Décodage de graphe à l'aide de colonies de fourmis

Benjamin Lecouteux, Didier Schwab

Reconnaissance automatique de la parole à l'aide de colonies de fourmis

Benjamin Lecouteux Didier Schwab
Groupe d'Étude en Traduction Automatique/Traitement Automatisé des Langues et de la Parole
Laboratoire d'Informatique de Grenoble
Univ. Grenoble Alpes
benjamin.lecouteux@imag.fr, didier.schwab@imag.fr

RÉSUMÉ

Cet article présente une approche originale permettant de décoder des graphes issus d'un Système de Reconnaissance Automatique de la Parole (SRAP) à l'aide d'un algorithme constructif : les colonies de fourmis. L'application d'un modèle de langage d'ordre supérieur à un graphe nécessite son extension afin de construire des historiques correspondants à chaque nouvel n-gramme observé. Cette extension peut rapidement engendrer des calculs lourds et une consommation de mémoire conséquente. Nous proposons une alternative où des colonies de fourmis explorent le graphe avec un nouveau modèle de langage sans la nécessité d'effectuer une extension. Nous présentons des premiers résultats encourageants se basant sur le corpus anglais TED où des bi-grammes sont réévalués en quadri-grammes. Finalement, nous discutons sur les atouts de cette approche qui permet d'envisager des décodages massivement parallélisables pour un même graphe ainsi que des contrôles stricts du temps de calcul et de la mémoire.

ABSTRACT

Automatic speech recognition using ant colony algorithm

In this paper we propose an original approach allowing to decode Automatic Speech Recognition Graphs by using a constructive algorithm based on ant colonies. When a graph is decoded with a language model using a higher order ; classical algorithm must to extend graph in order to develop each new observed n-gram. This extension increase time and memory consumption. We propose to use ant colony algorithm in order to explore ASR graphs with a new language model, without the necessity to extend it. We present first results based on TED english corpora where graphs are decoded with 4-grams. Finally, we show that our approach allows one to provide higly threaded decoding and a control of time and memory consumption.

MOTS-CLÉS : Décodage de graphe, colonies de fourmis, modèle de langage.

KEYWORDS: Graph decoding, ant colony algorithm, language model.

1 Introduction

Dans cet article nous présentons un travail préliminaire introduisant un nouveau paradigme permettant d'explorer les graphes issus d'un système de reconnaissance automatique de la parole (SRAP). Un problème rencontré lors de l'exploration des graphes de mots est l'application sur ces derniers d'un modèle de langage d'ordre supérieur. En effet, le nombre de chemins croît avec l'ordre du modèle de langage. Ainsi un graphe de mots initialement décodé avec un modèle bi-gramme qui est étendu avec un modèle quadrigramme peut voire sa taille multipliée par plus de 10. Plusieurs techniques ont été appliquées avec succès pour approcher (*compact-expansion*) ou effectuer des coupures dans la recherche (*beam-search*). Nous proposons une méthode alternative se basant sur un algorithme constructif très utilisé dans le domaine de la recherche opérationnelle : les colonies de fourmis. Cet algorithme a notamment été appliqué à la résolution du voyageur de commerce ou encore à la désambiguisation lexicale.

L'article s'organise de la manière suivante : dans la section 2 nous faisons un rapide tour d'horizon des techniques de décodage pour les SRAP grand vocabulaire ainsi que de l'algorithme des colonies de fourmis. La section 3 présente l'ensemble du système qui a été utilisé pour effectuer nos travaux. Ensuite nous abordons l'ensemble des expériences menées dans la section 5 avec leur analyse, suivies par une discussion dans la section 6. Enfin nous concluons en proposant quelques perspectives intéressantes liées à l'utilisation de cet algorithme.

