Reconnaissance de la parole continue à grand vocabulaire

Modélisation acoustique et décodage

C. Barras
éléments de cours de F. Yvon, A. Allauzen, H. Schwenk, M. Adda-Decker

LIMSI-CNRS & Université Paris-Sud

décembre 2017

Plan

Reconnaître la parole = transcrire les sons

- un cas d'école : la dictée
  $\Rightarrow$ la parole comme interface de saisie
- l'écrit comme support : indexer des documents audio
  $\Rightarrow$ l'écrit comme représentation alternative efficace
- traduction de la parole $\Rightarrow$ l'écrit comme représentation intermédiaire
- interfaces et commandes vocales : pourquoi transcrire ?

Reconnaissance de parole : le paradigme statistique

Le problème

Étant donné un signal $X = X_1 ... X_T$, trouver parmi tous les énoncés possibles celui qui s'accorde le mieux avec le signal.

Décision statistique

Trouver $w^* \in V^*$ tel que :

$$\begin{aligned} w^* &= \operatorname{argmax} P(w_1^n \mid X_1^T) \\ &= \operatorname{argmax} P(X_1^T \mid w_1^n) P(w_1^n) \end{aligned}$$

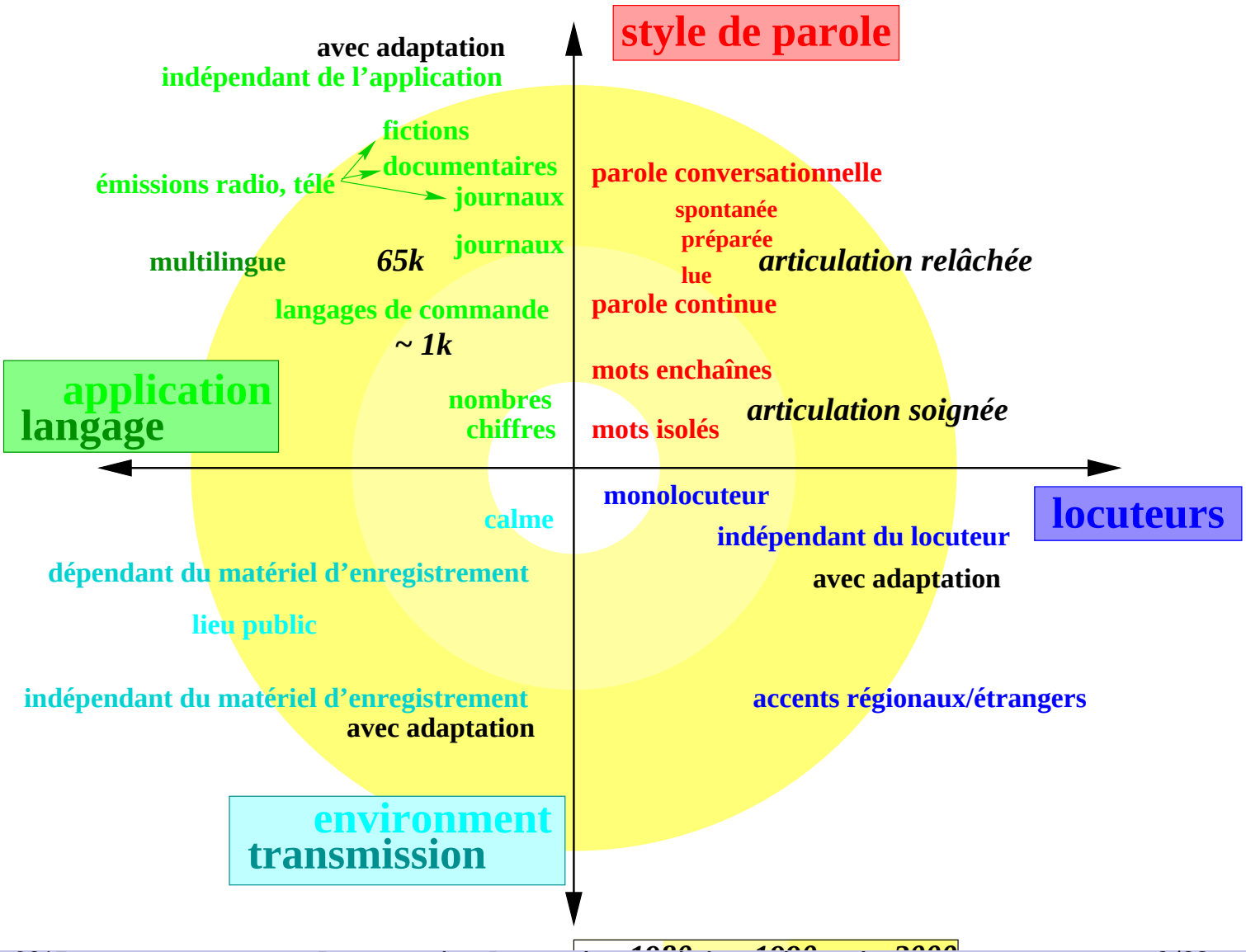
Trois soucis :

- modéliser $P(x_1^T \mid w_1^n)$: fourni par le modèle acoustique
- modéliser $P(w_1^n)$: fourni par le modèle linguistique
- calculer l'argmax : algorithmes de recherche

Reconnaître la parole "naturelle"

- Grand vocabulaire :
  - la modélisation acoustique de chaque mot est impossible (taille des modèles, manque de données, vocabulaire dynamique)
  - forte confusion acoustique (homophones et quasi-homophones) :
    lɛ = {les, lait, laid, lez, laits, laids, l'est, l'es...}
  - l'ensemble des énoncés possibles est infini
- Parole enchaînée (incertitude sur les frontières)
  - augmente la confusion acoustique :
    décodage, *des codes âge*, *dé qu'ode à je*...
  - modéliser la co-articulation et les ajustements phonologiques inter-mots (liaisons, élisions...)
- Parole "spontanée" : lapsus, hésitations, faux départs, reprises, voix superposées, "bruits" (rires, cris, toux), etc

Difficultés de la reconnaissance vocale

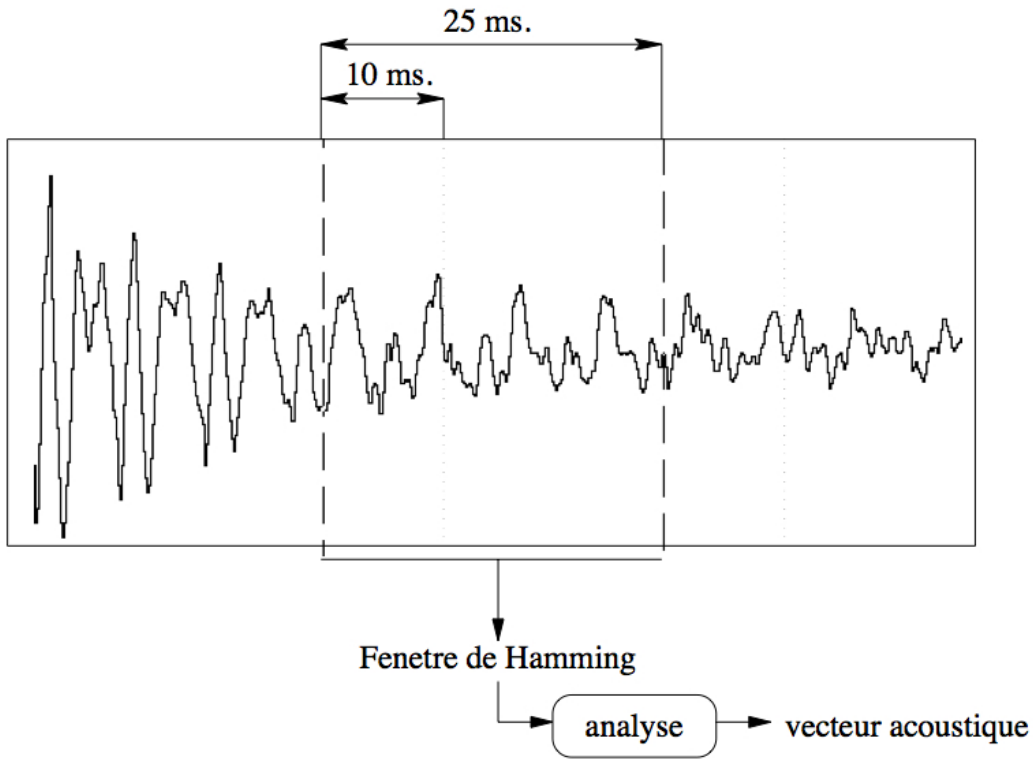


Les solutions qui marchent

- Modélisation acoustique (HMM) d'unités courtes : phones, "triphones"
- Apprentissage statistique (HMM/GMM/DNN) sur des bases de données étiquetées
- Restriction à un vocabulaire de taille fini V
- Représentation phonétique des mots du vocabulaire
- Modélisation (déterministe ou probabiliste) de V^*
- Recherche heuristique dans $H \subset V^*$

La parole ? Un flux de paramètres

- analyse locale de fenêtres se recouvrant ⇒ vecteur de paramètres (cepstres, LPC...)
- modélisation de la dynamique : ajout des "dérivées" premières et seconde



⇒ flux discret de vecteurs acoustiques (*trames*) ($\approx$ 100/s)

Plan

Plan

Probabiliser des séquences : les HMM

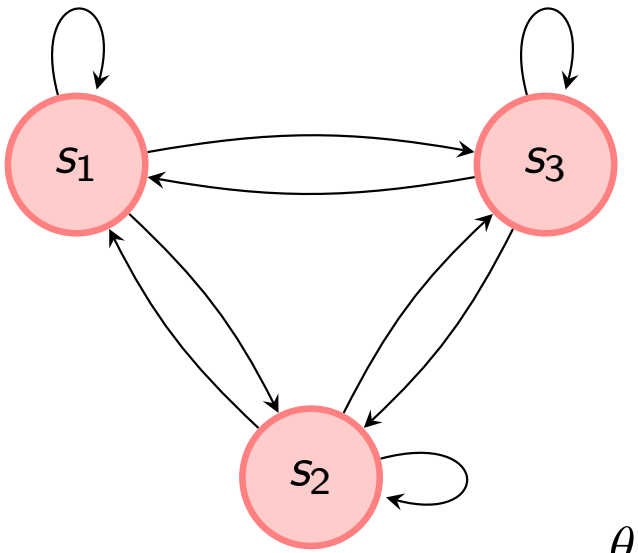
- un HMM est une machine à états
- une matrice stochastique spécifie les changements d'états
- à chaque état est associé un mécanisme de tirage aléatoire sur un ensemble $\mathcal{X}$

Générer des observations avec un HMM

- tirer un état initial q_O, $t = t_0$
- répéter :
  1. tirer l'observation x_t avec le mécanisme de tirage de q_t
  2. tirer le nouvel état q_{t+1}
- exemple : promenade de l'ivrogne

Modèles de Markov Cachés

Définition



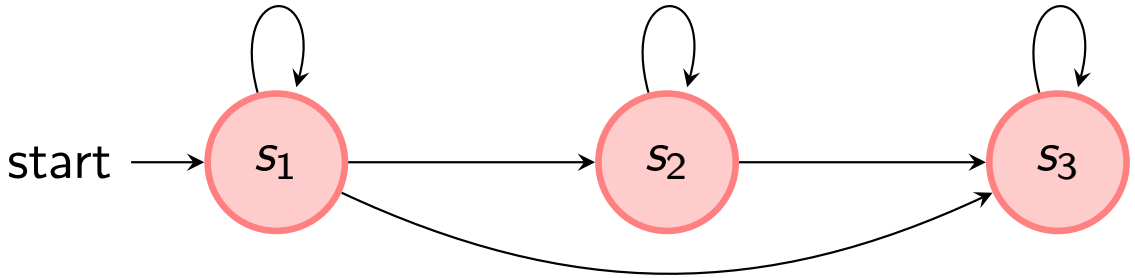
Un modèle de Markov est défini par $(Q, \mathcal{X}, \pi, \mathbf{A}, \mathbf{B})$.

