Principes et performances du décodeur parole continue *Speeral*

NOCERA Pascal, LINARES Georges, MASSONIÉ Dominique

Laboratoire d'Informatique d'Avignon
LIA, Avignon, France
Tél.: ++33 (0)4 90 84 35 07 - Fax: ++33 (0)4 90 84 35 01
Mél: { pascal.nocera, georges.linares, dominique.massonie }@lia.univ-avignon.fr

ABSTRACT

This paper presents the continuous speech recognition system *Speeral* developed in the LIA. *Speeral* uses a modified A* algorithm to find in the search graph the best path taking into account acoustic and linguistic constraints. Rather than words by words, the A* used in *Speeral* is based on a phoneme lattice previously generated. To avoid the backtraking problems, the system keeps for each frame the deepest nodes of the lexical tree (partially explored) starting at this frame. If a new hypothesis to explore is ended by a word and the lexicon starting where this word finishes has already been developed, then the next hypothesis will "jump" directly to the deepest nodes.

1. INTRODUCTION

L'objectif des systèmes de Reconnaissance Automatique de la Parole (RAP) est de trouver la meilleure phrase (liste de mots) en tenant compte de contraintes acoustiques et linguistiques [deMori98]. Pour effectuer cette recherche, certains systèmes utilisent l'algorithme de recherche A* [Pearl84].

L'algorithme A* est un algorithme asynchrone. L'ordre d'exploration des nœuds *n* est donné par la valeur d'une fonction d'évaluation *F(n)* représentant une estimation du coût du meilleur chemin passant par ce nœud *n*. Ceci signifie qu'il est envisageable de revenir sur une hypothèse (théorie) très éloignée de la théorie la plus avancée, parce qu'elle possède maintenant la meilleure fonction d'évaluation.

Comme la meilleure phrase doit être trouvée parmi la liste de toutes les phrases possible, la structure du graphe de recherche dans lequel doit évoluer l'algorithme est celle d'un arbre constitué de la concaténation d'arbres lexicaux. Chaque feuille du lexique (mot complet) est reliée à un autre lexique. L'algorithme A* appliqué à un tel graphe va explorer les mêmes arcs plusieurs fois pour un début de phrase différent (historique).

C'est pourquoi l'algorithme A* est la plupart du temps utilisé sur des graphes de mots plutôt que sur des unités de plus petite taille (phonèmes, HMM, ..). La progression des hypothèses se fait alors mots par mots. Il est utilisé dans des systèmes multi-passes pour chercher la meilleure phrase parmi un treillis de mots [Huang93], ou bien dans des systèmes à une seule passe après l'application d'un algorithme de recherche rapide (*fast match*) qui isole une liste de mots candidats pouvant prolonger la théorie courante [Paul91]. Cependant, dans les deux cas, le nombre de mots doit être suffisamment élevé pour obtenir de bon résultats.

Pour palier à ce problème, nous proposons de baser notre recherche sur les phonèmes plutôt que sur les mots. La progression de notre algorithme se fait donc phonème par phonème. Pour optimiser la recherche et ne pas parcourir à nouveau, lors d'un retour en arrière, les chemins déjà parcourus, nous conservons pour chaque arbre lexical exploré (même partiellement) les nœuds les plus profonds qui correspondent soit à des mots complets soit à des « débuts de mots ». Lors d'un *backtraking*, le lexique déjà exploré ne le sera plus, et l'algorithme de recherche ira directement aux nœuds conservés, qui seront accrochés à l'hypothèse courante pour former de nouvelles hypothèses.

Après une présentation succincte de l'algorithme A* classique, nous expliquerons les modifications apportées pour son application à un treillis de phonèmes. Ensuite, le système de RAP du Laboratoire d'Informatique d'Avignon (LIA) baptisé *Speeral* sera détaillé. En dernier point, nous ferons une analyse des résultats obtenus sur le corpus de la campagne de test ARC B1 de l'AUPELF.

2. L'ALGORITHME A*

L'algorithme A* est un algorithme de recherche du meilleur chemin dans un graphe. Pour cela, il utilise une fonction d'évaluation *F(n)* pour chaque nœud exploré. Cette valeur représente une estimation du coût du meilleur chemin passant par le nœud *n*. Elle est obtenue en sommant le coût réel *g(n)* du chemin du début du graphe à *n*, et une estimation du coût *h(n)* (appelée sonde) du chemin restant à parcourir (du nœud *n* à la fin du graphe).