2 État de l'art

2.1 Approches classiques

Le principe d'un SRAP est de trouver dans un graphe l'hypothèse qui maximisera les probabilités des modèles linguistiques et acoustiques. Généralement des heuristiques sont appliquées afin de limiter l'espace de recherche. On trouve ainsi dans la littérature plusieurs approches d'exploration dynamique pour générer les hypothèses :

— Les algorithmes de graphes synchrones qui génèrent une copie virtuelle du graphe pour l'ensemble des hypothèses se finissant à un temps t. Le mot suivant est partagé, mais associé à plusieurs historiques. Dans cette approche l'algorithme dynamique d'alignement est appliqué pour tout nouveau mot sans avoir à conserver d'historique. La mise en oeuvre de cet algorithme s'avère rare (Ortmanns et Ney, 2000).

— Les algorithmes de graphes réentrants où une copie virtuelle du graphe est explorée pour chaque contexte linguistique. L'information relative à chaque contexte est enregistrée pour chaque chemin et combinée avec la nouvelle racine virtuelle dépendant de l'historique relatif au modèle de langage (Ney *et al.*, 1992). Ces algorithmes sont très répandus et leur implémentation la plus courante est un Viterbi en faisceau (*beam-search*). Leur conception se rapproche de la programmation dynamique.

— Les algorithmes asynchrones à pile (Jelinek, 1969) dont le principe est d'explorer en profondeur et en priorité les hypothèses qui semblent prometteuses. Ceci se fait en étendant mot par mot l'hypothèse sélectionnée. Les réalisation de ce type d'algorithme se base sur des piles ordonnant les hypothèses à explorer. La mise en oeuvre la plus courante est l'implémentation de l'algorithme A^* (Paul, 1991; Nocera *et al.*, 2002).

Les documents (Aubert, 2002; Lecouteux, 2008) présentent en détail les principales techniques de décodage des systèmes de reconnaissance de la parole grand vocabulaire.

2.2 Algorithmes à colonies de fourmis

Les algorithmes à base de fourmis font partie des approches constructives. Dans ces approches, de nouvelles configurations sont générées par ajout itératif d'éléments de solution aux configurations en cours de construction. Ils ont pour origine la biologie et les observations réalisées sur le comportement social des fourmis. En effet, ces insectes ont collectivement la capacité de trouver le plus court chemin entre leur fourmilière et une source d'énergie. Il a pu être démontré que la coopération au sein de la colonie est auto-organisée et résulte d'interactions entre individus autonomes. Ces interactions, souvent très simples, permettent à la colonie de résoudre des problèmes complexes. Ce phénomène est appelé intelligence en essaim (Bonabeau et Théraulaz, 2000).

En 1989, Jean-Louis Deneubourg étudie le comportement des fourmis biologiques dans le but de comprendre la méthode avec laquelle elles choisissent le plus court chemin et le retrouvent en cas d'obstacle. Il élabore ainsi le modèle stochastique dit de Deneubourg (Deneubourg *et al.*, 1989), conforme à ce qui est observé statistiquement sur les fourmis réelles quant à leur partage entre les chemins. Ce modèle stochastique est à l'origine des travaux sur les algorithmes à fourmis.

Le concept principal de l'intelligence en essaim est la stygmergie, c'est-à-dire l'interaction entre agents par modification de l'environnement. Une des premières méthodes que l'on peut apparenter aux algorithmes à fourmis est l'écorésolution qui a montré la puissance d'une heuristique de résolution collective basée sur la perception locale, évitant tout parcours explicite de graphe d'états (Drogoul, 1993).

En 1992, Marco Dorigo et Luca Maria Gambardella conçoivent le premier algorithme basé sur ce paradigme pour le célèbre problème combinatoire du voyageur de commerce (Dorigo et Gambardella, 1997). Dans les algorithmes à base de fourmis artificielles, l'environnement est généralement représenté par un graphe et les fourmis virtuelles utilisent l'information accumulée sous la forme de chemins de phéromone déposée sur les arcs du graphe. De façon simple, une fourmi se contente de suivre les traces de phéromones déposées précédemment ou explore au hasard dans le but de trouver un chemin optimal, fonction du problème posé, dans le graphe. Le lecteur trouvera dans (Dorigo et Stützle, 2004) ou (Monmarche *et al.*, 2009) de bons états de l'art sur la question.

