem quite small. It would be beneficial to also include statistical significance tests. It would also be good if the authors could more specifically elaborate how graph measures could still be useful for improving performance of BBOSLM models.

8.2.3 Évaluateur no 3

Comments and Recommended Changes:
The differences in perplexity are rather small, as well as the differences in WER. Moreover better error rates are obtained with models having a perplexity worse than that of the baseline (KNLM).

Hovewer, only selected best results appear in Table II, and most of thme correspond to combinations that do not appear in Table 1.

This let the reader wonder on the significance and reliability of the results. No confidence interval, nor any significant test are reported. Would the same 'best' configuration also be the best on another data set?

Is exactly the same behavior observed on the various subsets (spoke 1, spoke 2 and spoke 9)?

8.2.4 Évaluateur no 4

Comments and Recommended Changes:
The authors propose in this article to use graph metrics to capture additional semantic or relational information found in ontologies. The paper is well written, enough information being given to understand the general idea of the paper. The paper is well written. I particularly appreciated the description of all training parameters.

I have few questions regarding the paper :

- Could the authors provide the best results results (in terms of perplexity and WER) of the considered corpus?

- Could the authors give more information about the reason why not that particular bins (1, 3 and 5) perform better WER than all the bins? Regarding to perplexity, this is not observed since all bins used together allows to get the lowest perplexity, why?

Are the gains significant?

8.3 Résumé

Cet article examine l'utilisation de mesures tirées de la théorie des graphes pour améliorer les performances d'un algorithme de lissage de modèle de langage. Le *Bin-Based Ontological Smoothing* a été utilisé avec succès pour améliorer la performance de modèles de langage dans des tâches de reconnaissance automatique de la parole. Cette technique utilise les ontologies pour estimer la vraisemblance de nouvelles expressions pour un modèle de langue donné. Comme les ontologies peuvent être représentées sous forme de graphes, nous étudions l'utilisation de paramètres des graphes comme facteur de lissage supplémentaire afin de saisir l'information sémantique ou relationnelle intrinsèque aux ontologies. Plus précisément, nous étudions l'effet des mesures HITS, PageRank, Modularité et degré pondéré sur la performance de la reconnaissance automatique de la parole, de même que par rapport à la perplexité des modèles de langage. L'ensemble complet de l'interpolation des groupements est évalué. Nos résultats montrent que l'interpolation des bacs originaux à des distances 1, 3 et 5 a donné lieu à une amélioration du taux d'erreur des mots de 0,71 % par rapport à l'interpolation des bacs 1 à 5. Finalement, la modularité, le PageRank et l'HITS sont prometteurs et méritent une étude plus approfondie.

Le texte qui suit est une copie exacte de l'acte de conférence référencé ici haut[39].

[39] La numérotation des sections, équations, figures, graphiques, tableaux et références a été adaptée pour correspondre à l'ordre retrouvé dans ce document.

8.4 Abstract

This paper investigates the use of graph metrics to further enhance the performance of a language model smoothing algorithm. Bin-Based Ontological Smoothing has been successfully used to improve language model performance in automatic speech recognition tasks. It uses ontologies to estimate novel utterances for a given language model. Since ontologies can be represented as graphs, we investigate the use of graph metrics as an additional smoothing factor in order to capture additional semantic or relational information found in ontologies. More specifically, we investigate the effect of HITS, PageRank, Modularity, and weighted degree, on performance. The entire power set of bins is evaluated. Our results show that the interpolation of the original bins at distances 1, 3 and 5 resulted in an improvement in WER of 0.71% relative over the interpolation of bins 1 to 5. Furthermore, modularity, PageRank and HITS show promise for further study.

8.5 Introduction

Language modeling is a crucial part of automatic speech recognition (ASR) systems as well as of many other tasks such as part-of-speech tagging, natural language processing, document classification, information retrieval, etc. More specifically for ASR however, language modeling enables recovering the probability of a sentence or sequence of words $S = w_1, ..., w_i$ given an acoustic signal A, that is

$$P(S|A) = \ P(A|S)P(S) \ P(A),$$

where $P(S)$ represents the language model (LM), the probability of the sentence S. The goal of *n*-gram language models is to simplify the prediction of the *i*-th word based on the $n-1$ previous words, essentially a Markov chain of order $(n-1)$, using maximum likelihood estimates (MLE):

$$P_{MLE}(w_i|h) = C(h, w_i)C(h) \ . \tag{8.1}$$

8.5.1 Motivation

Two well-known drawbacks of language modeling with ngrams are its sensitivity to the corpus and data sparseness, i.e., if we train our model on text from Shakespeare, it is evident that the model will not be adequate for the prediction of astrophysical text. That is why multiple approaches exist to either obtain better language models (LM) or smooth existing ones in order to improve their statistical accuracy and significance. Notable techniques include discounting, interpolation and various backoff schemes such as Good-TuringDiscounting [35], [146], Witten-Bell smoothing [36], Modified Kneser-Ney smoothing [36] or Hierarchical PitmanYor language models [138] used to smooth out low-order counts. Another drawback of n-gram language models is that they only can capture natural language semantics to a certain extent, mainly by capturing direct co-occurrences which do not completely reflect the co-occurrence networks from natural language. This is well documented in work by Biemen *et al.* [147]. Furthermore, work by Yan *et al.* [148] provide interesting insights on the use of ontologies in language modeling but only concentrates on document retrieval task. This is why bin-based ontologically smoothed language models (BBOSLM) [86] were proposed to exploit the semantic information found within the WordNet [88] ontology.

Graph theory, or network analysis, has gathered a lot of research interest over the last few decades. It can be seen as a powerful tool to quantify and qualify the importance and influence of nodes within a graph or detect hidden communities of interest. It is widely used in natural language processing for modeling relations between words. It is this ability that we are interested in examining in this work. We propose to study the effect of different network metrics used as additional weighting factors to the distance based similarity measure used in the original BBOSLM.

8.5.2 Contribution

The contribution of this work is the evaluation of the use of graph measures in ontology distance optimisation between words in BBOSLM. Furthermore, whereas previous work restricted itself to evaluating a linear incrementation of bins combination, we extend this work by evaluating the entire powerset of bins interpolated with the original Witten-Bell smoothed language model. That is, we look at the entire power set (155 distinct combinations) of bins (described in Section 8.6

below) interpolated with the original language model. We also extend the evaluation to the Spokes 2 and 9 of the Wall Street Journal Continuous Speech Recognition Phase II corpus for a total of 1285 sentences up from the 582 sentences used in the original work.

The outline of this paper is as follows. In Section 8.6.2, network analysis and the graph metrics of interest are briefly presented. Section 8.6.4 describes the bin-based ontological n-gram count smoothing algorithm with the additional α weighing factor. Section 8.7 proceeds with the description of the experiment set-up and the evaluation of the use of different network metrics. Finally, in section 8.8, we conclude and discuss our results.

8.6 Bin-based ontological smoothing

Bin-based Ontological Smoothing [86] was proposed in order to smooth counts based on the ontological distance d between words found in an ontology, namely the WordNet ontlology. As stated in the introduction, the intuition behind this proposed technique was to exploit the relations found between words in ontologies to try to capture the fact that a given message could be expressed in different ways and that similar messages could be expressed in a similar way. For example, “the dog barked” and “the bird sang” are both very similar and different at the same time. For this, we consider the relations between words in the WordNet as an “is-a” hierarchy. This makes it possible to assume that the shortest path between words can be used as a simple measure of conceptual/semantical distance [91].