- Q est un ensemble (fini) d'états,
- π le vecteur des probabilités initiales : $\pi(i) = P(q_0 = s_i)$,
- $\mathbf{A}$ la matrice de transition : $a(i,j) = P(q_t = s_j | q_{t-1} = s_i) = P(s_j | s_i)$
- X est l'espace des observations,
- $b(i,x)$ spécifie la loi des observations x pour s_i : $b(i,x) = P(x | q_t = s_i) = P(x | s_i)$

θ note l'ensemble des paramètres : $\pi, \mathbf{A}, \mathbf{B}$

Des HMMs pour les sons

- orientation temporelle : pas de retour (modèle de Bakis)



  $\Rightarrow$ chaque état modélise un segment de la séquence sonore
- la loi des obervations ($\in \mathbb{R}^n$) :
  - un *mélange* de n_i gaussiennes ($\sum_k \lambda_k = 1$)

$$b(x,i) = \sum_{k=1}^{n_i} \lambda_k \frac{1}{(2\pi)^{n/2}|\Sigma_{i,k}|^{1/2}} \exp\left(-\frac{(\mathbf{x}-\mu_{\mathbf{i},\mathbf{k}})^T \Sigma_{i,k}^{-1}(\mathbf{x}-\mu_{\mathbf{i},\mathbf{k}})}{2}\right)$$

    $\Rightarrow$ les composantes du mélange modélisent différentes classes de locuteurs
  - la sortie d'un réseau de neurones

Générer des observations

- un HMM M à p états $s_1 \dots s_{|S|}$, paramétré par θ
- la probabilité de générer une séquence $x_{1,T} = x_1 \dots x_T$ $(T > Q)$ le long du chemin $q_{1,T} = q_1 \dots q_T$?

$$\begin{aligned} P(x_{1,T}, q_{1,T}) &= \pi(q_1) P(x_1|q_1) \prod_{t=2}^{T} P(q_t|q_{t-1}) P(x_t|q_t) \\ &= \pi(q_1) b(x_1, q_1) \prod_{t=2}^{T} a(q_{t-1}, q_t) b(x_t, q_t) \end{aligned}$$

- deux problèmes voisins :
  - la probabilité que $x_{1,T}$ soit émise par M ? $\Rightarrow$ sommer sur toutes les séquences d'états de longueur T
  - la séquence d'états la plus probable sachant $x_{1,T}$? $\Rightarrow$ maximiser sur toutes les séquences d'états.
- résolution par programmation dynamique

Décoder avec un HMM

Position du problème

Connaissant θ, observant $x_{1,T}$, quelle est la séquence d'états la plus probable ?

$$q^*_{1,T} = \text{argmax}\, q_{1,T} P(q_{1,T}|x_{1,T},\theta) = \text{argmax}\, q_{1,T} P(q_{1,T},x_{1,T}|\theta)$$

Résolution par programmation dynamique

$\delta_t(i)$ est la probabilité de la meilleure séquence émettant $x_{1,t}$ et arrivant en t dans l'état s_i :

$$\delta_t(i) = max_{(q_1,\ldots,q_{t-1})} P(q_1,\ldots,q_{t-1},q_t = s_i,x_1,\ldots,x_t|\theta)$$

Algorithme de Viterbi

Calcul itératif de $\delta_t(i)$, en gardant la trace de la meilleure séquence dans $\psi_t(i)$:

1. **Initialisation :**
   $\delta_1(i) = \pi(i)b(q_1, x_1)$ et $\psi_1(i) = 0$ pour $1 \leq i \leq |S|$.
2. **Récurrence** pour $t = 1 \ldots T$:
   $\delta_t(i) = max_{1 \leq j \leq |S|} \delta_{t-1}(j)a(j,i))b(s_i, x_t)$ pour $1 \leq j \leq |S|$
   $\psi_t(i) = \text{argmax}_{1 \leq j \leq |S|}\, \delta_{t-1}(j)a(j,i))b(s_i, x_t)$ pour $1 \leq j \leq |S|$
3. **Fin :**
   $P^* = max_{1 \leq i \leq |S|} \delta_T(i)$, et
   $q_n^* = argmax_{1 \leq i \leq |S|} \delta_T(i)$
4. Le meilleur chemin est obtenu par *backtracking* (retour arrière) :
   $q_t^* = \psi_{t+1}(q_{t+1}^*)$

Complexité ? $O(T * |S|^2)$

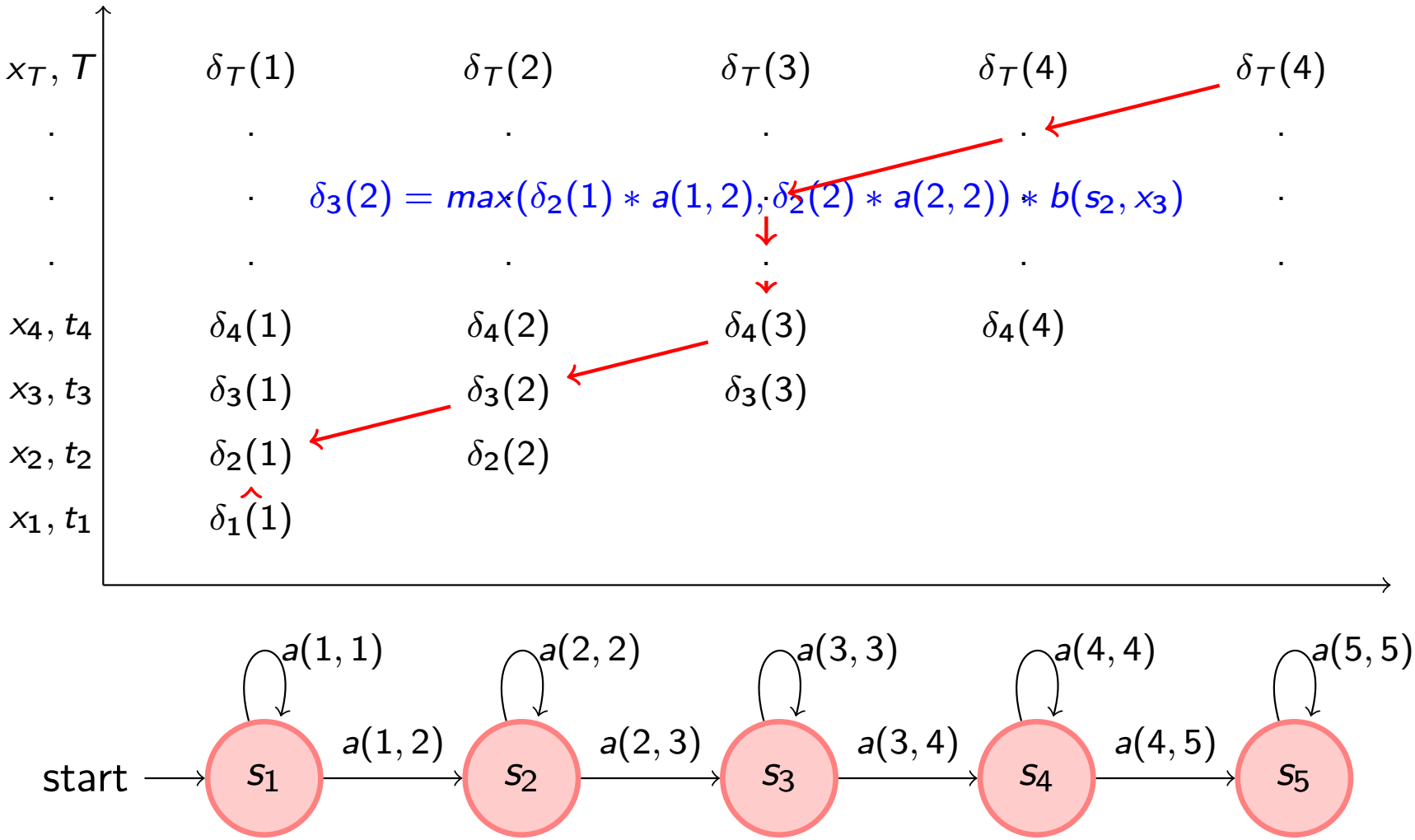
Aligner la parole : le problème

- Données :
  - un HMM (Gauche-Droit) à m états modélisant un mot
  - un enregistrement de ce mot $\{x_t, t = 1 \dots T\}$
- Imputation des trames aux états $|\mathcal{S}|^T$ possibilités ?
  - $q_1 q_1 q_2 q_2 q_2 q_2 q_3 \dots . q_T q_T$
  - $q_1 q_1 q_1 q_2 q_2 q_3 q_3 \dots . q_{t-1} q_T$
  - $q_1 q_1 q_1 q_2 q_2 q_3 q_3 \dots . q_T q_T$
  - ...
- Application : localiser début et fin des différentes parties du mot, apprendre les lois d'émission
- Formellement :

$$\text{argmax}\, q_{1,T} P(q_{1,T}|x_{1,T}, \theta) = \text{argmax}\, q_{1,T} P(q_{1,T}, x_{1,T}|\theta)$$

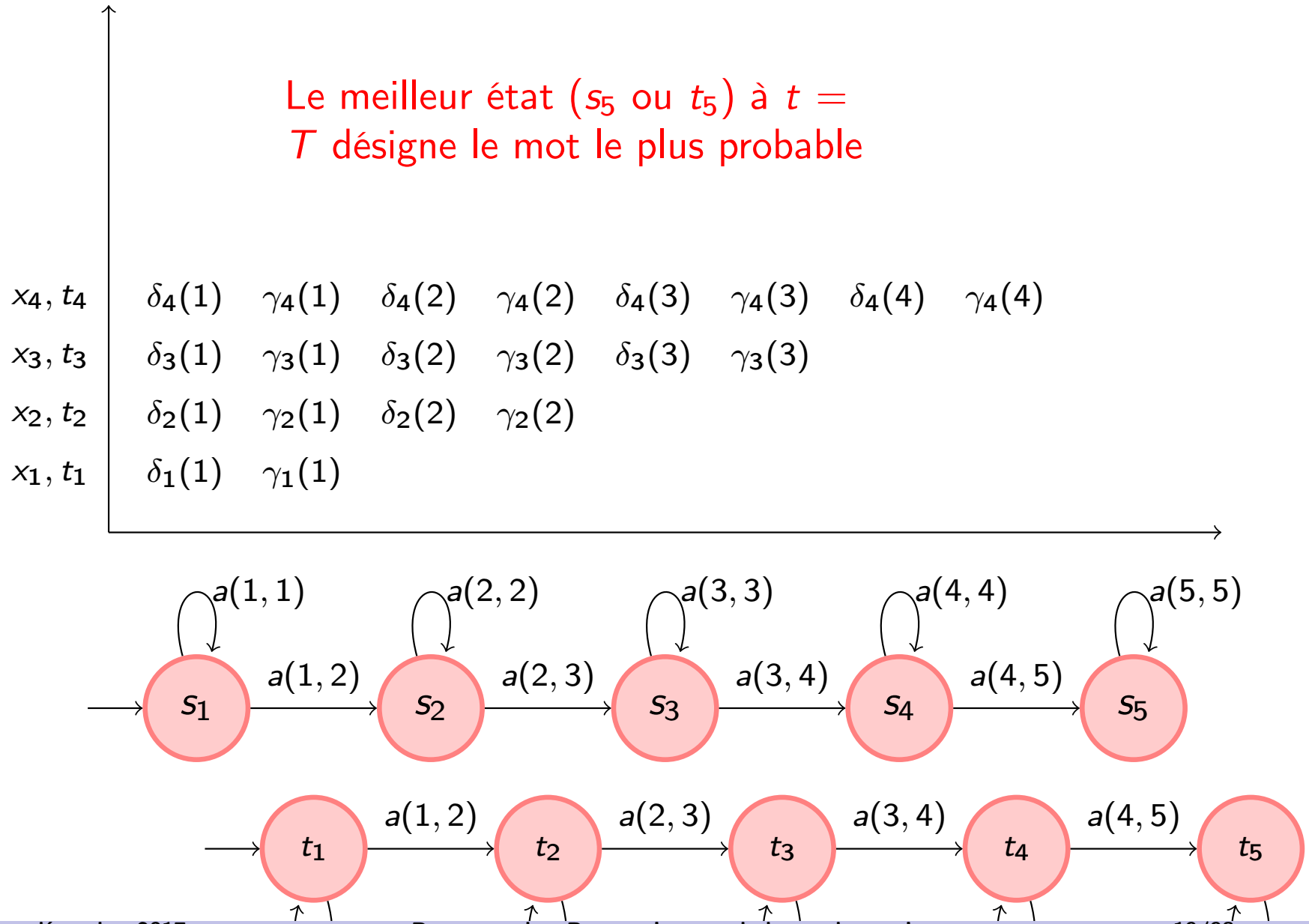
$\Rightarrow$ un problème pour Viterbi !