Cet algorithme utilise une pile nommée *Open* contenant la liste des nœuds à explorer, triée en fonction des valeurs de *F*. A chaque itération, le nœud *n* au sommet de *Open* est sorti, et pour chaque successeur *m* de ce nœud dans le graphe, la valeur $F(m)=g(n)+c(n,m)+h(m)$ est calculée (figure 1). Les nouvelles hypothèses sont insérées dans la pile, le terme $c(n,m)$ représente le coût de l'arc *n* ? *m*.

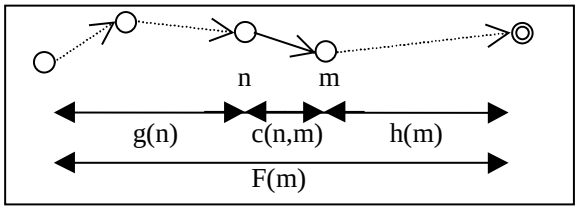

Figure 1: Calcul de la fonction d'évaluation *F*.

L'algorithme s'arrête lorsque le nœud au sommet de la pile est un nœud terminal du graphe. Il est démontré que si la fonction *F* est toujours supérieure ou égale au coût du meilleur chemin, l'algorithme s'arrêtera sur le meilleur chemin. Cette condition sur la fonction d'évaluation est assurée par une sonde *h* optimale.

3. ALGORITHME ADAPTÉ

L'algorithme A* est difficilement applicable tel quel au problème de la RAP. Dans ce cadre, le graphe est constitué de la concaténation de mots (lexique) et les scores acoustiques et linguistiques sont matérialisés par les arcs. Afin d'utiliser l'algorithme A* sur un treillis de phonèmes plutôt que sur un treillis de mots, nous avons dû l'adapter en tenant compte des caractéristiques d'un tel graphe de recherche.

3.1 Graphe de recherche

Les données acoustiques utilisées pour la recherche sont exprimées sous la forme d'un treillis de phonèmes. Ce treillis contient une liste d'hypothèses de phonèmes *(p, deb, fin, sc)* où *sc* représente un score de vraisemblance du phonème *p* de la trame *deb* à la trame *fin*.

Le lexique est quant à lui décrit par une liste de phonèmes pour chaque mot du vocabulaire. Il est représenté sous la forme d'un arbre à tête commune où tous les mots commençant par les mêmes phonèmes partagent les mêmes arcs. Chaque feuille de l'arbre représente un mot (figure 2).

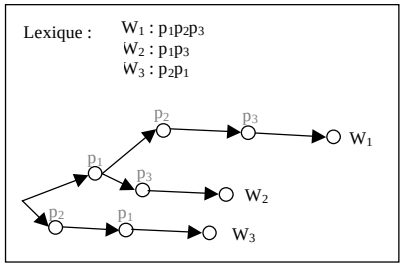

Figure 2 : Représentation du lexique.

Le graphe de recherche est constitué de la concaténation d'arbres lexicaux partiels. Un nouveau lexique commence à chaque fois qu'un mot se termine. Les arbres lexicaux partiels sont la projection de l'arbre lexical complet sur le treillis de phonèmes. Ainsi, une chaîne phonétique présente dans le lexique mais absente dans le treillis ne se trouvera plus dans un lexique partiel.

3.2 Score linguistique

En plus du score acoustique donné par le treillis, l'algorithme de recherche tient compte du score linguistique de l'hypothèse en train d'être décodée. Le score linguistique est calculé par deux fonctions selon la nature du dernier nœud de l'hypothèse :

- la fonction *Mot_ML* est appliquée si la nouvelle hypothèse est terminée par un mot. Elle reçoit en paramètre la liste des mots la constituant et détermine son score linguistique.
- La fonction *Part_ML* est appliquée si la nouvelle hypothèse se termine par un nœud de l'arbre (début de mots). Cette fonction donne un score linguistique en anticipant la liste des hypothèses pouvant être générées à partir de ce nœud.

$$\text{Part\_ML}(w_1..w_k,n) = \text{Argmax}_{Wn}(\text{Mot\_ML}(w_1..w_k w_n))$$

Où w_n est une feuille (un mot) du sous-arbre commençant au nœud *n*. Cette anticipation linguistique nous permet de pénaliser au plus tôt les branches lexicales aboutissant à des mots improbables.