In Bin-Based Ontological Smoothing, we begin by generating smoothed bins $\mathrm{B} = \{B_1, B_2, \ldots, B_{d_{max}}\}$ of smoothed *n*-gram counts that take into account all of the related words $\{(w_i, w_r) : \min\big(d(w_i, w_r)\big) = d\}$, completing the ngram context w_{i-1}^{i-n+1}. Then, let

$$B_d = \Big\{(w_i^{i-n+1}, w_r | w_{i-1}^{i-n+1}) : min\{d(w_i, w_r) = d\}\Big\}$$

be the bin B_d of smoothed n-grams where w_i^{i-n+1} is the original *n*-gram, w_r w_{i-1}^{i-n+1} is the set of related *n*-gram with ontological distance d between the original word w_i and related word w_r. We than generate smoothed counts following equation (8.2):

$$C_{B_d}(w_i|w_{i-1}^{i-n+1}) = \frac{1}{dR}\sum_{w_r} C(w_r|w_{i-1}^{i-n+1}) \tag{8.2}$$

where R is the total number of words related to w_i up to the maximal distance d_{max} such that $R = |\{(w_i, w_r)\colon d(w_i, w_r) \leq d_{max}\}|$ and d is the current distance. In practice we pre-compute a vector of related words for each w_i up to the maximum distance of interest d_{max} with a modified depth-limited iterative deepening depth first search. This greatly reduces the computational complexity of generating the smoothed counts. We then use those new smoothed counts to derive statistical language models using available complimentary smoothing techniques. However, previous work [86] found that Witten-Bell smoothing produced better overall results than Kneser-Ney-based smoothing. Finally, these language models are linearly interpolated into the final smoothed Bin-based Ontological Smoothed LM:

$$P'(w_i|w_{i-1}^{i-n+1}) = \lambda_0 P(w_i|w_{i-1}^{i-n+1}) + \sum_{d=1}^{d_{max}} \lambda_d P_{B_d}(w_i|w_{i-1}^{i-n+1}). \tag{8.3}$$

where d_{max} is the maximum edge distance from the original in the ontology, $P_{B_d}(w_i|w_{i-1}^{i-n+1})$ are the probabilities obtained from the smoothed counts in each bin and where λ are the mixture weights and

$$\sum_{i=0}^{d_{max}} \lambda_i = 1$$

can be obtained with the help of any linear optimization algorithm.

8.6.1 Ontological smoothing algorithm

The first step in the ontological smoothing process is to convert the ontology to an efficient graph format for processing. Given an ontology O (WordNet in this case) and part of speech classes POS, we convert the ontology to an undirected graph G where each node N is a lexeme (lemma + forms). Each node can be from multiple classes, e.g. `cat` is both a noun and a verb and each edge E represents a relation between two nodes.

The second step is to tag the corpus's terms by their part-of-speech. For this task, we used the Penn-State TreeTagger [137]. This is to preserve the meaning of the n-gram counts. For example,

we want to avoid smoothing the ngram entry of a verb with that of a noun. As per [91] we restrict the search space to synonymous terms for this work. Once the corpus is tagged, we proceed with the counting task using the SRILM .

The smoothing algorithm then operates in dmax passes: each pass creates a bin B_d containing the smoothed counts for distance d using (8.2). As stated in section II each pass also adds unseen $w_r|h$ events in order to smooth zero count *n*-grams.

8.6.2 Network Analysis

Graph theory or network analysis has gathered a lot of research interest over the last few decades. It can be seen as a powerful tool to quantify and qualify the importance and influence of nodes within a graph or detect hidden communities of interest. Its use in many areas of research, be it biology, sociology, economy etc. speaks volumes of its versatility in modeling relations. It is this ability that we are interested in studying in this work. We propose to study the effect of different network metrics used as additional weighting factors to the distance based similarity measure used in the previous work. More specifically, we investigate if using HITS Centrality, PageRank Centrality, Modularity, and a baseline weighted degree can improve the performance of the smoothing method.

8.6.2.1 Weighted Degree

Degree and its ponderated form, Weighted degree, are the simplest measures of centrality in graphs. It essentially counts the number of edges incident to a node and is defined as:

$$WD_i = \sum_j A_{ij}. \tag{8.4}$$

where WD_i is the weighted degree of edge i, the sum of the weight of all edges attached to it, and A_{ij} is the edge weight between nodes i and j. This equation can easily be rewritten with respect to a word network as follows:

$$WD(w_i) = \sum_j A(w_i, w_j). \tag{8.5}$$

That is, $WD(w_i)$ is the weighed degree of word w_i and $A(w_i, w_j)$ is the edge weight between words w_i,w_j.

8.6.2.2 HITS

The HITS (Hyperlink-Induced Topic Search) algorithm [96] rates Web pages by analysing their links. It gives a list of hub and authority centralities for each node in a graph. The relation between hubs and authorities is maintained by the mutually recursive algorithm

$$auth(p) = \sum_{i=1}^{n} hub(i)\,, \quad hub(p) = \sum_{i=1}^{n} auth(i), \tag{8.6}$$

which can be rewritten as:

$$auth(w_i) = \sum_{w_j} hub(w_j)\,, \quad hub(w_i) = \sum_{w_j} auth(w_j), \tag{8.7}$$

where $auth(w_i)$ and $hub(w_i)$ are the authority and hub scores of word w_i based respectively on the hub and authority score of connected words w_j (inbound for authority and outbound for hub). That is, good hubs point to authoritative words and authoritative words are pointed by good hubs. Each node starts with a value of 1 for both authority and hub, we update $auth$ values followed by hub values. They are then are normalized respectively by the square root of the sum of their squares and the process repeats until the variation falls bellow a set threshold.

8.6.2.3 PageRank

The PageRank [97] algorithm is an iterative algorithm that gives the probability of a random surfer clicking its way randomly to any given web page. We assume word w_i has words w_j ... w_{j+n} which point to it. The PageRank of a word w_i is then given as follows:

$$PR(w_i) = \frac{1-\delta}{N} + \delta \left(\sum_{w_j \in M(w_i)} \frac{PR(w_j)}{L(w_j)} \right), \tag{8.8}$$

where $PR(w_i)$ is the PageRank of word w_i and $L(w_j)$ is the number of words linked by w_j. The parameter δ is a damping factor which can be set between 0 and 1 and is usually set to $\delta = 0.85$, $w_j \in M(w_i)$ is defined as the set of words linked by word w_i and N is the total number of words in the vocabulary.

One advantage of the PageRank algorithm is that it can also be applied to undirected graphs. The assumption of this work is that a low PageRank suggests a low probability of substitution of word w_i for word w_j and is then used as a weighing factor with $0 \leq PR \leq 1$.

8.6.2.4 Modularity

Modularity is a technique used for the detection and characterization of community structure in any given network. The quality -1 $\leq Q \leq$ 1 of such communities c is defined by their Modularity defined in [98] such that:

$$Q = \frac{1}{2m} \sum_{w_i, w_j} \left[A_{ij} - \frac{D(w_i)D(w_j)}{2m} \right] \sigma \left(c(w_i), c(w_j) \right), \tag{8.9}$$

where A_{ij} is the edge weight between words w_i and w_j, $D(w)$, the (unweighted) degree of word w, the σ-function $\sigma(u, v)$ is 1 if $u = v$ and 0 otherwise (in other words, 1 if they are in the same community, 0 otherwise). For this work, modularity is used to penalize words that are from different communities c, that is, when $c(w_i) \neq c(w_r)$. We use it to try to uncover unknown topics or lemma classes within the ontology and penalize counts from words in different communities.

8.6.3 The WordNet ontology

The WordNet [88] ontology is used in this work. It is a essentially a large lexical database of English. The main relation between defined words is synonymy, the state of being a synonym [88]. It was chosen due to it's well documented nature and widespread use in the literature.