Aligner la parole avec Viterbi

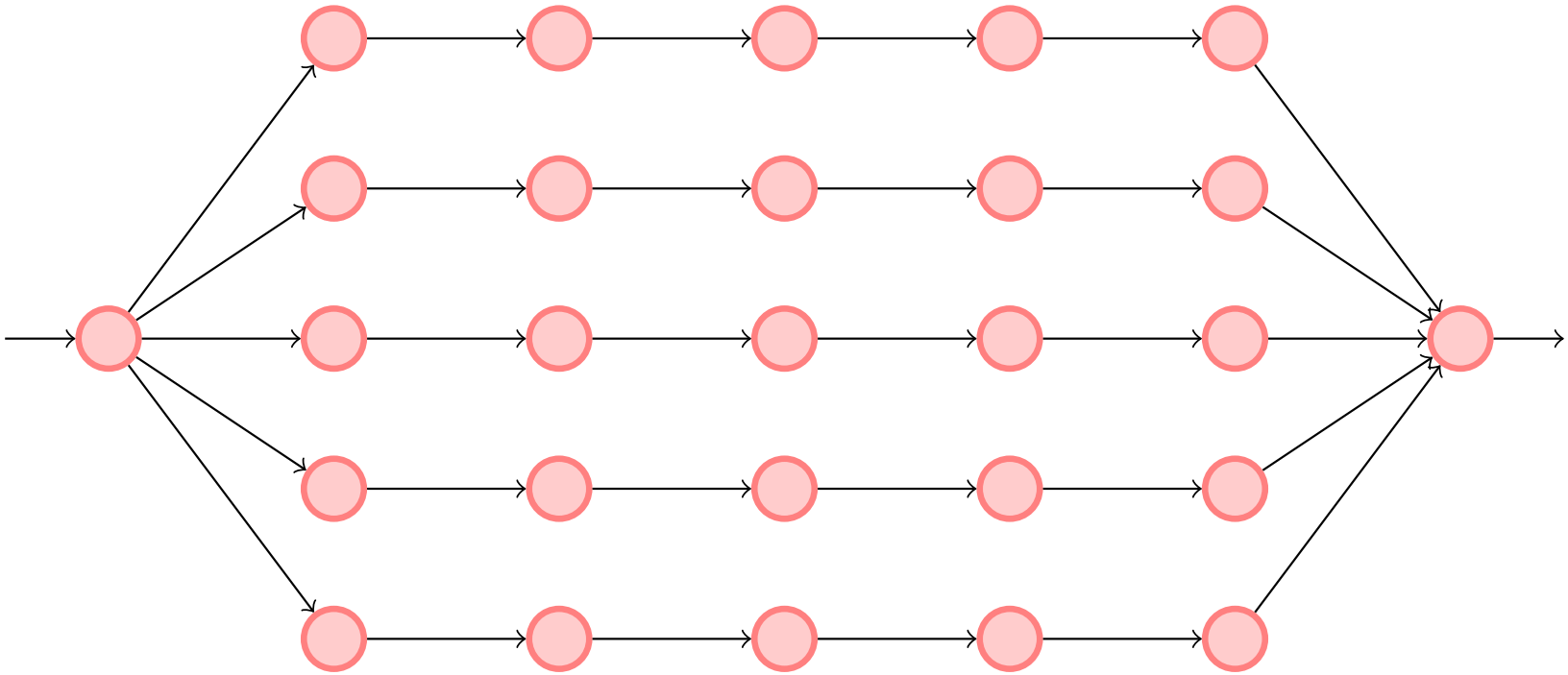


alignement : $s_1/x_1, s_1/x_2, s_2/x_3, s_3/x_3 \ldots$

Reconnaître deux mots avec Viterbi



n mots en compétition



Plan

Les unités acoustiques et leur représentation

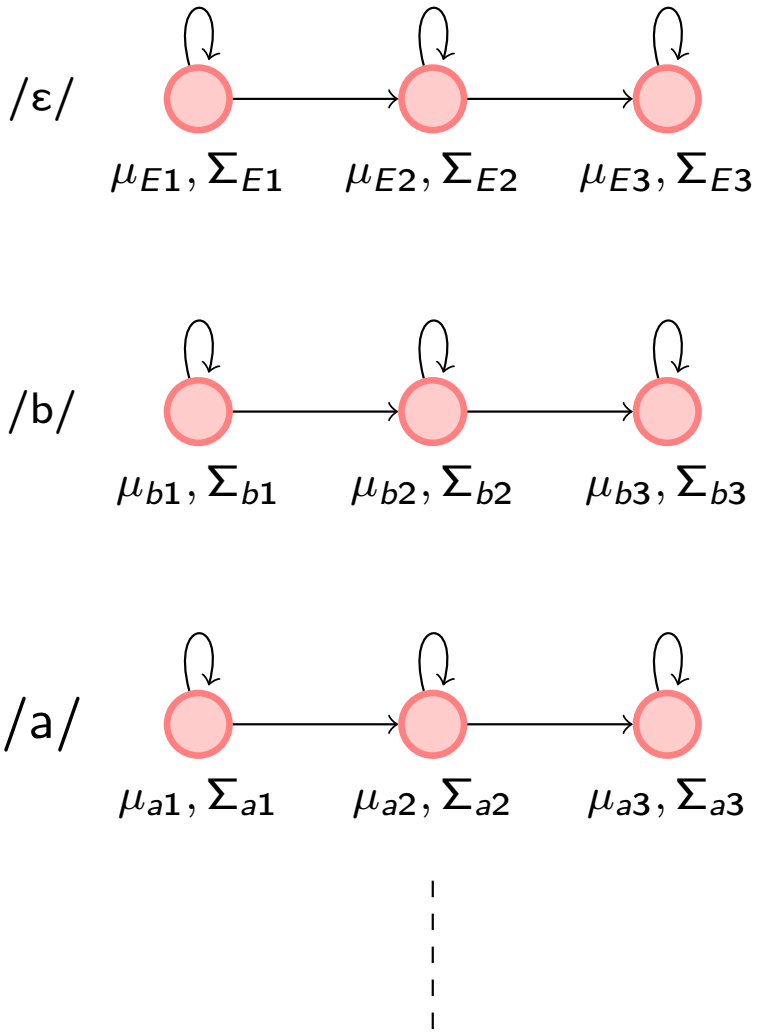
- $P(X|W)$ est calculée par un HMM.
  - un HMM pour chaque phrase ? est-ce utile ? faisable ?
  - un HMM pour chaque mot ?
    - demande de fixer la forme du modèle mot par mot
    - et des données d'apprentissage suffisantes...

  $\Rightarrow$ possible en petit vocabulaire (chiffres, commandes), irréaliste pour des vocabulaires plus grands
  - un HMM par syllabe ? par phonème ? par diphone ?
    - hypothèse simplificatrice : tous les modèles ont la même forme (gauche-droit, entre 3 à 5 états)
    - modélisation de la co-articulation

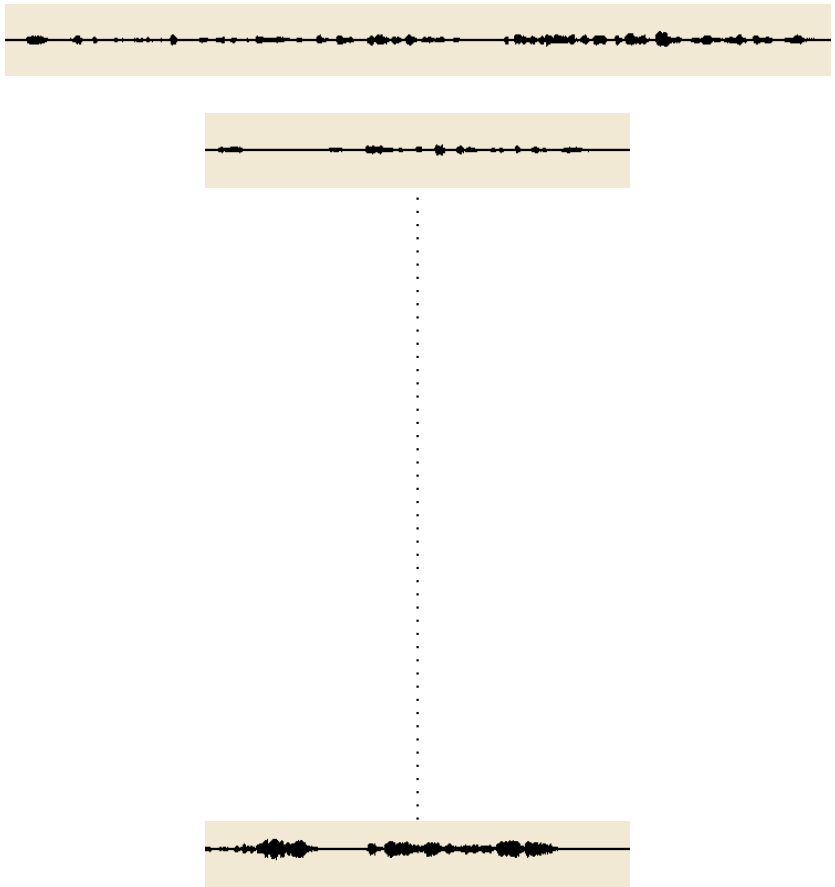
  $\Rightarrow$ un HMM par phone en contexte (/x-y+z/) : "triphones", "quintiphones".

Apprendre des modèles acoustiques

Estimer :



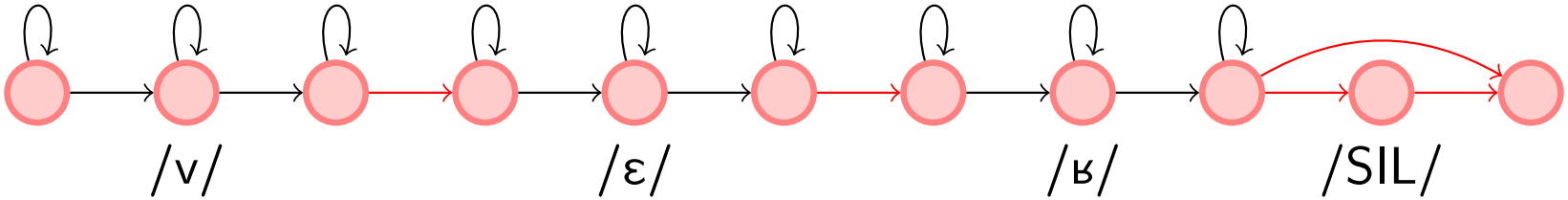
Avec des enregistrements :



Apprentissage acoustique : traitement d'une phrase

vers une reconnaissance sans erreur

vɛʁ yn ʁəkɔnɛsãs sãz ɛʁœʁ



Plan

Deux ressources indispensables

- des transcriptions « fidèles » des données d'apprentissage
  - représentant la variété des locuteurs (hommes/femmes, accents, style, etc)
  - des conditions d'enregistrement (micros, bruits,...)
  - et des conditions d'élocution
- des dictionnaires
  - contenant les variantes de prononciation...
  - leurs probabilités...
  - et leur contexte d'apparition

Produire des transcriptions

- segmenter la parole : indices acoustiques ou indices linguistiques
  - les “phrases” orales
  - une approche purement acoustique : HMM à deux états (silence/parole)
- réécrire la parole :
  - normaliser l'écriture (notamment les extra lexicaux) et la segmentation en mots
  - noter les "disfluences" : pauses remplies, hésitations, faux-départs, etc
  - noter/modéliser les autres évènements acoustiques (inspirations, bruits de bouches...) ?

Plan

Dictionnaires pour la reconnaissance

- Modélisation de la variabilité ?
  - degré de finesse, nombre de variantes
  - probabilisation des variantes
  - modélisation des effets de coarticulation
  - phénomènes phonologiques intra-mots (liaisons, élisions...) et variantes contextuelles
- Production des transcriptions :
  - méthodes à base de règles
  - méthodes à base de données

La règle de récriture contextuelle

GS → PC / CG _ RD

- ai → /ɛ/ / _ (*maison*, *saine*...)
- g → /ʒ/ / _ e (*mangeais*, *vengeance*...)
- a → /ɛ/ / _ i (*maison*, *saine*...)
- i → /ɛ/ / a _ (*maison*, *saine*...)

Il y a toujours plus d'une façon de faire les choses !

⇒ Penser à la productivité / pertinence linguistique ; maintenabilité (conflits/interactions entre règles)...

Gérer une collection de règles

Séquencement des règles

1. `aill` → `/aj/` / `trav _ V` (*travailleur*)
2. `aill` → `/ɑj/` / `_ V` (*caille*)
3. `ain` → `/ɛ̃/` / `_ C,#` (*vain*)
4. `ai` → `/ə/` / `f _ s[ae]` (*faisan*, *faiseur*)
5. `ai` → `/ɛ/` / `_` (*aigle*, *faisceau*)
6. `a` → `a` / `_` (*la*)

Ordre : du plus spécifique au plus général, la première règle applicable s'applique, **pas de retour arrière**.

Récriture complète d'une forme graphique

Forme graphique	Règles	Sortie
chasseur	ch → /ʃ/ / _	/ʃ/
chasseur	a → /a/ / _	/ʃa/
chasseur	ss → /s/ / _	/ʃas/
chasseur	eu → /œ/ / _ [rlpbvfiy]	/ʃasœ/
chasseur	r → /ʁ/ / _	/ʃasœʁ/

Classes, Contextes et méta-contextes

- `an` → `/ɑ̃/ / _ [bcdfgjklmpqrstvwxz#]` (*manger*, *danser*...)
- `an` → `/ɑ̃/ / _ C,#` (*manger*, *danser*...)
  où $C = [bcdfgjklmpqrstvwxz]$ (mais que fait *p* dans cette liste ?)
- `ill` → `/ij/ / C _` (*bille*, *fillette*...)
  où $C = [bcdf(gu)jklmp(qu)rstvwxyz]$ (en fait, les consonnes *phonétiques*)
- `e` → `/e/ / _ ((C|CD)V)+[o]#` (*banderillo*, *azulejo*...),
  où C représente les consonnes, CD les consonnes doubles, V les voyelles.
- `ea` → `/i/ / _ [`L_{eng}`]` (*stream*, *dealer*...)
  où L_{eng} liste les lexies d'origine anglaise.
- `gin` → `/ʤin/ / # _ #` (*gin*)

Utiliser un ensemble de règles

- parcours linéaire
- indexation
- organisation arborescente
- compilation de la machine à états finis correspondante
- gestion du non-déterminisme (variantes)

Quelques leçons

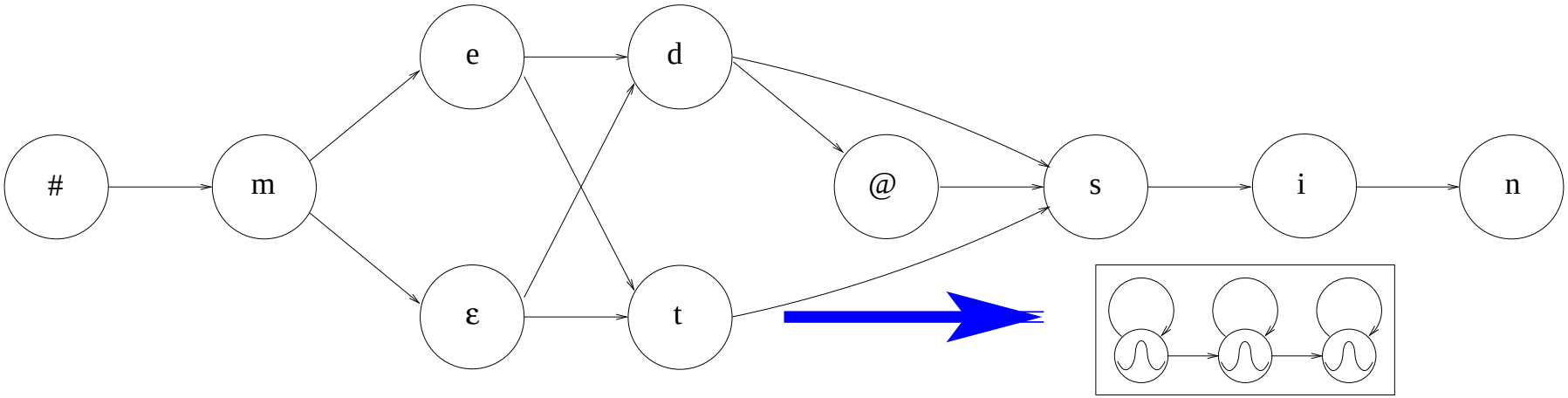
- un mot n'est pas qu'une suite de lettres : analyse de la *structure*
- les caractéristiques linguistiques (origine, catégorie morpho-syntaxique) des mots influent leur prononciation :
  $\Rightarrow$ typage des mots (étiquetage morpho-syntaxique)
- toutes les graphies ne sont pas des mots linguistiques :
  $\Rightarrow$ traitement des extra-lexicaux
- le contexte (phonologique, syntaxique, sémantique, pragmatique) influence la prononciation
- écrire/maintenir des règles est difficile : mélange des connaissances de niveaux distincts (morphologie, phonologie, syntaxe ?)