Ce type d'algorithme a déjà été appliqué à des problèmes en traitement automatique du langage, notamment dans le cadre de la désambiguisation lexicale (Schwab *et al.*, 2013). Nos travaux visent à étendre ce paradigme à la reconnaissance automatique de la parole.

3 Le système SRAP

Le système SRAP utilisé est basé sur le *toolkit* KALDI (Povey *et al.*, 2011). KALDI propose la mise à disposition des outils état de l'art pour la RAP. Les modèles acoustiques et les phonétisations de notre système ont été appris avec les données mises à disposition par le LIUM (Rousseau *et al.*, 2012) qui représentent 118 heures de données annotées. L'enrichissement des paramètres par les coefficients delta et delta-delta, ainsi que plusieurs transformations telles que LDA et MLLT

Données	TEM 2-gr	TEM 3-gr	TEM 4-gr
Dev 2010 (4h12)	22.01 %	18.86%	17.6%
Test 2010 (7h30)	21.67%	17.98%	17.0%

Ordre ML	Nb n-grammes	Perplexité
1g	41K	-
2g	+ 35M	234
3g	+ 238M	159
4g	+ 524M	150

TABLE 1 – Perplexité des différents modèles de langage utilisés par notre système, ainsi que les taux d'erreur mots (TEM) obtenus par notre système de référence.

(Gopinath, 1998) sont appliquées. Enfin, un apprentissage adapté au locuteur (*Speaker Adaptive Training* - SAT) est réalisé et combiné avec une adaptation dans l'espace des paramètres de type fMLLR (Gales, 1998).

Le modèle de langage a quant à lui été appris sur l'ensemble des données fournies pour la campagne IWSLT 2012 ainsi que sur la partie d'apprentissage fournie par le LIUM. Un modèle a été appris pour chaque sous-corpus et l'ensemble des modèles a été interpolé en minimisant la perplexité sur les données de développement, sans appliquer de sélections.

Au final la perplexité obtenue sur le corpus *dev* est de 159 pour le modèle trigramme et 150 pour le quadrigramme. Les quantités de n-grammes et la perplexité des différents modèles utilisés pour nos expériences sont présentées dans le tableau 1.

L'ensemble des expériences ont été effectuées sur un corpus mis à disposition par la communauté pour la campagne d'évaluation IWSLT 2010 : le corpus TED qui correspond à un ensemble de conférences enregistrées en anglais (Paul *et al.*, 2010). Les résultats du système de référence sont également présentés dans le tableau 1.

3.1 Conversion des graphes

Il est important de noter que KALDI est conçu autour d'une modélisation basée sur des automates à états finis. Dans un premier temps, nos expériences sont menées sur des graphes plus classiques utilisant le formalisme HTK. Nous avons donc converti l'ensemble des graphes issus de KALDI en supprimant les epsilon-transitions afin de travailler avec une représentation HTK.

4 Approches proposées

Dans un premier temps, nous souhaitons valider l'approche en simplifiant certaines étapes du décodage. Pour cette raison, l'ensemble des travaux présenté est réalisé sur des graphes de mots extraits du SRAP KALDI. Nous avons volontairement utilisé un modèle de langage initial bi-gramme afin de ne pas introduire d'information supplémentaire au niveau linguistique. Nous utilisons donc l'algorithme de colonies de fourmis pour étendre le graphe à des ordres de n-gramme supérieurs. Notre implémentation générale de l'algorithme est la suivante : À chaque arc du graphe est associé une variable de type phéromone (un compteur) initialisée à 1 au début de chaque tour. Nos fourmis sont lancées à partir du noeud d'entrée du graphe. Pour sortir du noeud, elles vont devoir choisir un arc. Ce choix est effectué via un tirage aléatoire dont la fonction de distribution dépend de la présence de phéromones sur chaque arc (et éventuellement de facteurs locaux). À l'arrivée de la fourmi sur le noeud final, si la solution trouvée – c'est-à-dire le produit des probabilités