8.6.4 Weighting adjustments

The basis for this work is the hypothesis that less important words (essentially less connected words) should contribute less to smoothing counts of n-grams and more important words should contribute more. For this work, we consider the ontology as an undirected graph and we use the previously mentioned measures from network theory to extract meaningful information. It is then used to compute adjustment weights from an α-function for each ontological relation between words w_i and w_r. As such, this work proposes the following modification to the original equation (8.2):

$$CB_d(w_i|w_{i-1}^{i-n+1}) = \frac{1}{dR}\sum_{w_r}\alpha(w_r)C(w_r|w_{i-1}^{i-n+1}), \tag{8.10}$$

where $\alpha(w_r)$ is the adjustment weight for related word w_r for the given network metric function. It represents the importance of w_r within the ontology. Furthermore, for the special case of Modularity, we define the α-function as such:

$$\alpha_{Mod}(w_i, w_r) = \begin{cases} 1, \text{if } c(w_i) = c(w_r) \\ \frac{1}{e}, \quad \text{otherwise} \end{cases}.$$

All graph metrics were computed using Gephi [144] and as such, the standard algorithms for HITS ($\epsilon = 1.0E-4$) and PageRank (damping factor $\delta = 0.85$) were used. A notable exception being the Modularity computation, which uses the fast algorithm for unfolding communities proposed by Blondel *et al.* [149]. It is an efficient algorithm that results in high modularity partitioning. To speed up computation time, all properties are pre-computed and stored in a hashkey based binary storage matrix.

8.7 Experiments and results

8.7.1 Experimental Setup

The experiments reported in this paper were performed using the Wall Street Journal-based Continuous Speech Recognition Phase II corpus (WSJ1). The training of our models uses the 76k sentences provided by the basic WSJ1 training sets for a total of 1.2M words. For the evaluation we used the Hub 1 (read WSJ baseline), Spoke 1 (language model adaptation), Spoke 2 (domain-independance) and Spoke 9 (spontaneous WSJ dictation) test data (23k words in 1285 sentences) without filtering out sentences with outof-vocabulary words. The standard 20K words WSJ closed vocabulary (OOV rate of 5.64%) was used for the evaluation and backed-off trigram language models were generated using SRILM with the -float-counts option enabled for the bins.

The SRILM toolkit [93] and HTK toolkit [119] were used for language modeling and automatic speech recognition respectively. We first generated the raw training counts using the SRILM toolkit. These raw counts are then used to generate ontologically smoothed floating point raw count bins for each distance group. Then, Witten-Bell LMs were created for each bin. Finally, a mixture-

based language model was created by mixing up each bin-based LM with the original Witten-Bell LM. Mixup weights were obtained through the use of Powell's optimization algorithm [101] using the SciPy library [118] with the objective function defined as the perplexity (PPL) of the held-out development set, consisting of 4.3k sentences. We chose the Powell optimization algorithm instead of the EM algorithm provided by compute-best-mix script from the SRILM toolkit because of empirical observations where we consistently obtained better results by using the Powell algorithm. Secondly, the obtained mixture weights do not achieve the same level of performance (in terms of perplexity and ASR (without bayes)) that we get from the use of the Powell algorithm. Unfortunately, the SRILM toolkit does not support float counts for Kneser-Ney modeling and as such attempts to mix KN-based bins with the original KN LM resulted in severe performance degradation.

Perplexity (PPL) and Word Error Rate (WER) are used to compare the performance of the different language models. For the acoustic models, we used the recipe from Vertanen [120] with additional discriminative training (MPE criterion). Our model is trained by using all of the WSJ1 training data using the 40 phones set of the CMU dictionary. Full details of the configuration used can be found in [86]. The HDecode tool with parameters for the pruning beam width, word insertion penalty, and the language model scale factor of 220.0, -4.0, and 15.0 respectively were used for the ASR test.

The average path length of the ontology is 59.24 with a network diameter of 15 measured using the Gephi toolkit. The network diameter is a measure of the maximum distance between any two pairs in the graph. Since it would be too computationally intensive to fully explore each node, we study bins with words that are up to $d_{max} = 5$. From the $\sim$ 346k original raw counts we end up with $\sim$ 15.3M smoothed counts for $d_{max} = 5$.

8.7.2 Perplexity evaluation

Table 8. shows a summary of the best results obtained (due to the number of different combinations (155), only the best results are presented). These results show that our technique is able to significantly reduce the perplexity of the baseline Witten-Bell smoothed LM (WBLM) with the best model showing a maxium perplexity reduction of 9% relative for the interpolation of bins B1 to B5 and with only a difference of 1.2% relative to Kneser-Ney smoothing. Results from the LMs

using graph metrics closely match the original results. The Authority and PageRank centrality slightly reduce perplexity compared to the original for single bin interpolation at $d = 4$ and produce equivalent results to each other. We believe that this is due to the similarity (mean difference of 1.79E-05) between the measures obtained from both algorithms. Modularity shows particular interest since it performs better than the original counts from *B3* to interpolation of bins *B1,B2,B3,B4*. As for LMs with bins adjusted with the weighted degree measure, their performance is worse across the board, with a perplexity score higher than the original WBLM.

8.7.3 Automatic speech recognition experiments

This experiment evaluates the effect of the graph measures as well as the exploration of the entire powerset on BBOSLMs in the context of automatic speech recognition.

We provide only the best results in Tables 8.1 and 8.2 for each graph metric as well as the unmodified BBOSLM with $\alpha(m) = 1$. Results from the LM combining unmodified bins 1 to 5 are also included for comparison purposes. The BBOSLM using only the three bins *B1, B3, B5* gave the best results with a WER of 23.93% compared to a WER of 24.10% for the original BBOSLM using all five bins. This strongly suggests that not all bins contribute positively to the final language model. It is also interesting to see that better performance can be achieved by using fewer data. By

Table 8.1 : Comparison of perplexity on wsj1 hub 1, spoke 1, spoke 2 and spoke 9 evaluation corpora using 20k word trigrams using single and multiple bins mix-up for each network metric.

LMS	BBOSLM	BBOSLM Auth	BBOSLM Mod	BBOSLM PR	BBOSLM WD	WB	KN
original	-	-	-	-	-	149.11	**134.65**
$d = 1$	147.01	147.01	148.44	147.01	160.24	-	-
$d = 2$	145.11	145.11	147.65	145.11	159.47	-	-
$d = 3$	144.71	144.71	142.85	144.71	157.12	-	-
$d = 4$	142.66	142.66	141.86	142.66	156.08	-	-
$d = 5$	142.87	142.83	141.63	142.83	158.4	-	-
$1 \leq d \leq 2$	139.26	139.18	140.39	139.18	158.37	-	-
$1 \leq d \leq 3$	138.55	138.55	137.56	138.55	158.57	-	-
$1 \leq d \leq 4$	137.41	137.5	**136.27**	137.5	156.47	-	-
$1 \leq d \leq 5$	136.31	136.33	137.24	136.59	155.76	-	-

Table 8.2 : ASR performance comparison of the best performing LMs on WSJ1 Hub 1, Spoke 1, Spoke 2 and Spoke 9 evaluation corpora using 20k word trigrams using single and multiple bins mix-up.