Des prononciation aux HMMs

Chaque mot du vocabulaire V est représenté par sa ou ses transcriptions phonétiques :

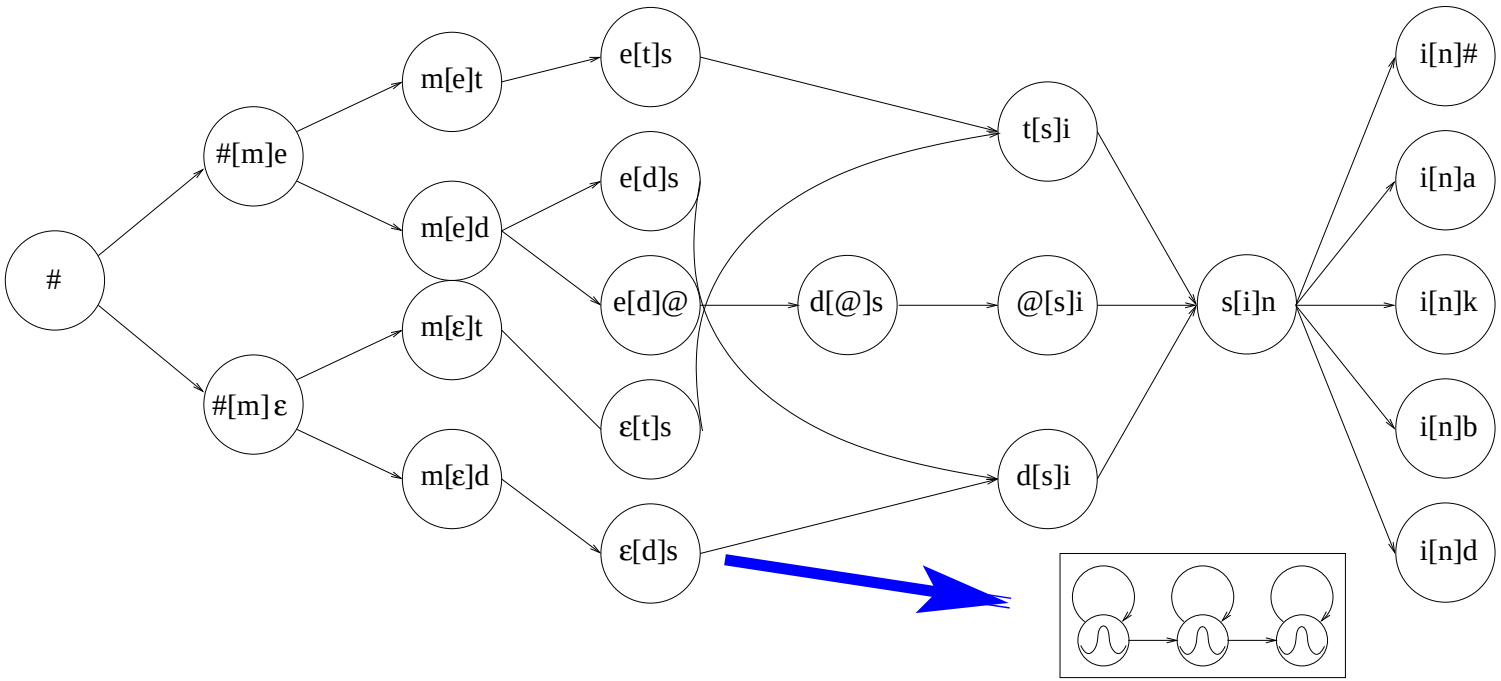
médecine $\rightarrow$ *medəsin, medsin, mɛdsin, metsin, mɛtsin...*

Chaque symbole phonétique correspond à un *HMM* $\Rightarrow$ construction d'un HMM de mot par *composition formelle* :



Avec des modèles contextuels (triphones)

Un modèle acoustique (HMM) différent pour *chaque phone en contexte* : triphone, pentaphone...
Triphone $x[y]z$: un modèle pour y précédé de x suivi de z. Pour *médecine* :



Plan

Apprentissage acoustique : vue d'ensemble (I)

1. Fixer un vocabulaire, un ensemble d'unités acoustique
2. Etant donné un corpus d'apprentissage (signal + transcriptions) ; pour chaque phrase w_1^n :
   - construire par concaténation des HMMs de mots un HMM pour w_1^n ;
   - attribution déterministe (Viterbi) ou probabiliste (forward/backward) des trames aux états ;
3. Mise à jour des paramètres θ (composantes, moyennes, variances) ; mise à jour des modèles ; retour en (2)

Apprentissage acoustique : vue d'ensemble (II)

input : $\mathcal{S} = \{\mathbf{x_1} : \mathbf{w_1} \ldots \mathbf{x}_n : w_n\} \in (\mathbb{R}^p)^*$ un ensemble d'échantillons transcrits

output : un ensemble de modèles

repeat

  foreach $x_1 ... x_T \in \mathcal{S}$ **do**

    $M \leftarrow$ `ConstruireHMM(W)` ;

    `ForwardBackward(`$M, x_1 ... x_T$`)` ;

    foreach $s, s' \in M, t = 1 ... T$ **do**

      $p(s,t) = P(q_t = s | x_1 ... x_T)$;

      $r(s,s',t) = P(q_t = s, q_{t+1} = s' | x_1 ... x_T)$;

      $\mu_{s,k} \leftarrow \mu_{s,k} + p(s,t) x_{t,k}$;

      $\Sigma_{s,k,k} \leftarrow \Sigma_{s,k,k} + p(s,t) x_{t,k}^2$;

      $a_{s,s'} \leftarrow a_{s,s'} + r(s,s',t)$;

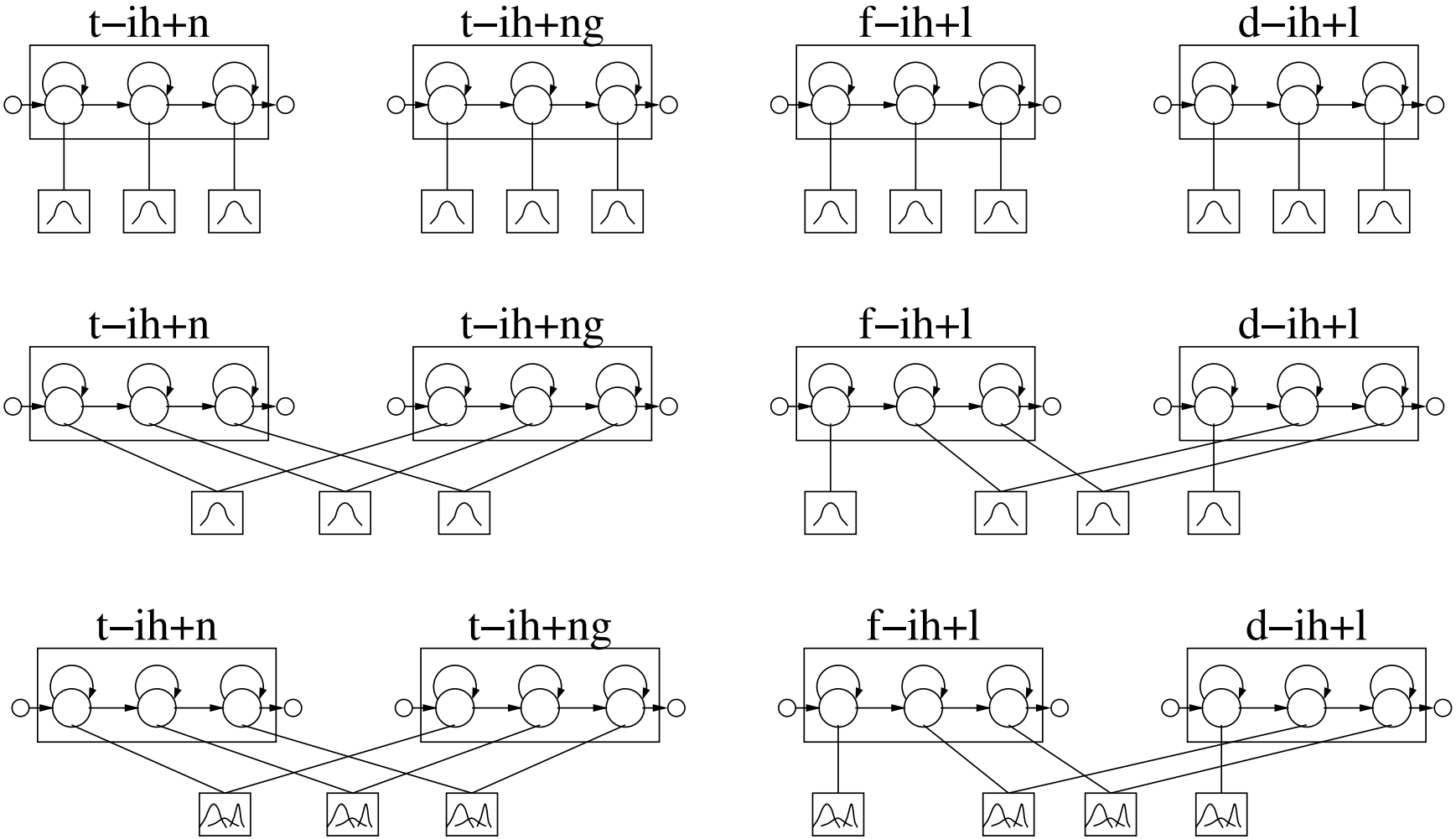
  $(\mu, \Sigma, a) \leftarrow$ `UpdateParameters()`

until *(Convergence des paramètres)*;

Quelques considérations pratiques

- 40 phones, 64000 (!) triphones, inégalement fréquents (eg. zzz)
- fusion (statistique ou linguistique) de contextes (eg. m[i]t, m[i]d, n[i]d...)
- 64000 (triphones) * 3 (états) * 16 (mixtures) * 2 (gaussiennes) * 39 (dimension) $\approx$ 240 millions (! !) de paramètres :
  $\Rightarrow$ clustering de modèles, partage de paramètres
- robustesse & adaptation : au sexe, aux conditions (bande large vs étroite ; bruit), au locuteur
- calcul des vraisemblances : évaluer 16 ou 32 gaussiennes par état 100 fois par seconde est coûteux $\Rightarrow$ techniques de calcul rapide

Partage d'états : *tying*



Plan

Adaptation : Motivations

On ne sait pas (encore) construire des modèles acoustiques universels

- Performances en reconnaissance mono-locuteur aujourd'hui très bonnes
  - Dégradation importante pour un autre locuteur, surtout du sexe opposé
    - les différentes réalisations d'un phonème sont trop dispersées dans l'espace acoustique et se superposent avec les réalisations d'autres phonèmes ;
  - Coût de la collecte des données pour chaque locuteur
- Reconnaissance de la parole continue indépendante du locuteur :
  - Grande base d'apprentissage ($>$ 100h) avec beaucoup de locuteurs et des conditions acoustiques différentes pour les modèles acoustiques
  - Bonnes performances mais inférieures à un système mono-locuteur.

$\Rightarrow$ Une adaptation du système multi-locuteurs au locuteur ou aux conditions acoustiques est nécessaire

Adaptation : Différentes Catégories I

Suivant la disponibilité des transcriptions :

- Supervisé : on connaît la transcription des données utilisées pour l'adaptation (coûteux)
  exemple : adaptation d'un système de dictée vocale *universelle* à un locuteur et aux conditions acoustiques particuliers.
- Non-supervisé : on ne connaît pas la transcription.
  exemple : transcriptions des émissions radio- ou télévisées. On sait détecter les changements de locuteur et/ou conditions d'environnement ;
- Faiblement supervisé : on connaît des transcriptions *approximatives* ;
  exemple : transcriptions des émissions radio- ou télévisées + *closed captions* / sous-titres

Adaptation : Différentes Catégories II

Valeurs adaptées :

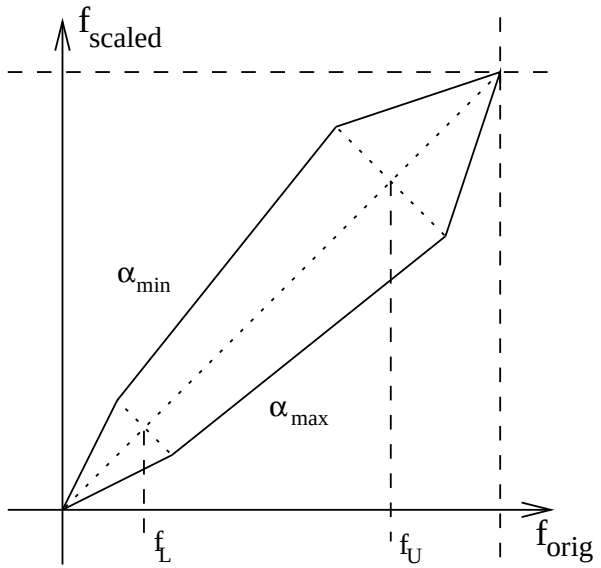
- Les modèles : on adapte les paramètres des modèles en utilisant une quantité limitée de données – deux approches : MAP et MLLR
- L'espace de représentation : on essaie de diminuer la variabilité inter-locuteurs en transformant les données acoustiques – approches : transformation linéaire (par morceau), réseaux de neurones

Quand l'adaptation est effectuée ?

- Statique : off-line, en mode batch, lorsque un bloc entier de données est disponible.
- Incrémentale : en ligne, l'adaptation progresse au fur et à mesure que les données d'adaptation deviennent disponibles, par exemple après chaque phrase.

Adaptation VTLN

Normalisation acoustique par la longueur du conduit vocal
(*Vocal Track Length Normalisation*, VTLN)
Principe : appliquer un coefficient de dilatation sur l'échelle des fréquences



Coefficient estimé par locuteur à l'aide d'un transcription approximative

Probabilité maximale a-posteriori (MAP) I

Principe :

- L'incertitude sur les paramètres des modèles Θ est reflétée par une densité de probabilité au lieu d'une valeur fixe ;
- Au début on définit une distribution de probabilité a priori $p(\Theta)$ qui est en général très large (on ne sait rien sur Θ) ;
- Après l'observation des données, on calcule la distribution de probabilité a posteriori $p(\Theta|\mathbf{O})$ à l'aide du théorème de Bayes.