3.3 Calcul de la sonde

La sonde *h* retenue pour notre recherche doit représenter un coût optimal du chemin partant de la trame *t* jusqu'à la fin du graphe (de la phrase). Cette sonde est exclusivement acoustiques et calculée par l'application de l'algorithme de Viterbi arrière sur le treillis de phonèmes. Cette sonde est optimale puisque les contraintes lexicales et linguistiques ne sont pas prises en compte, alors que dans le décodage, elles le seront.

3.4 Description de l'algorithme

Afin de ne pas parcourir plusieurs fois des parties du graphe de recherche, le système conserve pour chaque trame, les nœuds les plus profonds de l'arbre lexical partiellement exploré commençant à cette trame. Si l'hypothèse courante est terminée par un mot et que le lexique commençant à la fin de ce mot a déjà été exploré, les hypothèses suivantes héritent de la recherche. Elles « sautent » directement aux nœuds les plus profonds du lexique, sans recommencer le parcours à partir de la racine.

Données Manipulées :

- *Hyp* représente un nœud du graphe de recherche. Chaque *Hyp* pointe sur le nœud d'un arbre lexical partiel et est associé à une trame de fin.
- *Tab_Lex[t]* contient la liste des *Hyp* pour le lexique commençant à la trame *t*.
- *Tab_Mot[t]* contient la liste des débuts de phrases déjà explorées se terminant à la trame *t*. Ces débuts de phrases constituent les différentes hypothèses de mots « complets » qui ont déjà été traitées.

Description :

Comme nous l'avons expliqué précédemment, seuls les nœuds les plus profonds de chaque lexique sont conservés dans *Tab_Lex* (Figure 3). Ces nœuds correspondent soit à des débuts de mots soit à des mots complets (feuilles). Si l'algorithme « backtrack », ce tableau permettra de ne pas explorer à nouveau les mêmes branches du graphe.

L'algorithme produit de nouvelles hypothèses à partir du nœud *Hyp* au sommet de *Open* (la meilleure théorie courante) jusqu'à la trame finale et :

- si *Hyp* est un phonème (un nœud d'un arbre lexical), pour chaque *New* successeur de *Hyp* :
  - si *New* est un phonème, il est mis dans *Tab_Lex* à la place de *Hyp* et la meilleure des hypothèses allant du début de la phrase jusqu'à *New* est mise dans *Open* (Figure 4).
  - si *New* est un mot, toutes les hypothèses de début de phrase se terminant par *New* sont stockées dans *Tab_Mot* et la meilleure est mise dans *Open*.
- si *Hyp* est un mot (une feuille d'un arbre lexical) :
  - si le lexique commençant à la trame de fin *t* de *Hyp* a déjà été exploré, toutes les nouvelles hypothèses contenant *Hyp* et suivies par les nœuds conservés dans *Tab_Lex[t]* sont générées (Figure 5).
  - sinon, l'exploration d'un nouveau lexique est commencée et les nouveaux nœuds explorés sont stockés dans *Tab_Lex[t]* (Figure 6).

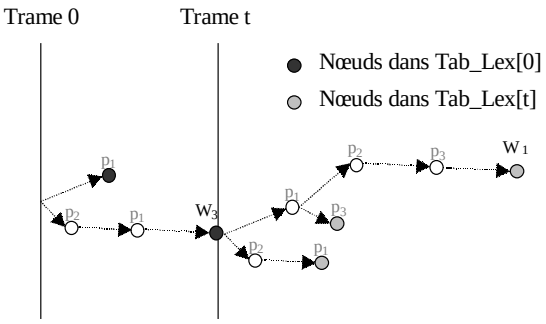

Figure 3 : Exemple d'état de Tab_Lex

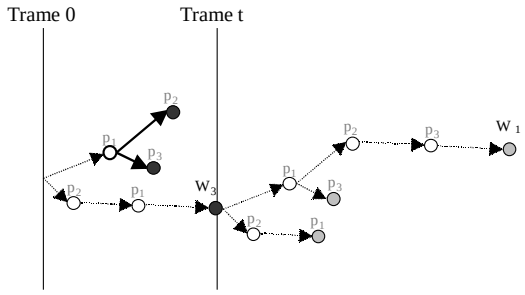

Figure 4 : L'exploration du nœud p_1 génère deux nœuds p_2 et p_3. Ces nœuds sont insérés dans *Open* et prennent la place de p_1 dans Tab_Lex[0]