LMS	PPL	SUB	DEL	INS	WER
WBLM	149.11	17.96	**4.32**	3.14	25.42
KNLM	**134.65**	17.88	5.63	2.29	25.79
BBOSLM					
$bins_{d1-d5}$	136.31	16.61	5.27	**2.22**	24.1
$bins_{d1,d3,d5}$	137.36	**16.53**	5.17	2.23	**23.93**
$aut_{d1,d3,d5}$	138.11	16.62	5.17	**2.22**	24.01
$mod_{d1,d5}$	137.65	16.58	5.14	2.26	23.98
$pr_{d1,d3,d5}$	138.11	16.62	5.17	**2.22**	24.01
$wdeg_{d5}$	158.4	17	4.88	2.43	24.31

looking at the other best performers, we see that the trend is maintained in the sense that bins *B1, B3* and *B5* are used in the majority of best performers. This behaviour deserves further investigation in future work. Even though the α weighted BBOSLMs did not outperform the best LM, they still perform better than both Witten-Bell and Kneser-Ney smoothed language models. Like with the perplexity experiments, weighted degree bins perform worse than the other graph metrics, yet this results in smaller word error rate than both KN and WB smoothing using only bin 5. We suspect that it might be symptomatic of a certain amount of robustness built into the BBOSLM smoothing algorithm.

8.8 Conclusion and discussion

In this paper, we evaluated the impact of using different graph metrics to weigh the relations between words in the WordNet ontology for use with the Bin-Based Ontological Smoothing (BBOS) proposed in previous work [86]. We observed from our experiments that the hypothesis that less important words (network node) should contribute less to the overall smoothed counts of bins is inconclusive and should be the subject of further study. Although performance using network metrics slightly degraded compared to the best performing unmodified BBOSLM we were still able to obtain better performance than Witten-Bell and Kneser-Ney smoothing. Furthermore, we were able to obtain better performance by reducing the number of bins used during

interpolation from five to three. This seems to indicate that only certain bins contribute positively to the smoothing of counts and again gives us other interesting avenues for future research.

9 CONCLUSION

Ce travail s'est concentré dans un premier temps sur le développement d'un algorithme de lissage de modèles de langage, des *N*-grammes, permettant d'exploiter l'information sémantique latente aux ontologies afin de permettre une meilleure modélisation probabiliste d'énoncés qui ne se retrouvent pas dans le corpus d'entrainement. Comme nous avons souvent tendance, lorsque nous parlons, à substituer un mot exact pour un autre mot qui nous vient à l'esprit plus rapidement, l'objectif était d'attribuer des probabilités aux *N*-grammes non vues basées sur les expressions sémantiquement similaires observées dans les corpus d'entrainement.

Dans un deuxième temps, nous nous sommes attaqué à un autre problème que nous rencontrons en modélisation du langage, soit l'optimisation des paramètres de lissage des modèles de langage. La plupart des travaux recensés se limitent à l'utilisation de l'algorithme Expectation Maximisation (très rapide, mais très approximatif), ou l'algorithme de Powell pour optimiser les paramètres. Or, ces algorithmes produisent soit des optimisations approximatives, soit se prêtent mal à la parallélisation. C'est dans le but d'obtenir des résultats à la fois meilleurs et de manière plus rapide que nous nous sommes penché sur la possibilité d'utiliser les algorithmes évolutionnaires pour optimiser les paramètres de l'algorithme de lissage original que nous avons proposé.

9.1 Conclusions principales

Les ontologies peuvent être utilisées pour rapidement créer un ensemble de règles flexibles lors de la conception d'interfaces vocales de type commande et contrôle.

Il est possible d'utiliser les ontologies pour lisser les modèles de langage basés sur les *N*-grammes :

- Il est possible, à partir d'un petit corpus, d'obtenir un modèle de langage qui prenne un lissage des énoncés jamais vus, mais qui sont sémantiquement similaires à ceux observés dans le corpus d'entrainement.
- Une normalisation de l'algorithme, par exemple pour le rendre compatible avec la méthode de Kneser-Ney, devrait, en principe, nous permettre d'obtenir des gains additionnels.

- Une approche par bacs, où les mots ayant un nombre similaire de « voisins », similaire à l'approche de la méthode Witten-Bell ou Kneser-Ney où les paramètres de lissage des mots ayant un nombre de contextes similaires sont regroupés en bacs, devraient aussi nous permettre de simplifier le modèle et d'améliorer sa performance.
- Nous avons posé les bases pour utilisation dans corpus restreints. Nous sommes d'avis que la méthode de lissage que nous avons proposé dans cette thèse pourra être utilisée par d'autres chercheurs dans des langes où les ressources textuelles sont restreintes.

Les métriques issues de la théorie des graphes sont prometteuses pour la pondération des relations ontologiques :

- Bien que nous n'avons pas obtenu la réduction de perplexité escomptée suite à leur utilisation sous forme de poids de pondération, nous sommes d'avis qu'une intégration plus approfondie de ces métriques, par exemple en ajustant les poids selon un ensemble de plus petits corpus comme pour la classification de documents et en considérant l'ontologie comme un graphe dirigé plutôt qu'un graphe non dirigé et en ajustant les paramètres utilisés pour les calculer, devrait permettre d'obtenir des gains de performance additionnels.

Les algorithmes évolutionnaires, en particulier l'optimisation par essaims particulaires permettent de rapidement optimiser les poids d'interpolation des modèles de langage :

- Les résultats obtenus, en termes de perplexité, à l'aide des algorithmes évolutionnaires sont en moyenne supérieurs à ceux obtenus par l'optimisation par l'algorithme de Powell.
- L'optimisation par algorithme génétique produit des résultats beaucoup plus variables (plus grande variance) que l'optimisation par essaims particulaires, mais, en contrepartie, permet d'obtenir la plus grande réduction absolue de perplexité.
- Nous avons montré qu'il est possible de partir des résultats obtenus avec l'algorithme d'espérance-maximisation, dans le but de simuler un paramétrage initial expert, d'obtenir des résultats beaucoup plus rapidement et avec une moins grande variance.
- Les algorithmes évolutionnaires ont un avantage inhérent en termes de vitesse comparativement à l'algorithme de Powell grâce à la plus grande prédisposition au parallélisme de ces derniers.

- Nous avons montré que les résultats obtenus permettent de réduire le taux d'erreur des mots de façon statistiquement significative dans une tâche de reconnaissance automatique de la parole.

9.2 Contributions principales

Les contributions principales de notre thèse se résument comme suit :

- Nous avons proposé un nouvel algorithme de lissage ontologique exploitant l'information sémantique retrouvée dans les ontologies pour le lissage des modèles de langage et comparaison empirique avec d'autres algorithmes de lissage statistique.
- Nous avons évalué l'utilisation des mesures issues de la théorie des graphes pour pondérer les relations ontologiques pour le lissage des modèles de langage.
- Nous avons comparé la performance d'algorithmes évolutionnaires, algorithmes génétiques et optimisation par essaims particulaires, pour l'optimisation des poids d'interpolation des modèles de langage lissés à l'aide de notre algorithme de lissage ontologique.
- Nous avons conçu une suite logicielle permettant de générer les *N*-grammes lissés :
    - Compatible avec le standard C++11.
    - Optimisation de la vitesse de traitement :
        - Traitement parallèle.
        - Algorithmes efficaces.
- Nous avons produit une suite de procédures d'entrainement pour la reproduction des résultats sous Linux (Python + Bash + Perl + Matlab).