Probabilité maximale a-posteriori (MAP) II

Application à l'adaptation :

- Règle de Bayes :

$$\begin{aligned}\hat{\Theta} &= \arg\max_{\Theta} P(\Theta|O) \\ &= \arg\max_{\Theta} \frac{P(O|\Theta)p(\Theta)}{p(O)} \\ &= \arg\max_{\Theta} P(O|\Theta)p(\Theta)\end{aligned}$$

  On retombe sur l'estimateur de vraisemblance maximale si $p(\Theta)$ a une distribution uniforme ;

- La suite de la théorie est plutôt compliquée ...
- En pratique MLLR est plus rapide, mais MAP possède de meilleures performances asymptotiques.

Probabilité maximale a-posteriori (MAP) III

Mise en œuvre : cas de l'adaptation de la moyenne d'une gaussienne

$$\hat{\mu}_i = \frac{\gamma_i \mu_i^{new} + \tau \mu_i^{prior}}{\gamma_i + \tau}$$

avec γ_i "masse"des trames sur la gaussienne
Nécessite donc un modèle de départ (prior) et les données d'adaptation segmentées
Nécessite une quantité importante de données, sinon peu d'effet (note : lorsque l'on utilise autant de données que pour générer les modèles, l'estimation est équivalente à l'estimation standard MLE)

Adaptation MAP en Pratique

- Utilisé pour adapter des modèles générique aux locuteurs masculins et féminins.
- Surtout important pour la reconnaissance d'émissions radio- ou télévisées puisqu'il y a peu de locuteurs féminins
- Meilleurs résultats et plus rapide par rapport à l'apprentissage séparé sur les données masculines et féminine

Adaptation MLLR : Introduction

MLLR = *Maximum Likelihood Linear Regression*
= régression linéaire de vraisemblance maximale

Idée :

- Déplacer les moyennes des mélanges de Gaussiennes des probabilités d'observation des HMMs de façon à ce que chaque état représente mieux les données d'adaptation
- Estimer un ensemble de transformations des vecteur de moyennes
- Remarque : on peut aussi transformer les matrices de covariance, mais pour des questions de faisabilité mathématique il faut se limiter aux matrices de covariance diagonale ;

Adaptation MLLR : Transformation des Moyennes

$\hat{\mu} = \mathbf{W}\mu + \mathbf{b}$ où

μ = vecteur de moyennes original
$\hat{\mu}$ = vecteur de moyennes transformé
$\mathbf{W}$ = matrice de transformation de dimension $d \times d$
$\mathbf{b}$ = vecteur de biais de dimension d

Notation plus compacte :

$\hat{\mu} = \tilde{\mathbf{W}}\tilde{\mu}$ où

$\tilde{\mathbf{W}} = [\mathbf{bW}]$
$\tilde{\mu} = [1\mu_1 \dots \mu_d]^t$

$\Rightarrow$ il faut donc estimer $d(d+1)$ paramètres pour chaque transformations

Adaptation MLLR : Transformation des Moyennes

Rappel : codage de la parole

$$\mu = \begin{bmatrix} \mu_s \\ \mu_\Delta \\ \mu_{\Delta^2} \end{bmatrix} \quad \text{où} \quad \begin{array}{rcl} \mu_s & = & \text{moyennes des 12 coefficients MFCC} \\ \mu_\Delta & = & \text{moyennes des 13 coefficients delta} \\ \mu_{\Delta^2} & = & \text{moyennes des 13 coefficients delta-delta} \end{array}$$

- On suppose qu'il n'y a pas de corrélation entre ces trois parties en ce qui concerne la transformation
  $\rightarrow$ utilisation d'une matrice bloc diagonale

$\Rightarrow$ le nombre de paramètres des transformations à estimer est réduit

$\Rightarrow$ moins de données d'apprentissage sont nécessaires

- exemple : $12^2 + 2 \times 13^2 = 482 \ll 38^2 = 1444$

Adaptation MLLR : Transformation des Matrices de Covariance

On se limite en général à des matrices de covariance diagonales :

$$\circ = \begin{bmatrix} \sigma_1 & & & 0 \\ 0 & \sigma_2 & & 0 \\ & & \ddots & \\ 0 & & & 0 \\ 0 & & & \sigma_d \end{bmatrix}$$

Transformation de la matrice de covariance :

$$\Rightarrow \hat{\sigma} = \mathbf{V}\sigma \qquad \text{où} \qquad \mathbf{V} = \begin{bmatrix} v_1 & & & 0 \\ 0 & v_2 & & 0 \\ & & \ddots & \\ 0 & & & 0 \\ 0 & & & v_d \end{bmatrix}$$

$\Rightarrow$ il faut donc estimer d paramètres pour chaque transformation

Adaptation MLLR contrainte

Même principe que MLLR mais la même matrice est utilisée pour transformer les moyennes et les variances

$$\begin{aligned} \hat{\mu} &= W\mu \\ \hat{\Sigma} &= W\Sigma W^T \end{aligned}$$

Équivalent à une transformation des données

$\hat{o} = Wo$ avec

$$\begin{aligned} o &= \text{une observation} \\ W &= \text{matrice de transformation de dimension } d \times d \\ \hat{o} &= \text{l'observation transformée} \end{aligned}$$

Classes de transformation : de 1 par état à 1 transformation globale
Permet l'apprentissage adaptatif (*Speaker Adaptatif Training*, SAT)

Adaptation MLLR : Arbre de Regression I

Le nombre de transformations dépend de la quantité de données :

- Peu de données : on utilise une transformation globale
  la même transformation est appliquée à toutes les Gaussiennes
- Plus de données sont disponibles :
  plus de transformations sont possibles donnant une meilleure adaptation.
  Chaque transformation est plus spécifique et elle n'est appliquée
  qu'à un regroupement de plusieurs Gaussiennes.
- Cas limite :
  on utilise une transformation séparée pour chaque Gaussienne
  - nécessite énormément de données
  - les états non-utilisés ne sont pas adaptés

  $\rightarrow$ le regroupement doit inclure tous les états
- Regroupements possibles : voyelles, nasales, plosives, ...

Adaptation MLLR : Arbre de Regression II

Meilleure approche :

- Le degré de partage des transformations dépend de la quantité de données
- Regroupement de toutes les distributions du modèle indépendant du locuteur $\rightarrow$ création d'un arbre de régression
- Lorsque les données d'adaptation sont disponibles, on parcourt l'arbre et on compte le nombre de distributions associées à chaque noeud terminal
- Si ce nombre dépasse un seuil une matrice de transformation est estimée pour toutes ces distributions
- Si ce nombre ne dépasse pas le seuil, les distributions sont regroupées un niveau plus haut dans l'arbre

Adaptation MLLR : Algorithme Supervisé

- Construire un système de reconnaissance indépendant du locuteur (en utilisant une grande base d'apprentissage avec une grande variété de locuteurs et de conditions acoustiques) ;
- Fixer le nombre de regroupements et construire l'arbre de régression
- Adapter le système à un nouveau locuteur :
  - obtenir un (court) signal du locuteur et sa transcription
  - faire une ou plusieurs itérations d'adaptation :
    - calculer les matrices de transformations en regroupant les états selon l'arbre de régression
    - calculer les nouveaux paramètres en appliquant les transformations
- On dispose maintenant d'un système adapté au nouveau locuteur

Adaptation MLLR : Algorithme Non-Supervisé

Le cas le plus souvent rencontré en pratique

Deux problèmes à résoudre :

1. Il faut déterminer les changements de locuteur et/ou des conditions d'environnement

→ Pendant la phase de segmentation on regroupe des parties de parole avec des propriétés acoustiques proches (p.ex. même locuteur et/ou même conditions d'environnement).

Pendant la reconnaissance les modèles sont adaptés séparément à chaque bloc

2. On ne connaît pas les transcriptions

→ Utiliser la phrase reconnue par le système pour adapter les modèles
En pratique les performances ne sont que légèrement inférieures à celles obtenues par une adaptation supervisée

Adaptation MLLR en Pratique

- Utilisé pour adapter des modèles génériques aux locuteurs et aux conditions acoustiques en utilisant la réponse actuel du système comme hypothèse
- Telephone Speech : adaptation au locuteur pour chaque conversation (il y a deux locuteurs). Gains de 3% absolue environ (13% relatif)
- Broadcast News : adaptation sur des regroupements qui sont déterminés automatiquement (locuteurs et/ou conditions acoustiques homogènes).

Plan

Systèmes neuronaux en reconnaissance de parole

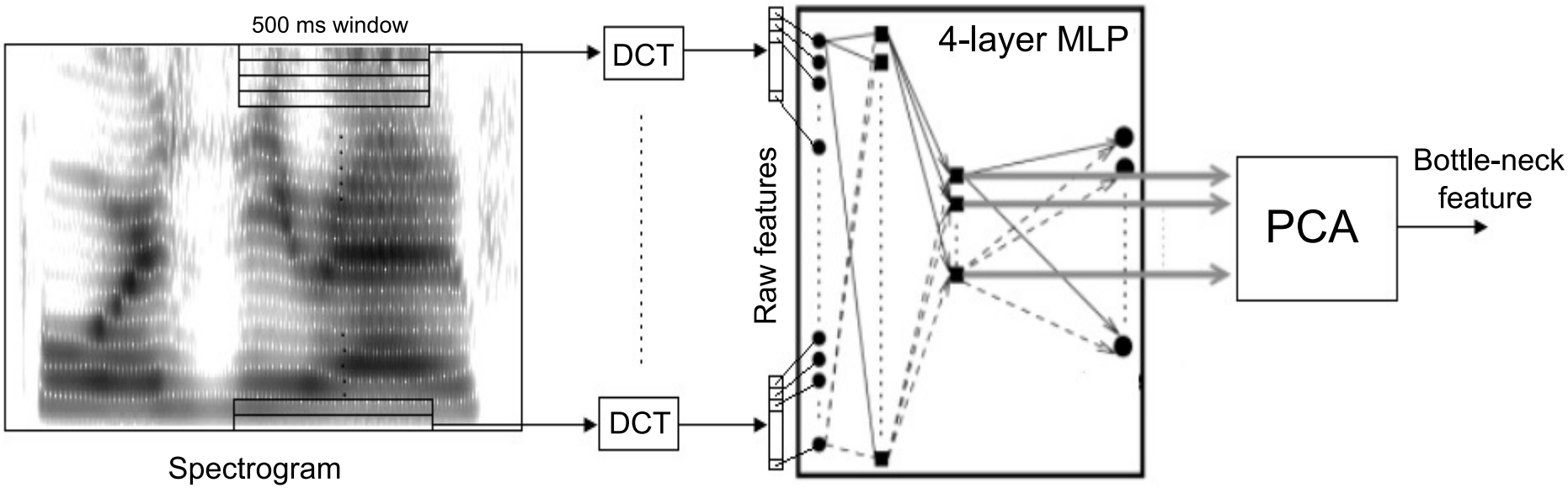
- nombreuses recherches publiées depuis 1990 avec des perceptrons multi-couches (MLP) et des réseaux récurrents.
  - Bourlard, Robinson, Bengio, Gallinari, Waibel...
- pendant une dizaine d'années, pas de gain significatif par rapport aux systèmes probabilistes "classiques" (GMM/HMM)
- intégration aux systèmes "état de l'art" pour les modèles linguistiques (depuis 2002) et acoustiques (depuis 2006)
- le décodage reste effectué par programmation dynamique (Viterbi ou équivalent) - on assiste à une progression de la CTC (connectionist temporal classification) proposée par Graves en 2006.

Systèmes neuronaux en reconnaissance de parole

- Modèles linguistiques neuronaux
  - proposés par Y. Bengio en 2001
  - intégré dans les systèmes de reconnaissance du LIMSI depuis 2002 (en combinaison avec un modèle n-gram)
  - apport significatif (gain relatif de 5% en performance)
- Modèles acoustiques neuronaux
  - Systèmes hybrides MLP/HMM
  - les paramètres acoustiques sont traités par un MLP qui estime les probabilités a posteriori des phonèmes ou états phonétiques
  - la sortie du MLP remplace le GMM de l'état d'un HMM
  - une variante : des paramètres extraits du MLP précédent sont combinés aux paramètres acoustiques standards (MFCC)
  - le reste du système est généralement un HMM standard

Exemple de système acoustique hybride

- TRAP-DCT (Grezl & Fousek, 2008)
- Entrée : 19 bandes x 25 coefficients
- 3e couche (bottleneck) : 39 coefficients + PCA (décorrélation)
- 4e couche (sortie) : probabilités des états phonétiques (46 x 3)



Réseaux de neurones pour la parole

Montée en puissance des approches neuronales avec les réseaux profonds (DNN)

- réseaux convolutifs (CNN) appliqués à la sortie d'un banc de filtre
- réseaux récurrents (Bi-LSTM = bi-directionnal long-short-term memory networks) pour le traitement du flux audio
- application en transcription automatique (assistants vocaux Siri d'Apple, Google Now...), détection de mots-clefs, reconnaissance du locuteur (modèles discriminants), synthèse de parole
- disponibilité de librairies logicielles performantes spécialisées DNN (TensorFlow, Keras, PyTorch...) ou dédiées à la parole (Kaldi).