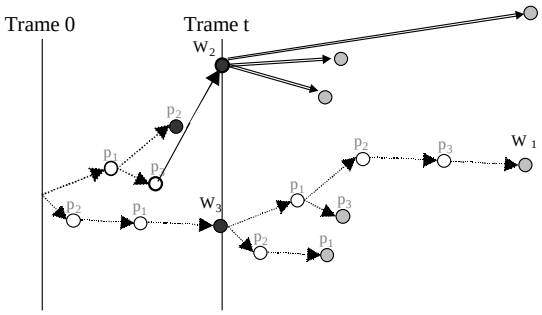

Figure 5 : Le mot W_2 est prolongé par les nœuds contenus dans Tab_Lex[t].

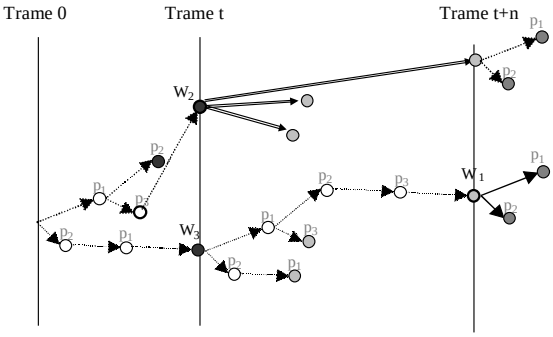

Figure 6 : Le lexique commençant à la trame t+n est « ouvert » et les premiers nœuds p1 et p2 stockés dans Tab_Lex[t+n].

4. LE SYSTÈME *SPEERAL*

Le système de RAP continue grand vocabulaire du LIA utilise pour la recherche de la meilleure hypothèse l'algorithme A* modifié présenté précédemment. La modélisation acoustiques est réalisée à l'aide de modèle de Markov cachés et le treillis de phonèmes est constitué des *n* meilleures hypothèses de phonèmes par trame. Ce treillis est obtenu par un Viterbi arrière qui en même temps calcule la sonde *h* nécessaire au A*.

La modélisation du langage est réalisée à l'aide de modèles tri-grammes pour le calcul de la fonction *Mot_ML*. De plus, une liste de mots est associée à chaque nœud de l'arbre lexical afin de calculer la fonction *Part_ML* à l'aide des tri-grammes issus de cette liste.

4.1 Limitation de la liste Open

Afin de limiter la taille de la liste *Open* et donc le nombre d'hypothèses à explorer durant le parcours du graphe, nous avons été amené à définir des critères de coupure lors de la recherche.

La première coupure est liée à la limitation de l'historique pris en compte par le ML tri-gramme. Ainsi, pour deux hypothèses ayant les deux derniers mots identiques, seules la meilleure sera conservée puisque rien par la suite ne pourra changer cet ordre. Cette coupure n'affecte pas la capacité de l'algorithme à trouver le meilleur chemin mais le limite à la recherche d'une seule solution (1-best).

La deuxième coupure est liée à la qualité de la sonde qui résulte uniquement d'un décodage acoustico-phonétique

entre chaque trame et la fin de la phrase. Comme elle ne tient pas compte du modèle de langage et que les suites phonétiques ne sont pas contraintes par les branches du lexique, sa qualité n'est pas excellente. C'est pour cela que nous interdirons les retours arrière trop importants. Si une hypothèse est trop éloignée de l'hypothèse la plus profonde, elle est alors supprimée de *Open*. Contrairement à la coupure précédente, celle-ci peut entraîner la perte de la meilleure solution.

4.2 Résultats expérimentaux

Les résultats présentés sont issus de la tâche ARC B1 de l'AUPELF [Dol00] comportant 20 locuteurs et 300 phrases extraites du journal « Le Monde » de 1997. Les conditions de l'expérience sont celles imposées lors de la campagne : lexique de 2000 mots, ML calculé à partir des textes du « Monde » de 1987 à 1996.

Deux modèles acoustique dépendant du genre ont été utilisés. Ils contiennent 588 diphones droits représentés par un CDHMM (Continuous Density Hidden Markov Model) de 3 états gauche-droite chacun (mixture de 32 gaussiennes). Ces paramètres ont été estimés à l'aide du corpus « Bref » comportant 120 locuteurs. Une adaptation MLLR est effectués sur ces modèles après une première phase de décodage.