9.3 Travaux futurs

Nous sommes d'avis que bien que l'algorithme de lissage que nous avons proposé permette d'obtenir des gains de performances en termes de réduction de la perplexité des modèles de langage et d'amélioration du taux de reconnaissance automatique de la parole, qu'il demeure que les concepts introduits pourront être approfondis dans le but d'obtenir d'encore meilleures

performances. Nous nous attendons donc à poursuivre nos travaux, en particulier dans les trois domaines suivants :

- Généralisation et ajustement de l'algorithme de lissage afin de simplifier les calculs.
- Généralisation de l'algorithme sur de plus grands contextes : paraphrases.
- Utilisation des ontologies pour améliorer les modèles de langage basés sur les réseaux de neurones, en particulier les modèles basés sur le *long term short memory*.
- Approfondir l'évaluation de l'effet du taux de mutation de l'optimisation par algorithmes génétiques sans amorçage. Les résultats obtenus ont privilégié un taux de mutation de 80 %, ce qui est contre-intuitif. Nous soupçonnons le processus de génération aléatoire de la population initiale qui produit peut-être des individus qui se trouvent tous très loin de la solution optimale. Nous proposerons l'implémentation d'un taux de mutation variable afin d'évaluer si l'algorithme privilégiera un taux de mutation plus raisonnable au fil des générations.
- Analyser pourquoi les dépendances ontologiques très lointaines contribuent à l'amélioration du modèle de langage. Est-ce qu'elles sont liées au contexte, au corpus ou est-ce dû à une meilleure (re)distribution des *N*-grammes de faibles comptes.
- Le problème d'optimisation des poids d'interpolation des modèles de langage comportait un espace de recherche assez restreint et les algorithmes évolutionnaires n'ont pas été en mesure, en moyenne, de proposer de meilleures solutions finales que celles obtenues à l'aide de l'optimisation par l'algorithme de Powell. Nous tenterons donc d'évaluer ces derniers dans un contexte plus complexe, par exemple pour l'optimisation des poids des bacs pour le lissage Kneser-Ney et un lissage plus complexe basé sur les ontologies.

RÉFÉRENCES

[1] K. Davis, R. Biddulph, et S. Balashek, « Automatic recognition of spoken digits », *J. Acoust. Soc. Am.*, vol. 24, nº 6, p. 637-642, 1952.

[2] « iOS 8 - Siri - Apple (CA) », *Apple*. [En ligne]. Disponible à: http://www.apple.com/ca/ios/siri/. [Consulté le: 03-févr-2016].

[3] L. de Vilmorin et J. Hugo, *L'alphabet des aveux*. NRF, 1954.

[4] G. Pobric, E. Jefferies, et M. A. L. Ralph, « Anterior temporal lobes mediate semantic representation: Mimicking semantic dementia by using rTMS in normal participants », *Proc. Natl. Acad. Sci.*, vol. 104, nº 50, p. 20137–20141, déc. 2007.

[5] K. A. Lindquist, A. B. Satpute, et M. Gendron, « Does Language Do More Than Communicate Emotion? », *Curr. Dir. Psychol. Sci.*, vol. 24, nº 2, p. 99-108, 2015.

[6] A. B. Satpute, J. Shu, J. Weber, M. Roy, et K. N. Ochsner, « The functional neural architecture of self-reports of affective experience », *Biol. Psychiatry*, vol. 73, nº 7, p. 631-638, 2013.

[7] « Cortana - Meet your personal assistant - Microsoft - Global ». [En ligne]. Disponible à: https://www.microsoft.com/en/mobile/experiences/cortana/. [Consulté le: 03-févr-2016].

[8] « Viv ». [En ligne]. Disponible à: http://viv.ai/. [Consulté le: 26-août-2016].

[9] Y. Benahmed, « Dialogue Humain-Machine sur les Dispositifs Mobiles », mastersthesis, Université de Moncton, Moncton, NB, Canada, 2009.

[10] L. R. Karl, M. Pettey, et B. Shneiderman, « Speech Versus Mouse Commands for Word Processing: An Empirical Evaluation », *Int. J. Man-Mach. Stud.*, vol. 39, nº 4, p. 667-687, 1993.

[11] G. Argoud et others, *Les pneumatiques d'Héron d'Alexandrie*, vol. 15. Université de Saint-Etienne, 1997.

[12] B. Woodcroft, « The pneumatics of hero of Alexandria », *Pneum. Hero Alex. Publ. TAYLOR Walt. MABERLY Lond. 1851*, 1851.

[13] « Zenbo - ASUS ». [En ligne]. Disponible à: http://zenbo.asus.com/. [Consulté le: 07-sept-2016].

[14] « Life with Pepper（ペッパーと暮らす） | 特集 | ロボット », ソフトバンク. [En ligne]. Disponible à: http://www.softbank.jp/robot/special/pepper/. [Consulté le: 07-sept-2016].

[15] « Anybots 2.0 Inc. - NEW QB2.0 WebRTC ». [En ligne]. Disponible à: https://www.anybots.com. [Consulté le: 07-sept-2016].

[16] « OpenBCI - Open Source Biosensing Tools (EEG, EMG, EKG, and more) ». [En ligne]. Disponible à: http://openbci.com/. [Consulté le: 07-sept-2016].

[17] « Deep Dream - Online Generator ». [En ligne]. Disponible à: http://deepdreamgenerator.com/. [Consulté le: 07-sept-2016].

[18] T. Mikolov, « Statistical Language Models Based on Neural Networks », phdthesis, Ph. D. thesis, Brno University of Technology, 2012.

[19] Wikipédia, « Pragmatique (linguistique) --- Wikipédia, l'encyclopédie libre », 2016.

[20] J. K. Baker, « Stochastic Modeling as a Means of Automatic Speech Recognition. », Carnegie Mellon University, 1975.

[21] « Ontologie », dans *Dictionnaire de la Philosophie: (Les Dictionnaires d'Universalis)*, Encyclopaedia Universalis.

[22] « Ontologie (informatique) », *Wikipédia*. 14-août-2016.

[23] É. Larousse, « Définitions : paraphrase - Dictionnaire de français Larousse ». [En ligne]. Disponible à: http://www.larousse.fr/dictionnaires/francais/paraphrase/57993. [Consulté le: 07-sept-2016].

[24] I. H. Witten et T. C. Bell, « The Zero-Frequency Problem: Estimating the Probabilities of Novel Events in Adaptive Text Compression », *Inf. Theory IEEE Trans. On*, vol. 37, nº 4, p. 1085–1094, 1991.

[25] « Ford SYNC | SYNC Support, Phone Compatibility, Updates, Manuals, Voice Guides, Instructions & More | Ford.ca ». [En ligne]. Disponible à: http://www.ford.ca/technology/sync/. [Consulté le: 04-févr-2016].

[26] P. Heisterkamp, « Linguatronic Product-Level Speech System for Mercedes-Benz Cars », dans *Proceedings of the first international conference on Human language technology research*, 2001, p. 1-2.

[27] P. F. Brown, V. J. D. Pietra, S. A. D. Pietra, et R. L. Mercer, « The mathematics of statistical machine translation: Parameter estimation », *Comput. Linguist.*, vol. 19, nº 2, p. 263-311, 1993.

[28] A. De Saint-Exupéry, *Le Petit Prince*. Gallimard, 1943.

[29] D. Jurafsky et J. H. Martin, *Speech and Language Processing. An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall, 2008.

[30] « Wikipedia ». [En ligne]. Disponible à: https://www.wikipedia.org/. [Consulté le: 12-sept-2016].

[31] T. Zesch, C. Müller, et I. Gurevych, « Extracting Lexical Semantic Knowledge from Wikipedia and Wiktionary », dans *Proceedings of the Conference on Language Resources and Evaluation (LREC)*, 2008.

[32] R. Collobert et J. Weston, « A unified architecture for natural language processing: Deep neural networks with multitask learning », dans *Proceedings of the 25th international conference on Machine learning*, 2008, p. 160-167.

[33] L. F. Lamel, J.-L. Gauvain, et M. Eskénazi, « BREF, a Large Vocabulary Spoken Corpus for French », dans *Second European Conference on Speech Communication and Technology*, Genova, Italie, 1991, p. 505-508.