acoustiques et linguistiques du chemin parcouru : $\prod_{p=debutChemin}^{finChemin} P_n(W_p|w_{p-n}..w_{p-1}).P(X_p)$ – est meilleure que la précédente meilleure solution trouvée ; alors des phéromones sont déposées sur le chemin. Dans la littérature, il a été montré qu'avec une version aussi simple des colonies de fourmis, ces dernières ont tendance à converger sur un chemin qui ne représente pas forcément la solution optimale. Il est donc nécessaire d'utiliser des heuristiques locales ou globales pour éviter ce type de phénomène. La sous-section suivante présente quelques heuristiques locales biaisant le choix des fourmis.

4.1 Fourmis anamorphiques

Nous proposons différentes fourmis, comme proposé dans (Bontoux, 2008), qui utilisent des heuristiques d'exploration biaisées par l'information déjà présente dans le graphe initial.

Des probabilités *a posteriori* sont ainsi calculées à partir du graphe bi-gramme. Cette probabilité est calculée pour chaque arc a étant donné le graphe de mot courant G : $p(a|G)$. Elle correspond au ratio entre l'ensemble des probabilités de tous les chemins passant par l'arc a et l'ensemble de tous les chemins dans le graphe G : $p(a|G) = \frac{\sum_{C \in G, a \supset C} p(C|G)}{\sum_{C \in G} p(C|G)}$. Où $C \in G$ désigne un chemin C complet dans le graphe G et $a \supset C$ un arc qui est compris dans ce chemin. Cette probabilité *a posteriori* est calculée classiquement par via un algorithme *forward-backward* comme présenté dans (Wessel *et al.*, 2000).

Nous proposons différents types de fourmis qui sont testés indépendamment :

— Les fourmis à l'écoute (1) : celles-ci sont influencées par la probabilité *a posteriori* acoustique du noeud suivant, sans prendre en compte l'aspect linguistique.
— Les fourmis verbeuses (2) : à l'inverse des fourmis précédentes, ces dernières sont uniquement influencées par la probabilité *a posteriori* linguistique du noeud suivant.
— Les fourmis oracles (3) : c'est un modèle hybride qui combine les deux précédentes et qui représente la probabilité *a posteriori* classique du noeud suivant.

Les différentes probabilités *a posteriori* sont calculées via un algorithme Viterbi appliqué sur le graphe bi-gramme en annulant respectivement les modèles acoustiques (fourmi (1)) et linguistiques (fourmi (2)) ou en conservant le score initial pour la fourmi (3).

Ces différents types de fourmis permettent d'orienter l'exploration en fonction d'informations linguistiques ou acoustiques. Dans la prochaine section, nous présentons les performances de ces fourmis.

Enfin, nous proposons de combiner l'exploration des différentes fourmis en fusionnant leurs phéromones déposées. Pour ce faire, nous faisons travailler chaque type de fourmi sur une copie du graphe. Une fois les graphes explorés, ils sont fusionnés en un seul où les quantités de phéromones sont additionnées. Quelques fourmis sont alors relancées sur ce nouveau graphe et influencées par toutes les phéromones déposées.

Pour l'ensemble des expériences présentées, le nombre de périodes est initialisé à 30 et le nombre de fourmis à 2000, ce qui correspond à 60000 chemins explorés pour chaque graphe. La durée moyenne d'un décodage avec ces paramètres est d'environ 5 secondes par graphe (sur un Intel Xeon 2.5 Ghz). Lors de l'utilisation de plusieurs fourmis, nous avons réalisé 10 périodes avec chacune des fourmis avant de fusionner leurs graphes.

L'algorithme détaillé est présenté ci-dessous.