Dans ces conditions, le système *Speeral* obtient un taux de WER (Word Error Rate) de 17,2%. L'étude suivante permet d'analyser les performances du système au delà de la simple confrontation avec la référence.

Classes d'erreurs et répartition :

Le jugement de l'apport d'un mot dans une phrase est soumis à des critères objectifs et subjectifs, chaque mot pèse différemment sur le fond et la forme. Cette partie décrit nos critères de classement des mots, l'objectif consistant à étendre l'analyse du taux d'erreurs de *Speeral*. Les erreurs commises lors du décodage sont classées suivant leur impact sur le sens du message (table 1).

Table 1: Répartition des erreurs de *Speeral* en classes.

WER	Lemmes	Accords	Total
Mots filtrés	7.5 %	1.1 %	8.6 %
Mots clefs	5.9 %	2.7 %	8.6 %
Total	13.4 %	3.8 %	**17.2 %**

Mots filtrés : Dans une phrase, tous les mots ne sont pas porteur de sens. Une *Stoplist* d'environ 400 mots a été établie *a priori* avec la fréquence, taille et catégorie syntaxique. Environ 2% du lexique permet de filtrer plus de la moitié du journal « Le Monde ».

Mots clefs : Le corpus correspondant aux mots clefs est composé de l'ensemble des mots non filtrés.

Mots mal accordés : Pour le score, un mot faux est celui dont l'orthographe diffère entre l'hypothèse et la référence. La réponse est simplement binaire et ne satisfait pas notre besoin de juger le poids de l'erreur. Nous avons ainsi remplacé les formes fléchies de chaque mots par leur lemme pour obtenir de nouveaux résultats.

Sources d'erreurs du système :

L'hypothèse retenue est une phrase, la meilleure de celles envisagées par *Speeral* à partir du signal de la référence, des contraintes acoustiques et linguistiques. Les mots d'une hypothèse constituent un chemin parmi tous ceux possibles à chaque trame. Une erreur possède plusieurs origines, le mot correct est peut être non envisagé (problème acoustique), non retenu (faiblesse linguistique) ou inconnu du système. Ce dernier cas s'applique aux mots clefs hors vocabulaire (HV) et constitue 3.6% du corpus de test (table 2).

Table 2: Résultats retenus par *Speeral* (contextuel C ou non-contextuel ¬C) et la meilleure hypothèse envisagée.

WER	Mots HV	Mots clefs	Mots filtrés	Total
Retenu ¬C	3.6 %	4.9 %	14.1 %	22.6 %
Retenu C	3.6 %	5 %	8.6 %	17.2 %
Envisagé	3.6 %	< 2.4 %	< 2.4 %	6 %

Bilan Les mots filtrés profitent seuls du passage au décodeur avec phonèmes contextuels. L'amélioration possible pour les erreurs de mots clefs (8.6%) se mélange donc entre correction de lemmes (5.9%) ou d'accords (2.7%) et sélection de mots non retenus (2.6%), non envisagés (< 2.4%) ou exclus (3.6%).

CONCLUSION

La présentation exhaustive du fonctionnement d'un décodeur de parole continue ne se résume pas en quelques pages. Toutefois, les principes qui font l'originalité de *Speeral* ont été présentés dans cet article. Les résultats obtenus valident l'approche utilisée. Enfin, nous soulignons que le A* apporte une souplesse, dès la première étape du décodage, que les algorithmes classiques de recherche n'ont pas nécessairement.

BIBLIOGRAPHIE

[DeMori98] De Mori R. (1998), « Spoken dialogue with computers », Academic Press

[Pearl84] Pearl J. (1984), Heuristics : Intelligent search strategies for computer problem solving

[Hon93] Hon H.W., Hwang M.Y., Lee K.F, Rosenfeld R., Haung X Alleva F. (1993), « The SPHINX II speech recognition system: An overview », Computer Speech and Language, Vol 7 n°2, pp.137-148

[Paul91] Paul D.B. (1991) « Algorithms for an optimal A* search and linearizing the search in the stack decoder »,ICASSP 91, pp. 693-696.

[Dol00] Dolmazon J., Bimbot F., Adda G., Caerou J., Zeiliger J., Adda-Decker M. (2000), « Première campagne AUPELF d'Evaluation des systèmes de Dictée Vocale », Ressources et Evaluation en ingénierie des langues, pp. 279-307.