Deux exemples :

- `https://machinelearning.apple.com/2017/10/01/hey-siri.html`
- `https://deepmind.com/blog/wavenet-generative-model-raw-audio/`

Plan

Plan

Viterbi : un algorithme de recherche

- le graphe de recherche :
  - un nœud = un état d'un HMM à un instant donné (q, t)
  - chaque nœud possède un score et un père (sur le meilleur chemin)
  - (q', t') suit (q, t) ssi $a(q, q') \neq 0$ et $t' = t + 1$
- Viterbi = recherche en *largeur d'abord* :
  1. initialiser les nœuds actifs à $t = 0$
  2. répéter pour t=1...T :
     - pour tous les nœuds (q, t), calculer le score des successeurs $score(n) = max_{m \in pere(n)} score(m) * a(m, n) * b(n, x_t)$
     - incrémenter t

Décodage de parole, une formulation du problème

- Les données :
  - un ensemble de mots V et leur(s) prononciation(s)
  - un ensemble d'unités acoustiques et leurs HMMs
  - un modèle de langage sur les mots de V
- Décodage de x_1^T :
  - en théorie : $\text{argmax}_{w_1^n \in V^*} P(X_1^T | w_1^n) P(w_1^n)$
  - en pratique : $\text{argmin}_{w_1^n \in V^*} -\log(P(X_1^T | w_1^n)) - \eta \log(P(w_1^n)) - n\xi$
- Les paramètres du décodeur :
  - η compense la différence d'échelle entre les vraisemblances acoustiques et les probabilités linguistiques
  - ξ compense les différences de longueur entre phrases
  - η et ξ sont optimisés sur des données de développement

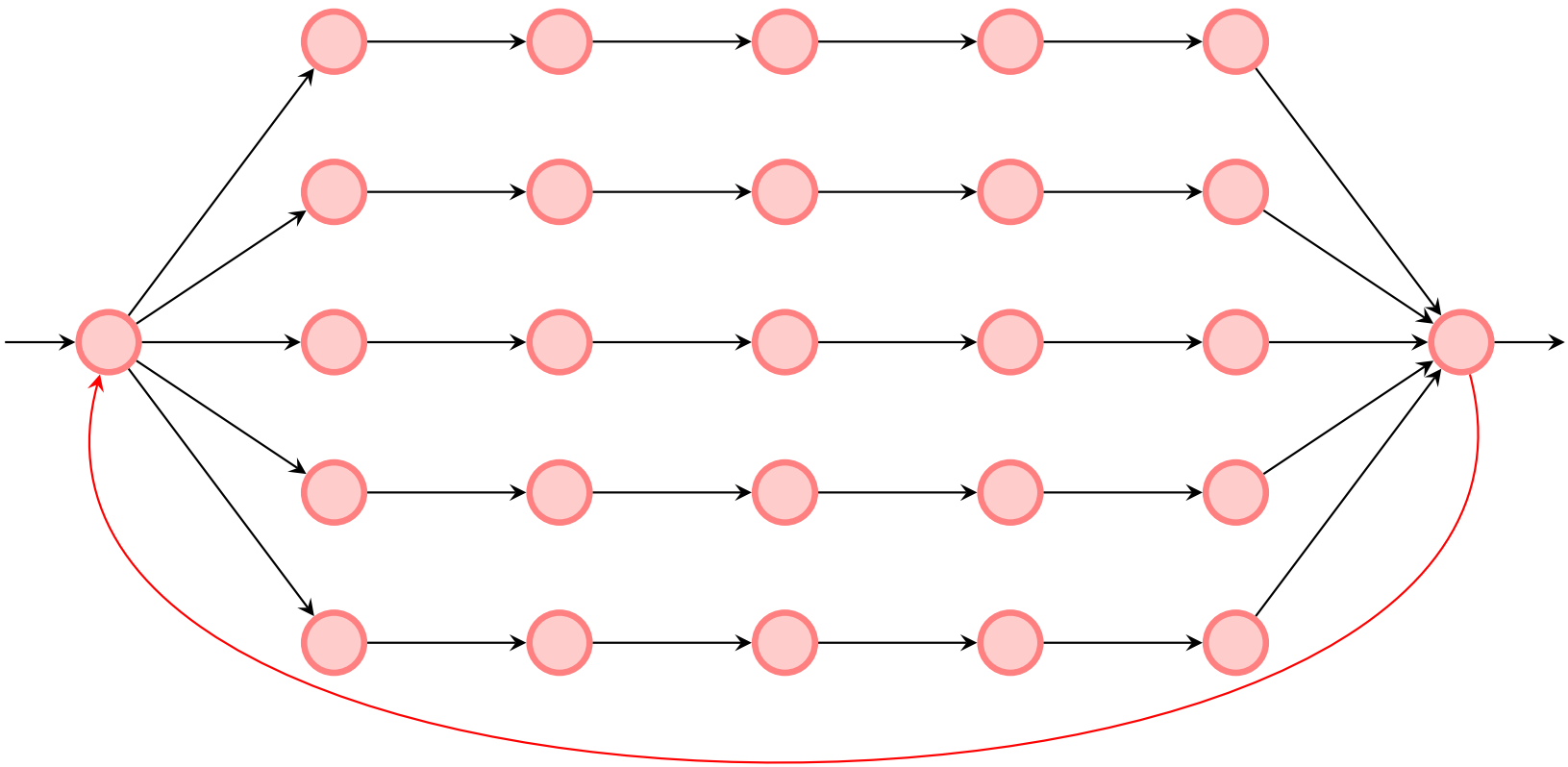
Décoder est un problème de recherche

Plan

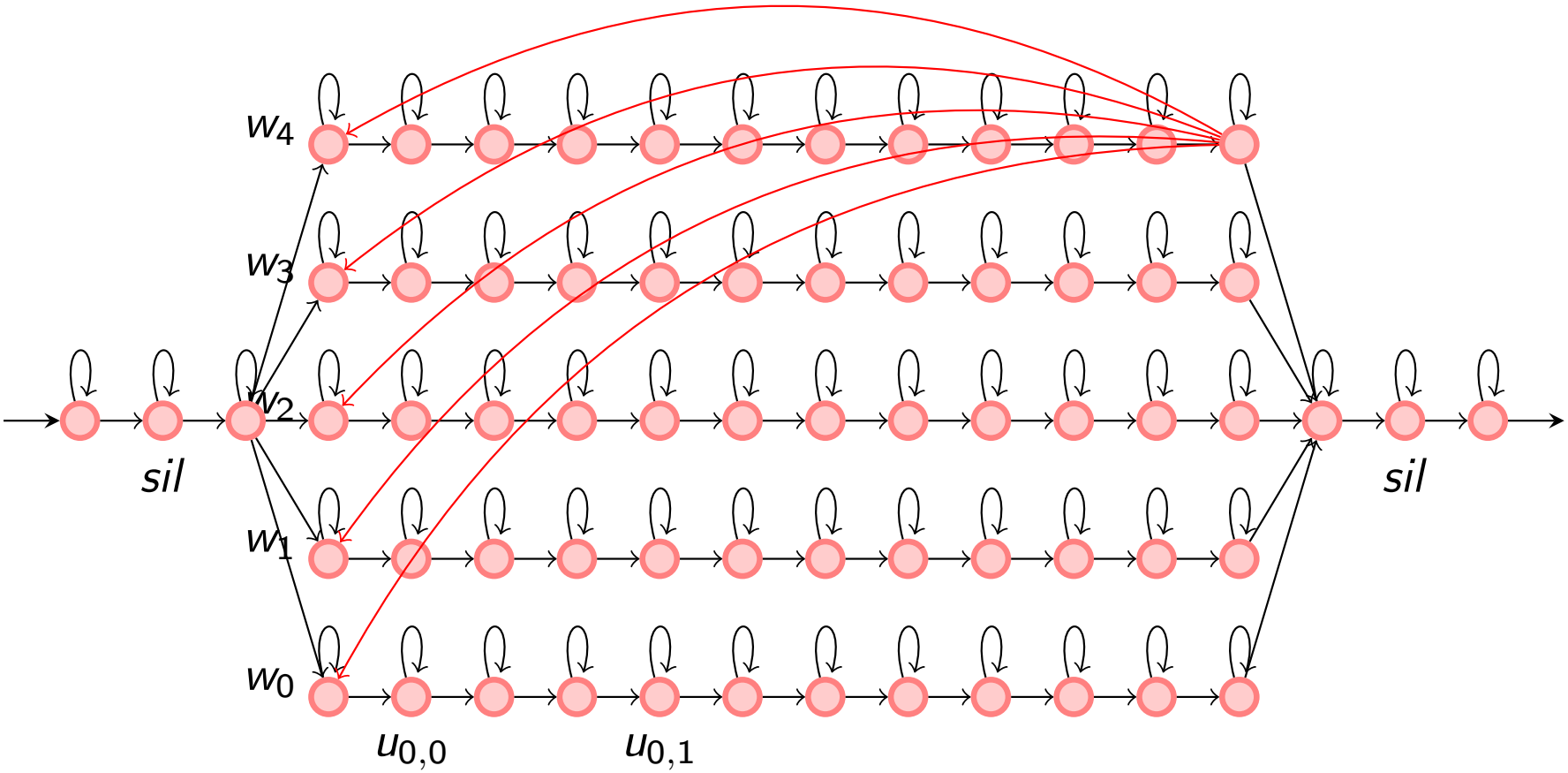
Plus court chemin dans un graphe d'états

- un ensemble d'états
- un état initial, un état final
- arcs pondérés $e \rightarrow e', c(e, e')$
- coût d'un chemin $s(e_0 ... e_i) = \sum_i c(e_i, e_{i+1})$
- pointeur arrière
- programmation dynamique : coûts positifs (Dijkstra), pas de cycle (Bellman)

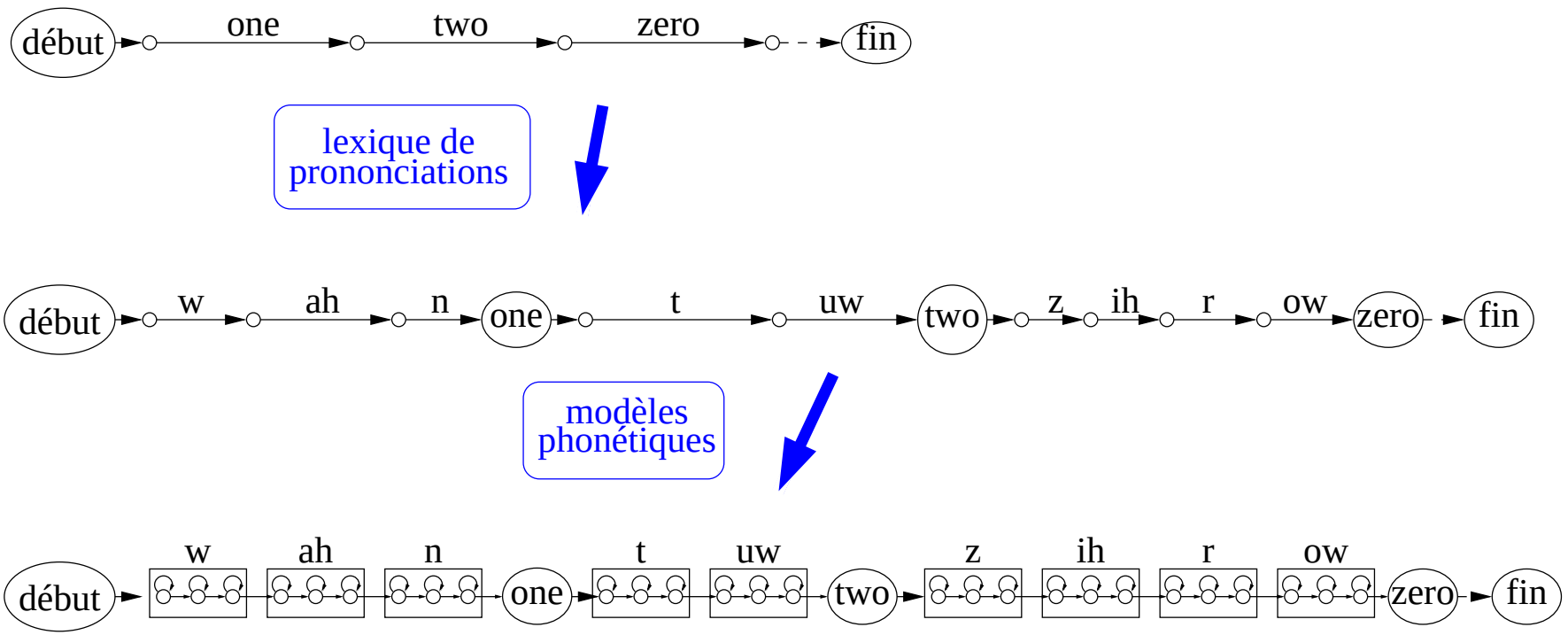
Les alternatives du décodeur



Le graphe du décodage



Le graphe de recherche : une composition formelle

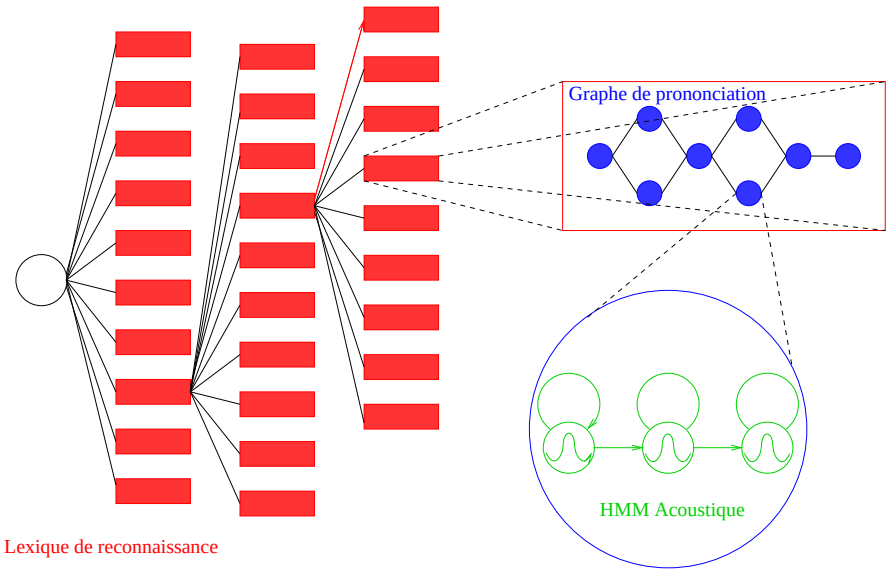


Le graphe de recherche de la reconnaissance vocale

- État : $e =$ [instant courant (t), mot courant (w_i), (tri)phone courant (=HMM) $(u_{i,p})$, état courant dans le HMM $(q_{i,p,k})$, meilleur état précédent (bp)]
- État initial : $e_i = [t = 0$, état initial d'un silence initial]
- État final : $e_f = [t = T$, état final d'un silence final]
- Successeurs de e :
  1. dans tous les cas : $\{[t+1, w_i, u_{i,p}, q_{i,p,k'}], \forall q_{p,k'} \in q_{p,k}.next()\}$
     $c(e, e') = -\log(a(q_{p,k}, q_{p,k'})) - \log(P(x_t|q_{p,'k}))$
  2. de plus, pour un état final de HMM interne :
     $\{[t+1, w_i, u_{i,p'}, q_{i,p',0}], \forall u_{i,p'} \in u_{i,p}.next()\}$,
     $c(e, e') = \log(P(x_t|q_{p',0}))$
  3. de plus, pour un état final de phone final : $\{[t+1, w_j, u_{j,p'}, q_{j,p',0}], \forall w_j \in V\}$,
     $c(e, e') = \log(P(w_j|w_i)) + \xi + \log(P(x_t|q_{p',0}))$