Algorithme 1 : Algorithme des colonies de fourmis appliqué au décodage de graphes de RAP

```
// 2000 fourmis et 30 périodes dans nos expériences;
Initialiser N_fourmis;
Initialiser N_periodes;
periode ← 0;
fourmis ← 0;
MeilleurScore ← −inf;
MeilleursChemins ← {};
pour tous les Noeuds N ∈ Graphe faire
    Initialiser probabilité a posteriori φ(N);
fin
tant que periode < N_periodes faire
    // Initialisation de l'ensemble des dépôts de phéromones, puis prise en
    considération des précédents meilleurs chemins;
    pour tous les Noeuds N ∈ Graphe faire
        Ph_N ← 1;
    fin
    pour tous les Noeuds N ∈ MeilleursChemins[0...periode] faire
        Ph_N ← Ph_N + 1;
    fin
    fourmi ← 0;
    tant que fourmi < N_fourmis faire
        Chemin ← {};
        Chemin=TrouverChemin(PremierNoeud);
        si Evaluer(Chemin) > MeilleurScore alors
            MeilleurScore ← Evaluer(Chemin);
            MeilleursChemins[periode] ← Chemin;
            pour tous les Noeuds N ∈ MeilleursChemins[periode] faire
                Ph_N ← Ph_N + 1;
            fin
        fin
        fourmi ← fourmi + 1;
    fin
fin
TrouverChemin(Noeud)
    NoeudSuivant = SelectionnerNoeudSuivant(Noeud) ;
    si NoeudSuivant = NoeudFinal alors
        Retourner NoeudSuivant
    fin
    Retourner {NoeudSuivant + TrouverChemin(NoeudSuivant)};
SelectionnerNoeudSuivant(Noeud)
    // Un arc de sortie est sélectionné aléatoirement en fonction d'une
    distribution discrète dépendant de la quantité de phéromones Ph et des
    probabilités a posteriori φ();
    Retourner TirageAleatoire({Liens_Noeud}, {Ph_NoeudSuivant * φ(NoeudSuivant)});
Evaluer(Chemin)
    // P_n(W) est un modèle de langage d'ordre n et P(X) le modèle acoustique;
    Score = Σ_{p=debutChemin}^{finChemin} (logP_n(W_p|w_{p−n}..w_{p−1}) + logP(X_p));
    Retourner Score ;
```

Données	Système de référence	Fourmi (3)	Fourmi (2)	Fourmi (1)	Fourmis (1)+(2)+(3)	Temps de Calcul référence/fourmis
Dev 2g	22.0%	22.0%	31.0%	29.3%	22.0%	1h/1h
Dev 3g	18.8%	20.8%	29.2%	26.4%	20.2%	13h/1h
Dev 4g	17.6%	19.5%	28.7%	26.0%	19.0%	18h/1h
Test 2g	21.6%	21.6%	30.1%	28.7%	21.6%	1h30/1h30
Test 3g	18.0%	20.0%	28.2%	25.9%	19.4%	19h/1h30
Test 4g	17.0%	19.1%	27.6%	25.0%	18.7%	27h/1h30

TABLE 2 – Résultats, en terme de TEM, des décodages avec l'algorithme des fourmis, les décodages avec les fourmis présentent des oscillations de $+/-$ 0.2 point d'une exécution à l'autre.

5 Expériences préliminaires et résultats

Le tableau 2 présente les résultats (en terme de TEM) obtenus avec l'algorithme présenté précédemment. L'expérimentation confirme qu'orienter au départ les fourmis sur les chemins optimaux influence grandement le résultat final. Les fourmis lancées sur des chemins privilégiant l'acoustique (fourmis (1)) ou la linguistique (fourmis (2)) obtiennent de piètres résultats, mais fusionnées avec les fourmis plus hétérogènes (fourmis (3)) elles apportent une information significative.

Ces premières expériences montrent que l'algorithme de colonies de fourmis arrive à étendre le graphe, sans toutefois atteindre les résultats étalons. Par contre, d'un point de vue temps de calcul, l'algorithme basé sur les fourmis est près de 15 fois plus rapide qu'une méthode d'extension classique : la configuration présentée dans le tableau est celle atteignant les scores maximaux ; si des itérations sont rajoutées, les colonies continuent de converger sur le même chemin.