[34] C. Chelba *et al.*, « One billion word benchmark for measuring progress in statistical language modeling », *ArXiv Prepr. ArXiv13123005*, 2013.

[35] I. J. Good, « The Population Frequencies of Species and the Estimation of Population Parameters », *Biometrika*, vol. 40, nº 3-4, p. 237-264, 1953.

[36] S. F. Chen et J. Goodman, « An Empirical Study of Smoothing Techniques for Language Modeling », dans *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, 1996, p. 310–318.

[37] S. Katz, « Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer », *Acoust. Speech Signal Process. IEEE Trans. On*, vol. 35, nº 3, p. 400–401, 1987.

[38] T. Brants, A. C. Popat, P. Xu, F. J. Och, et J. Dean, « Large Language Models in Machine Translation », dans *In Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 2007.

[39] F. Jelinek et R. L. Mercer, « Interpolated estimation of Markov source parameters from sparse data », dans *In Proceedings of the Workshop on Pattern Recognition in Practice*, Amsterdam, The Netherlands: North-Holland, 1980, p. 381-397.

[40] P. F. Brown, P. V. deSouza, R. L. Mercer, V. J. D. Pietra, et J. C. Lai, « Class-based N-gram Models of Natural Language », *Comput. Linguist.*, vol. 18, nº 4, p. 467–479, déc. 1992.

[41] H. Ney, U. Essen, et R. Kneser, « On Structuring Probabilistic Dependences in Stochastic Language Modelling », *Comput. Speech Lang.*, vol. 8, nº 1, p. 1–38, janv. 1994.

[42] K. W. Church et W. A. Gale, « A Comparison of the Enhanced Good-Turing and Deleted Estimation Methods for Estimating Probabilities of English Bigrams », *Comput. Speech Lang.*, vol. 5, nº 1, p. 19-54, janv. 1991.

[43] R. Kneser et H. Ney, « Improved Backing-Off for M-Gram Language Modeling », dans *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, 1995, vol. 1, p. 181-184 vol.1.

[44] J. Goodman et S. Chen, « An empirical study of smoothing techniques for language modeling », Technical Report TR-10-98, Harvard University, August, 1998.

[45] J. Uszkoreit et T. Brants, « Distributed Word Clustering for Large Scale Class-Based Language Modeling in Machine Translation. », dans *ACL*, 2008, p. 755-762.

[46] S. Watanabe et J. T. Chien, *Bayesian Speech and Language Processing*. Cambridge University Press, 2015.

[47] Y. W. Teh, « A Hierarchical Bayesian Language Model Based on Pitman-Yor Processes », dans *In Coling/ACL, 2006. 9*, 2006.

[48] S. Huang et S. Renals, « Hierarchical Pitman-Yor language models for ASR in meetings », dans *Automatic Speech Recognition & Understanding, 2007. ASRU. IEEE Workshop on*, 2007, p. 124-129.

[49] F. Wood et Y. W. Teh, « A Hierarchical Nonparametric Bayesian Approach to Statistical Language Model Domain Adaptation », dans *International Conference on Artificial Intelligence and Statistics*, 2009, vol. 12, p. 607.

[50] J. Pitman, *Combinatorial Stochastic Processes: Ecole D'Eté de Probabilités de Saint-Flour XXXII-2002*. Springer, 2006.

[51] L. Brillouin, *La science et la théorie de l'information*. Masson, 1959.

[52] D. M. Powers, « Applications and explanations of Zipf's law », dans *Proceedings of the joint conferences on new methods in language processing and computational natural language learning*, 1998, p. 151-160.

[53] D. M. Blei, A. Y. Ng, et M. I. Jordan, « Latent Dirichlet Allocation », *J Mach Learn Res*, vol. 3, p. 993–1022, mars 2003.

[54] J.-T. Chien et C.-H. Chueh, « Dirichlet class language models for speech recognition », *IEEE Trans. Audio Speech Lang. Process.*, vol. 19, nº 3, p. 482-495, 2011.

[55] Y. Bengio, R. Ducharme, P. Vincent, et C. Jauvin, « A Neural Probabilistic Language Model », *J. Mach. Learn. Res.*, vol. 3, p. 1137–1155, 2003.

[56] T. Mikolov, J. Kopecky, L. Burget, O. Glembek, et others, « Neural Network Based Language Models for Highly Inflective Languages », dans *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2009, p. 4725-4728.

[57] T. Mikolov, M. Karafiát, L. Burget, J. Cernocky, et S. Khudanpur, « Recurrent Neural Network Based Language Model. », dans *Interspeech*, 2010, vol. 2, p. 3.

[58] E. Arisoy, T. N. Sainath, B. Kingsbury, et B. Ramabhadran, « Deep Neural Network Language Models », dans *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, Stroudsburg, PA, USA, 2012, p. 20–28.

[59] A. Mnih et Y. W. Teh, « A Fast and Simple Algorithm for Training Neural Probabilistic Language Models », *ArXiv12066426 Cs*, juin 2012.

[60] E. Arisoy, S. F. Chen, B. Ramabhadran, et A. Sethy, « Converting Neural Network Language Models into Back-off Language Models for Efficient Decoding in Automatic Speech Recognition », *IEEEACM Trans Audio Speech Lang Proc*, vol. 22, nº 1, p. 184–192, janv. 2014.

[61] P. Vincent, A. de Brébisson, et X. Bouthillier, « Efficient Exact Gradient Update for Training Deep Networks with Very Large Sparse Targets », dans *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, et R. Garnett, Éd. Curran Associates, Inc., 2015, p. 1108-1116.

[62] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, et Y. Wu, « Exploring the Limits of Language Modeling », *ArXiv160202410 Cs*, févr. 2016.

[63] R. Bellman et R. Corporation, *Dynamic Programming*. Princeton University Press, 1957.

[64] R. Bellman et R. E. Bellman, *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1961.

[65] S. Hochreiter et J. Schmidhuber, « Long short-term memory », *Neural Comput.*, vol. 9, nº 8, p. 1735-1780, 1997.

[66] F. Gers, « Long short-term memory in recurrent neural networks », phdthesis, Universität Hannover, 2001.

[67] M. Sundermeyer, R. Schlüter, et H. Ney, « LSTM Neural Networks for Language Modeling. », dans *Interspeech*, 2012, p. 194-197.

[68] S. Merity, N. S. Keskar, et R. Socher, « Regularizing and Optimizing LSTM Language Models », *ArXiv170802182 Cs*, août 2017.

[69] N. Shazeer *et al.*, « Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer », *ArXiv170106538 Cs Stat*, janv. 2017.

[70] A. Krizhevsky, I. Sutskever, et G. E. Hinton, « ImageNet Classification with Deep Convolutional Neural Networks », dans *Advances in neural information processing systems*, 2012, p. 1097-1105.

[71] G. F. Montúfar et J. Morton, « When does a mixture of products contain a product of mixtures? », *SIAM J. Discrete Math.*, vol. 29, nº 1, p. 321-347, 2015.

[72] G. F. Montúfar, R. Pascanu, K. Cho, et Y. Bengio, « On the number of linear regions of deep neural networks », dans *Advances in neural information processing systems*, 2014, p. 2924-2932.

[73] B. Zoph et Q. V. Le, « Neural Architecture Search with Reinforcement Learning », *ArXiv161101578 Cs*, nov. 2016.

[74] K. Beneš, M. K. Baskar, et L. Burget, « Residual Memory Networks in Language Modeling: Improving the Reputation of Feed-Forward Networks », 2017, p. 284-288.