Espace de recherche : un bilan

- V^* l'ensemble des phrases sur V est un *graphe valué*, organisé *hiérarchiquement* (mots, phones, HMMs)



- Décodage = recherche du meilleur chemin dans V^*
- Une *hypothèse* est un chemin partiel, identifiée par l'état courant

Plan

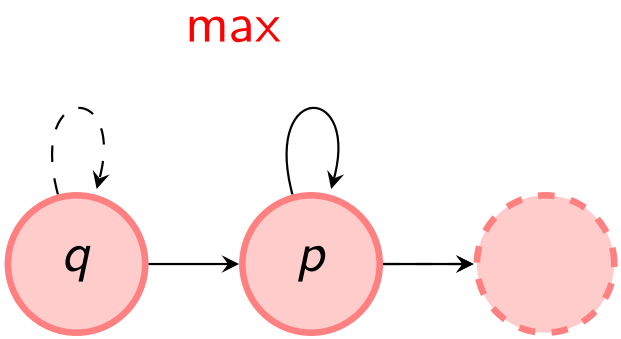
Algorithmes de recherche, quelques généralités

$S = \{[e_0, 0]\}$;
repeat /* Passe avant */
  $[e, s] \leftarrow \texttt{Pop}(S)$;
  foreach $(e' \in \texttt{Successor}(e))$ **do**
    $\texttt{Add}(S, [e', s + c(e, e')])$
until $(e = e_f)$;
$e = e_f$;
repeat /* Passe arrière */
  $e = e.bp$;
until $(e = e_0)$;

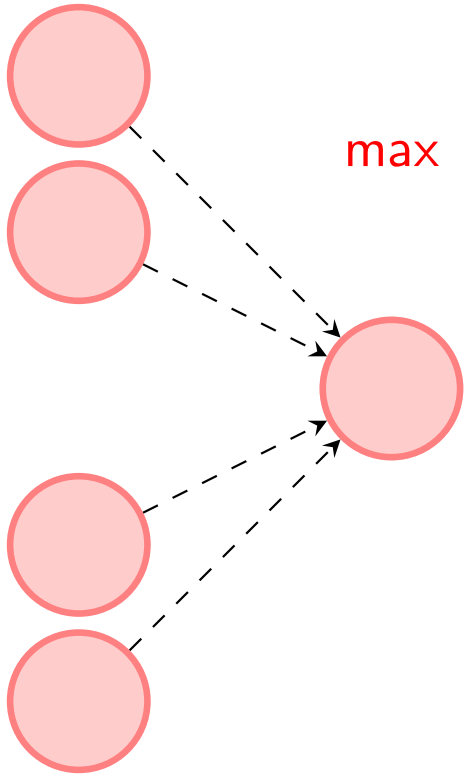
- S est une pile : *profondeur d'abord*
- S est une file : *largeur d'abord*
- S ordonné par score : *meilleur d'abord*

L'exhaustivité est impossible $\Rightarrow$ recherche heuristique

`Add`, quand e existe déjà : fusion d'hypothèses



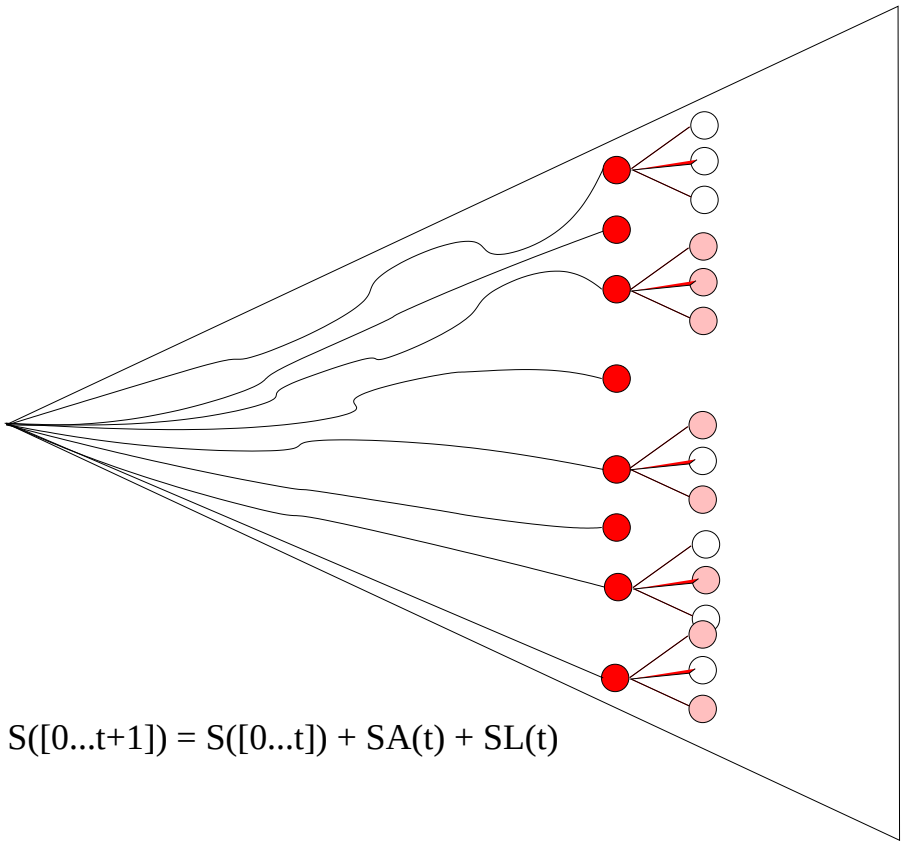
Dans un mot

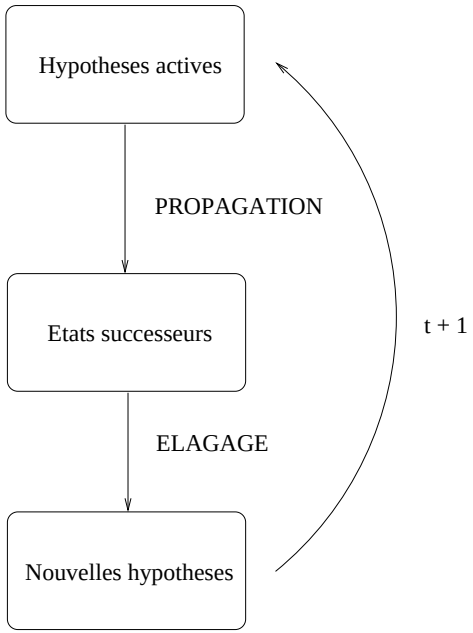


Entre deux mots

`Add(S,e)` ajoute ou *met à jour* (score, pointeurs arrière) l'hypothèse e.

Largeur d'abord + élagage : principe





Élaguer = compromis précision / vitesse :

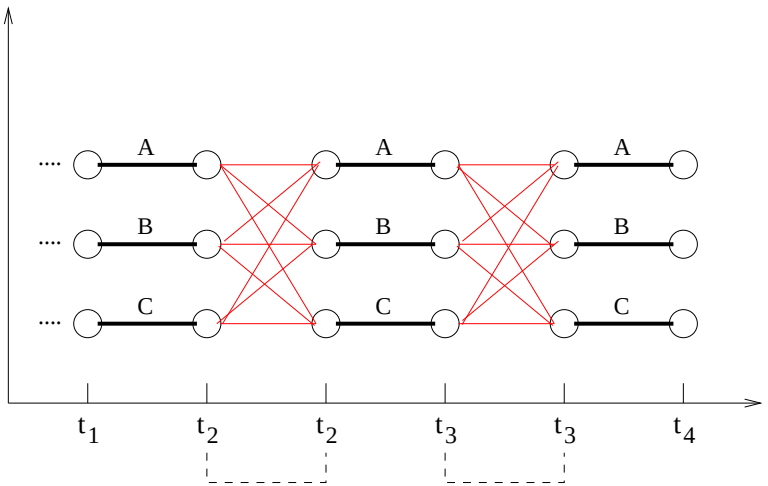
- éliminer les hypothèses << la meilleure (*beam-search*)
- limiter *a priori* le nombre maximum d'hypothèses actives

Largeur d'abord + élagage : formulation

$S_0 = \{e_0\}$;
foreach $t = 1 \ldots T$ **do** /* Passe avant */
  foreach $[e, s] \in S_{t-1}$ **do**
    foreach $(e' \in \texttt{Successor}(e))$ **do**
      $\texttt{Add}(S_t, [e', s + c(e, e')])$
  $S_t \leftarrow \texttt{Prune}(S_t)$;
$e = e_f$;
repeat /* Passe arrière */
  $e = e.bp$;
until $(e = e_0)$;

Largeur d'abord : spécifications

- Pour un modèle de langage bi-gramme, l'espace de recherche (avec un lexique "plat") est schématisé par ($|V| = 3$) :



- Programme :

$$
\begin{aligned}
w^* &= argmax_{w_1^n \in V^*} P(w_1^n) \sum_{\{q\}} P(q_1^T, x_1^T \mid w_1^n) \\
&\approx argmax_{w_1^n \in V^*} P(w_1^n) \max_{\{q\}} P(q_1^T, x_1^T \mid w_1^n)
\end{aligned}
$$

Largeur d'abord : quelques détails

- En transcription, seule importe la séquence de mots
  - distinguer états initiaux et non-initiaux des mots
  - les pointeurs arrière sont au niveau mot
- Équations de récurrence (bigramme) :
  - le score du meilleur chemin atteignant l'état initial de w :

$$s_w(w,t) = max_{\{v\}} \left\{ s_q(f(v),t) + \log P(w \mid v) \right\}, (f(v) = \text{ fin de } v)$$

    (+ pointeur arrière)
  - le score du meilleur chemin atteignant (q,w) à t :

$$\begin{aligned} s_q(t,i(w),w) &= s_w(w,t-1), (i(w) = \text{ début de } w) \\ s_q(t,q,w) &= max_{\{q'\}} \left\{ c(q',q) + s_q(t,q',w) \right\} \text{ sinon} \end{aligned}$$

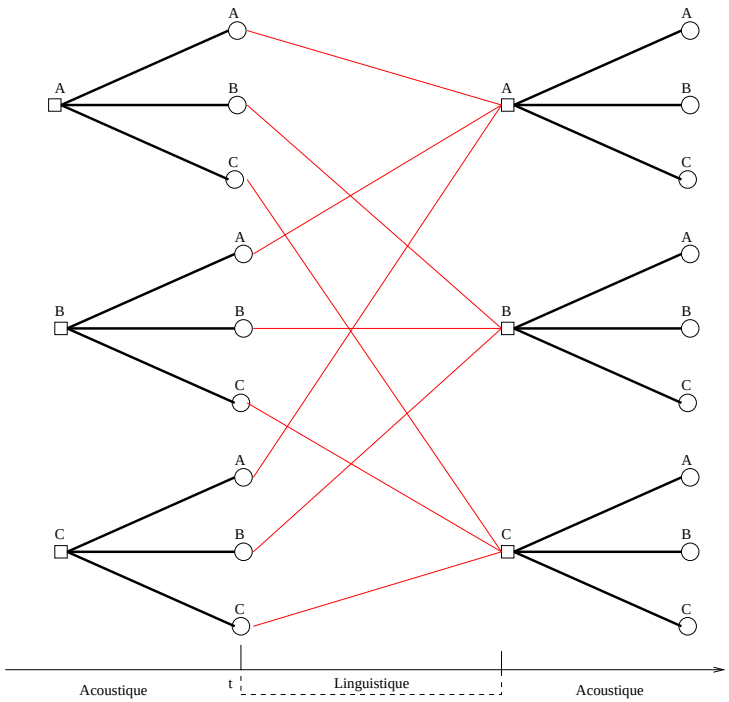
  - Début de w pour le meilleur chemin atteignant (q,w) à t :

$$\begin{aligned} B(t,0(w),w) &= t \\ B(t,q,w) &= B(t-1,\sigma_{max},w) \text{ pour } q \neq O(w) \end{aligned}$$