Les résultats en demi-teinte sont normaux étant donné la simplicité de l'algorithme utilisé actuellement. Dans la littérature, il a été montré que les heuristiques permettant au départ de distribuer les phéromones sur les noeuds avaient une importance capitale sur l'évolution de la fourmilière.

Par ailleurs (Dorigo et Gambardella, 1997) montre qu'un des soucis majeur de ce type d'algorithme est sa convergence vers des optimums locaux. Ce type de comportement peut être évité en utilisant plusieurs dimensions qui seront fusionnées *a posteriori*, comme présenté dans (Bontoux, 2008). La fusion des trois types de fourmis introduit cette heuristique et montre son efficacité.

6 Discussion sur les colonies de fourmis

6.1 Avantages

La parallélisation de l'algorithme des colonies de fourmis est triviale. En effet, les fourmis peuvent évoluer indépendamment les unes des autres à la recherche de nouveaux chemins. La parallélisation est également envisageable à un plus haut niveau : celui des colonies de fourmis. Différentes colonies peuvent évoluer sur des copies d'un même graphe puis échanger leurs informations en fusionnant les copies (c'est-à-dire en additionnant la quantité de phéromone présente sur chaque noeud). Un autre point non négligeable est la quantité de mémoire nécessaire. Elle est limitée à l'espace requis pour stocker le graphe en mémoire. Étant donné qu'il n'y a pas d'extension physique, la mémoire utilisée est constante. Dans nos expériences, l'algorithme utilise

en moyenne 100Mo contre plusieurs Go lors d'une extension classique. L'évaluation de la solution se fait lorsqu'une fourmi arrive sur le noeud final du graphe. Le nombre d'évaluations étant relativement faible (2000 par graphe dans notre cas), il est envisageable de rajouter d'autres informations affinant la solution (sémantique, modèle de langage à base de réseau de neurones etc.) Enfin, le temps de calcul peut être parfaitement maîtrisé : le nombre d'évaluations est connu à l'avance et son temps dépend uniquement de la profondeur du graphe.

6.2 Inconvénients

À moins de mémoriser la graine de génération aléatoire, les résultats de l'algorithme ne sont pas déterministes. Au cours de nos expériences, nous avons observé des différences de 0.2 point entre différentes explorations. Rien ne garantit l'optimalité de l'algorithme et ce dernier peut avoir tendance à se laisser piéger par des optimums locaux. Cependant, la littérature propose des solutions pour palier ces problèmes.

6.3 Évolutions envisagées

Le travail présenté ici est préliminaire. Nous souhaitons explorer et analyser plus en profondeur le paradigme des colonies de fourmis appliqué aux SRAP. Les prochaines évolutions se traduiront par l'utilisation en parallèle de fourmis ayant des comportements différents. Ainsi, une sélection des individus performants se fera au fur et à mesure de l'exploration. Il serait également intéressant d'analyser en détail de quelle manière sont explorés les graphes : cet aspect permettra d'optimiser les stratégies d'exploration. Enfin, à plus long terme, nous souhaiterions appliquer les colonies de fourmis dès le début de la chaîne du SRAP : du treillis de phonèmes à l'extension linguistique.

7 Conclusion

Dans cet article, nous avons présenté un nouveau paradigme permettant d'étendre des graphes de mots issus d'un système de reconnaissance automatique de la parole. Ce paradigme, basé sur les colonies de fourmis, permet d'explorer le graphe afin d'y trouver un chemin optimum sans avoir à le construire dynamiquement en fonction des historiques. À l'heure actuelle, notre méthode ne permet pas d'atteindre les résultats de notre système étalon où les graphes ont été étendus avec des méthodes classiques. Cependant, nous montrons que le paradigme fonctionne : l'application d'un modèle de langage d'ordre supérieur améliore systématiquement le décodage initial. L'implémentation présentée est très basique, nous souhaitons à court terme affiner les heuristiques d'exploration afin d'obtenir une approche compétitive avec celles existantes.