[75] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, et D. Meger, « Deep Reinforcement Learning that Matters », *ArXiv170906560 Cs Stat*, sept. 2017.

[76] R. Islam, P. Henderson, M. Gomrokchi, et D. Precup, « Reproducibility of Benchmarked Deep Reinforcement Learning Tasks for Continuous Control », *ArXiv170804133 Cs*, août 2017.

[77] N. Reimers et I. Gurevych, « Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging », dans *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Copenhagen, Denmark, 2017, p. 338–348.

[78] J. Goodman, « A Bit of Progress in Language Modeling », Technique MSR-TR-2001-72, 2001.

[79] A. A. Ye-Yi Wang, « Is Word Error Rate a Good Indicator for Spoken Language Understanding Accuracy », dans *IEEE Workshop on Automatic Speech Recognition and Understanding*, St. Thomas, US Virgin Islands, 2003.

[80] Y. Benahmed, S.-A. Selouani, D. O'Shaughnessy, et A. H. Abolhassani, « Real-Life Speech-Enabled System to Enhance Interaction with RFID Networks in Noisy Environments », dans *International Conference on Acoustics, Speech and Signal Processing*, Prague, Czech Republic, 2011, p. 1781-1784.

[81] A. H. Abolhassani, S.-A. Selouani, et D. O'Shaughnessy, « Speech Enhancement Using PCA and Variance of the Reconstruction Error in Distributed Speech Recognition », dans *IEEE ASRU Workshop*, Kyoto, 2007, p. 19-23.

[82] A. H. Abolhassani, J. Benesty, et P. Kabal, « Speech Enhancement Using Principal Component Analysis and Variance of the Reconstruction Error Model Identification », *Univ. Qué. Masters Degree Telecommun.*, 2008.

[83] Y. Benahmed, S.-A. Selouani, et D. O'Shaughnessy, « Ontology-based pattern generator and root semantic analyser for spoken dialogue systems », dans *Electrical & Computer Engineering (CCECE), 2012 25th IEEE Canadian Conference on*, 2012, p. 1-4.

[84] D. Estival, C. Nowak, et A. Zschorn, « Towards Ontology-based Natural Language Processing », dans *In RDF/RDFS and OWL in Language Technology: 4th Workshop on NLP and XML. ACL*, 2004, p. 59-66.

[85] D. Kauchak et R. Barzilay, « Paraphrasing for automatic evaluation », dans *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, Stroudsburg, PA, USA, 2006, p. 455-462.

[86] Y. Benahmed, S.-A. Selouani, et D. O'Shaughnessy, « A Bin-Based Ontological Framework for Low-Resource N-Gram Smoothing in Language Modelling », dans *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, Florence, Italy, 2014, p. 4918-4922.

[87] Y. Benahmed, S.-A. Selouani, et D. O'Shaughnessy, « Evaluation of Graph Metrics for Optimizing Bin-Based Ontologically Smoothed Language Models », dans *European Signal Processing Conference*, Budapest, Hongrie, 2016.

[88] G. A. Miller, « WordNet: A Lexical Database for English », *Commun. ACM*, vol. 38, p. 39–41, 1995.

[89] « jardon-u/wordnet-blast », *GitHub*. [En ligne]. Disponible à: https://github.com/jardon-u/wordnet-blast. [Consulté le: 04-oct-2016].

[90] T. Pedersen, S. Patwardhan, et J. Michelizzi, « WordNet::Similarity: Measuring the Relatedness of Concepts », dans *Demonstration Papers at HLT-NAACL 2004*, Boston, Massachusetts, 2004, p. 38-41.

[91] J. H. Lee, M. H. Kim, et Y. J. Lee, « Information Retrieval Based on Conceptual Distance in Is-a Hierarchies », *J. Doc.*, vol. 49, nº 2, p. 188–207, 1993.

[92] D. B. Paul et J. M. Baker, « The design for the Wall Street Journal-based CSR corpus », dans *Proceedings of the workshop on Speech and Natural Language*, 1992, p. 357-362.

[93] A. Stolcke, « SRILM - An Extensible Language Modeling Toolkit », 2002, p. 901-904.

[94] A. N. Langville et C. D. Meyer, « Deeper Inside PageRank », *Internet Math.*, vol. 1, nº 3, p. 335-380, janv. 2004.

[95] C. Ding, X. He, P. Husbands, H. Zha, et H. D. Simon, « PageRank, HITS and a Unified Framework for Link Analysis », dans *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Tampere, Finland, 2002, p. 353-354.

[96] J. M. Kleinberg, « Authoritative Sources in a Hyperlinked Environment », *J. ACM JACM*, vol. 46, nº 5, p. 604–632, 1999.

[97] S. Brin et L. Page, « The Anatomy of a Large-Scale Hypertextual Web Search Engine, 1998 », dans *Proceedings of the Seventh World Wide Web Conference*, 2007.

[98] M. E. Newman, « Analysis of Weighted Networks », *Phys. Rev. E*, vol. 70, nº 5, p. 056131, 2004.

[99] A.A.Puntambekar, *Theory Of Automata And Formal Languages*. Technical Publications, 2007.

[100] J. Atkinson et J. Matamala, « Evolutionary Shallow Natural Language Parsing », *Comput. Intell.*, vol. 28, nº 2, p. 156–175, 2012.

[101] M. J. Powell, « An Efficient Method for Finding the Minimum of a Function of Several Variables Without Calculating Derivatives », *Comput. J.*, vol. 7, nº 2, p. 155–162, 1964.

[102] R. A. Formato, « Central force optimization: a new metaheuristic with applications in applied electromagnetics », *Prog. Electromagn. Res.*, vol. 77, p. 425-491, 2007.

[103] R. A. Formato, « Central force optimization: a new nature inspired computational framework for multidimensional search and optimization », dans *Nature Inspired Cooperative Strategies for Optimization (NICSO 2007)*, Springer, 2008, p. 221-238.

[104] E. Rashedi, H. Nezamabadi-Pour, et S. Saryazdi, « GSA: a gravitational search algorithm », *Inf. Sci.*, vol. 179, nº 13, p. 2232-2248, 2009.

[105] M. Shams, E. Rashedi, et A. Hakimi, « Clustered-gravitational search algorithm and its application in parameter optimization of a low noise amplifier », *Appl. Math. Comput.*, vol. 258, p. 436-453, 2015.

[106] B.-J. P. Hsu et J. R. Glass, « Iterative language model estimation: efficient data structure & algorithms. », dans *INTERSPEECH*, 2008, p. 841-844.

[107] W. Briggs, *Uncertainty: The Soul of Modeling, Probability & Statistics*. Springer International Publishing, 2016.

[108] D. Simon, *Evolutionary Optimization Algorithms*. Wiley, 2013.

[109] « Genetic Algorithm Options - MATLAB & Simulink ». [En ligne]. Disponible à: https://www.mathworks.com/help/gads/genetic-algorithm-options.html. [Consulté le: 26-sept-2016].

[110] F. E. Fish, « Kinematics of ducklings swimming in formation: Consequences of position », *J. Exp. Zool.*, vol. 273, nº 1, p. 1-11, sept. 1995.

[111] S. R. Noren, G. Biedenbach, J. V. Redfern, et E. F. Edwards, « Hitching a ride: the formation locomotion strategy of dolphin calves », *Funct. Ecol.*, vol. 22, nº 2, p. 278-283, avr. 2008.

[112] J. L. Russell, « Kepler's laws of planetary motion: 1609--1666 », *Br. J. Hist. Sci.*, vol. 2, nº 1, p. 1-24, 1964.

[113] Y. Shi et R. Eberhart, « A modified particle swarm optimizer », dans *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)*, 1998, p. 69-73.