Références

AUBERT, X. L. (2002). An overview of decoding techniques for large vocabulary continuous speech recognition. *Computer Speech & Language*, 16(1):89–114.

BONABEAU, É. et THÉRAULAZ, G. (2000). L'intelligence en essaim. *Pour la science*, (271):66–73.

BONTOUX, B. (2008). *Techniques hybrides de recherche exacte et approchée : application à des problèmes de transport*. Thèse de doctorat, Université d'Avignon et des pays de Vaucluse.

DENEUBOURG, J.-L., GROSS, S., FRANKS, N. et PASTEELS, J.-M. (1989). The blind leading the blind : Modeling chemically mediated army ant raid patterns. *Journal of Insect Behavior*, 2:719–725.

DORIGO et STÜTZLE (2004). *Ant Colony Optimization*. MIT-Press.

DORIGO, M. et GAMBARDELLA, L. (1997). Ant colony system : A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1:53–66.

DROGOUL, A. (1993). When ants play chess (or can strategies emerge from tactical behaviors). *In Maamaw'1993*.

GALES, M. (1998). Maximum likelihood linear transformations for hmm-based speech recognition. *Computer Speech and Language*, 12:75–98.

GOPINATH, R. A. (1998). Maximum likelihood modeling with gaussian distributions for classification. *In Proceedings of ICASSP*, pages 661–664.

JELINEK, F. (1969). A fast sequential decoding algorithm using a stack. *IBM J. Res. Develop.*, 13.

LECOUTEUX, B. (2008). *Reconnaissance automatique de la parole guidé par des transcriptions a priori*. Thèse de doctorat, Université d'Avignon et des pays de Vaucluse.

MONMARCHE, N., GUINAND, F. et SIARRY, P., éditeurs (2009). *Fourmis artificielles et traitement de la langue naturelle*. Lavoisier, Prague, Czech Republic.

NEY, H., HAEB-UMBACH, R., TRAN, B.-H. et OERDER, M. (1992). Improvements in beam search for 10000-word continuous speech recognition. *In Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing ICASSP-92*, volume 1, pages 9–12 vol.1.

NOCERA, P., LINARES, G. et MASSONIÉ, D. (2002). Principes et performances du décodeur parole continue speeral. *In XXIVées journées d'étude sur la parole*. Laboratoire Informatique d'Avignon.

ORTMANNS, S. et NEY, H. (2000). The time-conditioned approach in dynamic programming search for. 8(6):676–687.

PAUL, D. (1991). Algorithms for an optimal a* search and linearizing the search in the stack decoder. *In Proc. International Conference on Acoustics, Speech, and Signal Processing ICASSP-91*, pages 693–696 vol. 1.

PAUL, M., FEDERICO, M. et SEBASTIAN (2010). Overview of the iwslt 2010 evaluation campaign. *In IWSLT 2010*.

POVEY, D., GHOSHAL, A., BOULIANNE, G., BURGET, L., SCHWARZ, O. G. N. G. M. H. P., SILOVSKY, J., STEMMER, G. et VESELY, K. (2011). The kaldi speech recognition toolkit. *In* SOCIETY, I. S. P., éditeur : *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*, volume IEEE Catalog No. : CFP11SRW-USB.

ROUSSEAU, A., DELÉGLISE, P. et ESTÈVE, Y. (2012). TED-LIUM : an automatic speech recognition dedicated corpus. *In LREC 2012*.

SCHWAB, D., GOULIAN, J. et TCHECHMEDJIEV, A. (2013). Désambiguïsation lexicale de textes : éfficacité qualitative et temporelle d ?un algorithme à colonies de fourmis. *journal Traitement Automatique des Langues, vol. 54-1*.

WESSEL, F., SCHLÜTER, R. et NEY, H. (2000). Using posterior word probabilities for improved speech recognition. *In* SCHLUTER, R., éditeur : *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing ICASSP '00*, volume 3, pages 1587–1590 vol.3.