[114] A. Shukla, R. Tiwari, et R. Kala, *Real life applications of soft computing*. CRC Press, 2010.

[115] « Features - Global Optimization Toolbox ». [En ligne]. Disponible à: https://www.mathworks.com/products/global-optimization/features.html#genetic-algorithm-solver. [Consulté le: 14-nov-2016].

[116] S. S. Chen, « Constrained Particle Swarm Optimization ». [En ligne]. Disponible à: http://www.mathworks.com/matlabcentral/fileexchange/25986-constrained-particle-swarm-optimization. [Consulté le: 10-oct-2016].

[117] M. E. H. Pedersen, « Good parameters for particle swarm optimization », Rapport technique HL1001, 2010.

[118] E. Jones, T. Oliphant, P. Peterson, et others, *SciPy: Open source scientific tools for Python*. 2001.

[119] S. J. Young, « The HTK Hidden Markov Model Toolkit: Design and Philosophy », *Entropic Camb. Res. Lab. Ltd*, vol. 2, p. 2–44, 1994.

[120] K. Vertanen, « Baseline WSJ Acoustic Models for HTK and Sphinx: Training Recipes and Recognition Experiments », Cavendish Laboratory, University of Cambridge, 2006.

[121] « The CMU Pronouncing Dictionary ». [En ligne]. Disponible à: http://www.speech.cs.cmu.edu/cgi-bin/cmudict#about. [Consulté le: 22-juin-2017].

[122] S. Young *et al.*, « The HTK book », *Camb. Univ. Eng. Dep.*, vol. 3, p. 175, 2002.

[123] *Speech Recognition Scoring Toolkit (SCTK)*. NIST.

[124] R. Brown, « Exploring New Speech Recognition and Synthesis APIs In Windows Vista », *MSDN Mag.*, 2006.

[125] T. Paek et D. Chickering, « Improving Command and Control Speech Recognition on Mobile Devices: Using Predictive User Models for Language Modeling », *User Model. User-Adapt. Interact.*, vol. 17, nº 1, p. 93-117, 2007.

[126] S. A. Shah, A. U. Asar, et S. W. Shah, « Interactive Voice Response with Pattern Recognition Based on Artificial Neural Network Approach », dans *IEEE International conference on Emerging Technologies*, 2007, p. 249–252.

[127] G. O. Sing, K. W. Wong, C. C. Fung, et A. Depickere, « Towards a More Natural and Intelligent Interface with Embodied Conversation Agent », dans *CyberGames '06: Proceedings of the 2006 international conference on Game research and development*, Perth, Australia, 2006, p. 177-183.

[128]N. Bush, R. Wallace, T. Ringate, A. Taylor, et J. Baer, « Artificial Intelligence Markup Language (AIML) Version 1.0. 1 », *ALICE AI Found. Work. Draft*, 2001.

[129]R. Vetter, N. Virag, P. Renevey, et J.-M. Vesin, « Single Channel Speech Enhancement Using Principal Component Analysis and Mdl Subspace Selection », dans *Proceedings of the 6th European Conference on Speech Communication and Technology (EUROSPEECH'99)*, 1999, vol. 5, p. 2411-2414.

[130]Y. Hu et P. Loizou, « Evaluation of Objective Quality Measures for Speech Enhancement », dans *IEEE Transactions on Speech and Audio Processing*, 2008, vol. 16 1, p. 229–238.

[131]UIT, « P.862.1 : Mapping function for transforming P.862 raw result scores to MOS-LQO ». [En ligne]. Disponible à: https://www.itu.int/rec/T-REC-P.862.1-200311-I/en. [Consulté le: 24-mai-2018].

[132]A. Varga et H. J. M. Steeneken, « Assessment for Automatic Speech Recognition: II. NOISEX-92: A Database and an Experiment to Study the Effect of Additive Noise on Speech Recognition Systems », *Speech Commun.*, vol. 12, nº 3, p. 247–251, 1993.

[133]J. M. W. M. Liu, K.A. Jellyman et N. Evans, « Assessment of Objective Quality Measures for Speech Intelligibility Estimation », dans *ICASSP 2006*, 2006, vol. 1.

[134]S.-A. Selouani, T.-H. Lê, Y. Benahmed, et D. O'Shaughnessy, « Speech-Enabled Tools for Augmented Interaction in E-Learning Applications », *Int. J. Distance Educ. Technol.*, vol. 6, nº 2, p. 1-20, 2008.

[135]W. O. W. Group, *OWL 2 Web Ontology Language: Document Overview*. W3C Recommendation, 2009.

[136]A. Pease, « Standard Upper Ontology Knowledge Interchange Format », *Unpubl. Lang. Man. Available Httpsigmakee Sourceforge Net*, 2004.

[137]H. Schmid, « Probabilistic Part-of-Speech Tagging Using Decision Trees », dans *Proceedings of the International Conference on New Methods in Language Processing*, Manchester, UK, 1994, p. 44-49.

[138]S. Huang et S. Renals, « Hierarchical Bayesian Language Models for Conversational Speech Recognition », *Audio Speech Lang. Process. IEEE Trans. On*, vol. 18, nº 8, p. 1941-1954, 2010.

[139]W. Naptali, M. Tsuchiya, et S. Nakagawa, « Topic-Dependent-Class-Based -Gram Language Model », *Audio Speech Lang. Process. IEEE Trans. On*, vol. 20, nº 5, p. 1513-1525, juill. 2012.

[140]X. Zhu et R. Rosenfeld, « Improving Trigram Language Modeling with the World Wide Web », dans *Acoustics, Speech, and Signal Processing, 2001. Proceedings. (ICASSP '01). 2001 IEEE International Conference on*, 2001, vol. 1, p. 533-536 vol.1.

[141]I. Bulyko, M. Ostendorf, et A. Stolcke, *Getting More Mileage from Web Text Sources for Conversational Speech Language Modeling Using Class-Dependent Mixtures*. Association for Computational Linguistics, 2003.

[142]A. Sethy, P. G. Georgiou, et S. Narayanan, « Building Topic Specific Language Models from Webdata Using Competitive Models. », dans *INTERSPEECH*, 2005, p. 1293-1296.

[143]D. Lin *et al.*, « New Tools for Web-Scale N-Grams », dans *Proceedings of LREC*, 2010.

[144]M. Bastian, S. Heymann, et M. Jacomy, « Gephi: An Open Source Software for Exploring and Manipulating Networks. », *ICWSM*, vol. 8, p. 361-362, 2009.

[145]L. Jing, L. Zhou, M. K. Ng, et J. Z. Huang, « Ontology-Based Distance Measure for Text Clustering », dans *Proceedings of the Text Mining Workshop, SIAM International Conference on Data Mining*, 2006.

[146]W. A. Gale et G. Sampson, « Good-Turing Frequency Estimation Without Tears », *J. Quant. Linguist.*, vol. 2, nº 3, p. 217-237, 1995.

[147]C. Biemann, S. Roos, et K. Weihe, « Quantifying Semantics Using Complex Network Analysis », dans *In COLING'12*, 2012.

[148]R. Yan, H. Jiang, M. Lapata, S.-D. Lin, X. Lv, et X. Li, « Semantic v.s. Positions: Utilizing Balanced Proximity in Language Model Smoothing for Information Retrieval », dans *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, Nagoya, Japan, 2013, p. 507–515.

[149]V. D. Blondel, J.-L. Guillaume, R. Lambiotte, et E. Lefebvre, « Fast Unfolding of Communities in Large Networks », *J. Stat. Mech. Theory Exp.*, vol. 2008, nº 10, p. P10008, 2008.