Largeur d'abord avec arbre des préfixes

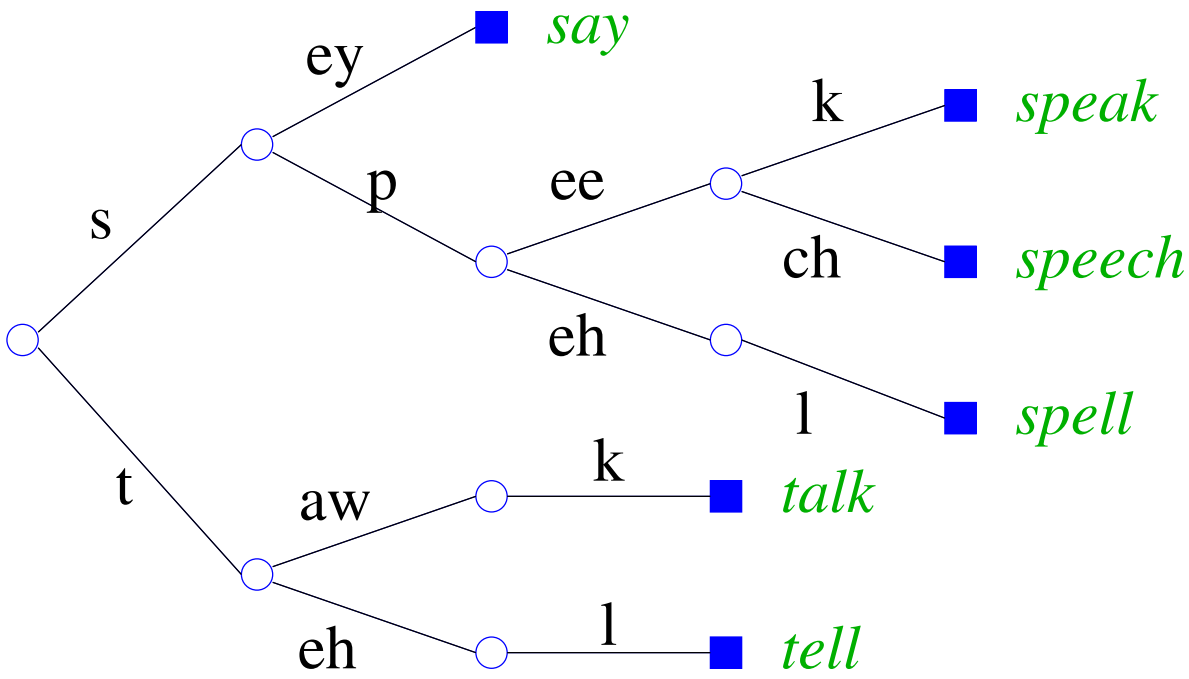
- La majorité des hypothèses actives sont en début de mot
- Factorisation des préfixes ⇒ *arbre lexical*

abre des préfixes

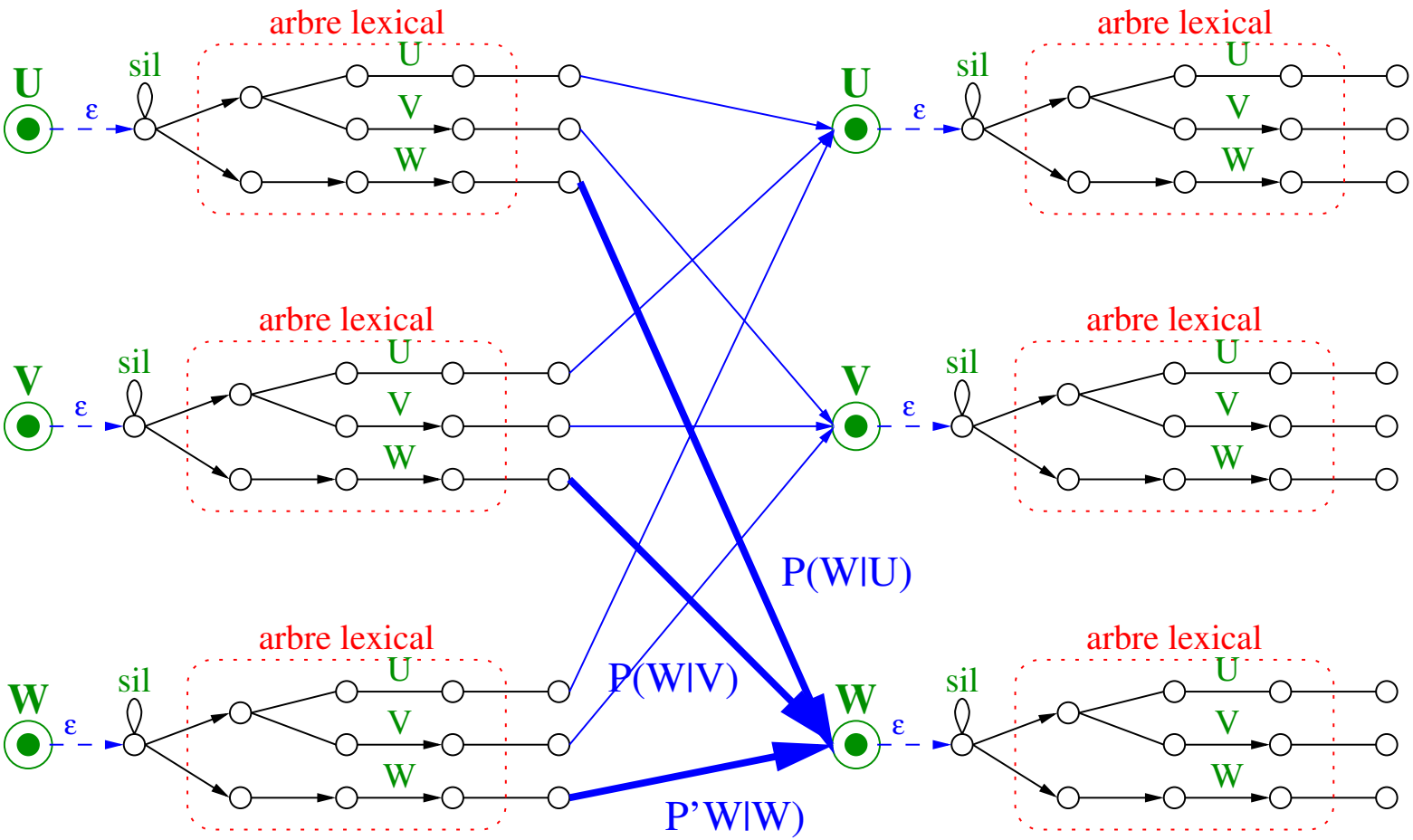


- + : compression du graphe ⇒ moins d'hypothèses
- - : retarde l'identification du mot courant, application du LM
- - : le contexte phonétique droit n'est plus connu

L'arbre des préfixes



Décodage avec arbre lexical : détail



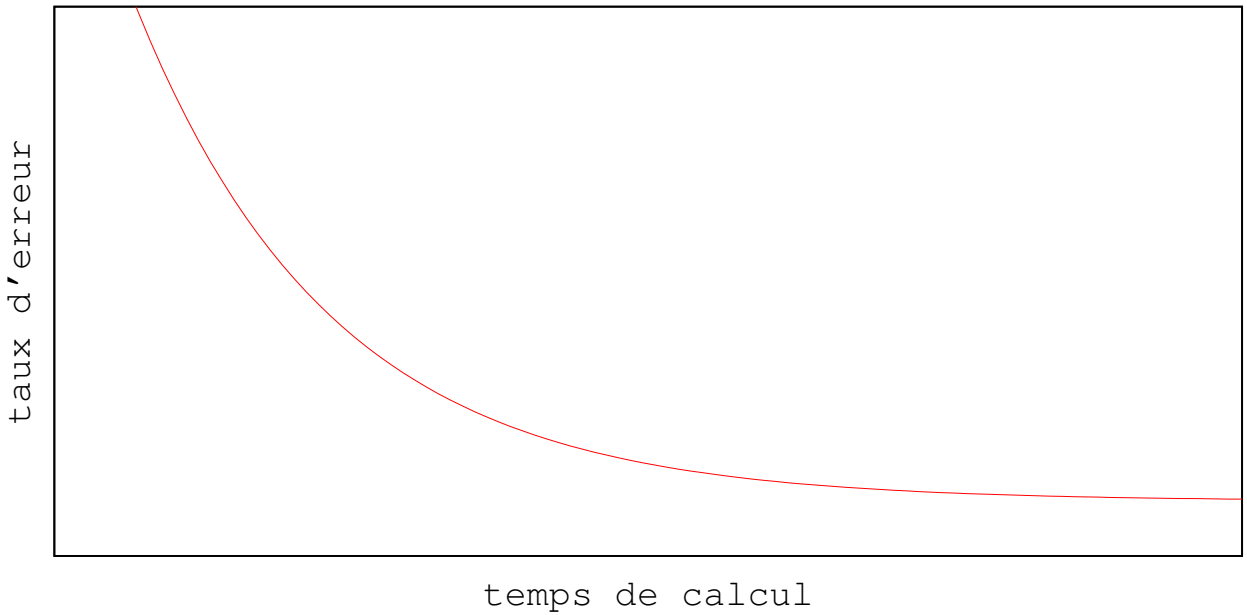
De l'élagage

- Élaguer = éviter l'examen de mauvaises solutions...
- Sans attendre de connaître toute la phrase
- Méthode heuristique $\Rightarrow$ erreurs de recherche
- Mise en œuvre :
  - conserver e ssi $s(best) - s(e) < \theta$ (*beam pruning*)
  - conserver un nombre fixe d'hypothèses (*histogram pruning*)
- Élaguer a un coût

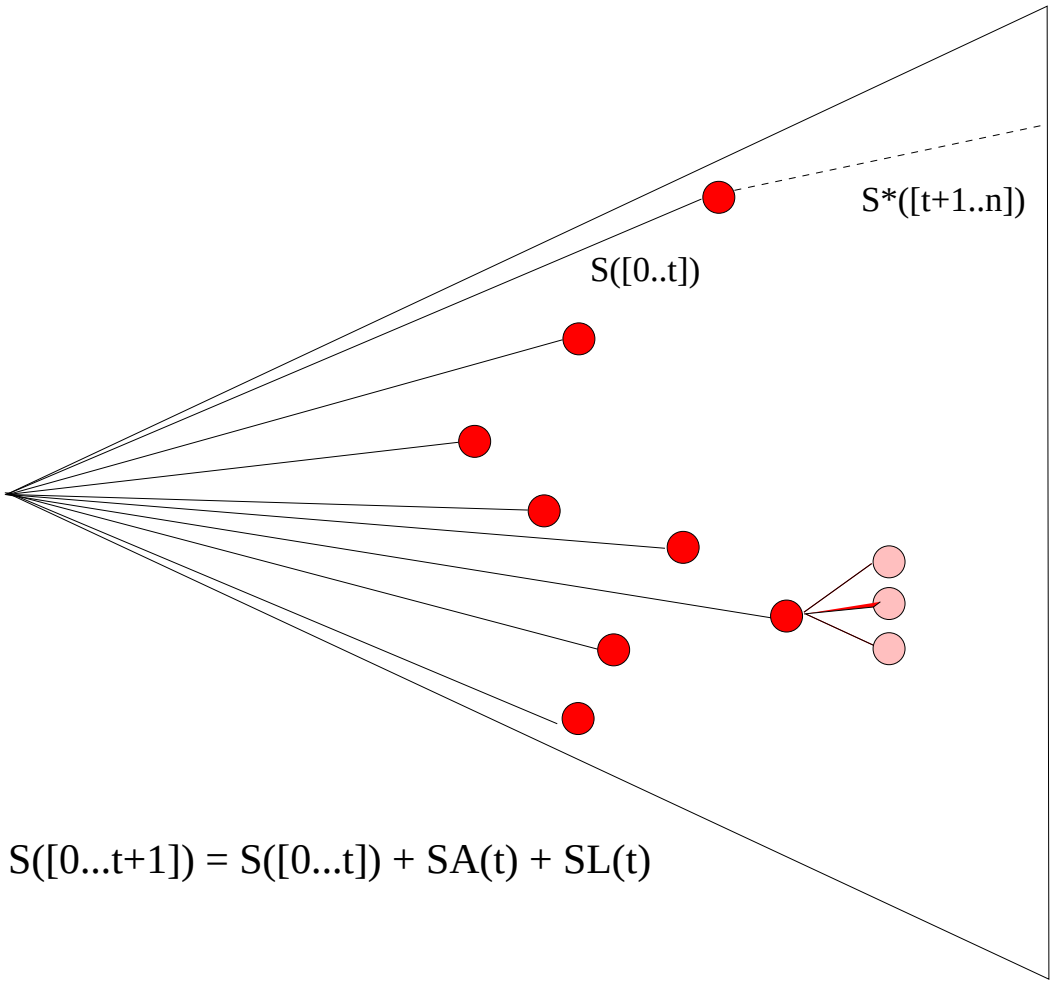
Les compromis de l'élagage

- θ ou n trop petit $\Rightarrow$ le meilleur chemin le plus probable risque d'être perdu avant la fin de l'observation.
- θ ou $n \Rightarrow$ saturation de la mémoire, augmentation du temps de calcul
- Élaguer est un compromis entre le temps de calcul et le taux d'erreur

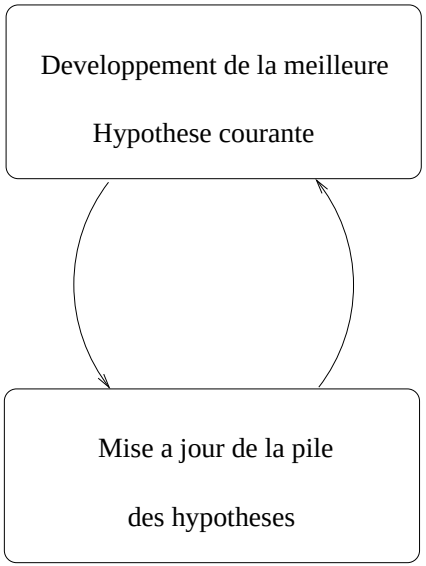
  Courbe typique :



Profondeur d'abord : principe



S([0...t+1]) = S([0...t]) + SA(t) + SL(t)



OPTIMALITE: DE LA RECHERCHE:

S* SURESTIME S

Stratégie best-first aka *stack decoding*

/* S est une liste de priorité */

$S = \{[e_0, 0]\}$;

repeat /* Passe avant */

  $[e, s] \leftarrow \texttt{Pop}(S)$;

  foreach $(e' \in \texttt{Successor}(e))$ **do**

    $\texttt{Add}(S, [e', s + c(e, e') + h(e')])$

until $(e = e_f \wedge \forall [e, s] \in S, s + h(e) < [e_f, s_f])$;

$e = e_f$;

repeat /* Passe arrière */

  $e = e.bp$;

until $(e = e_0)$;

- $h(e)$ estime le coût du meilleur chemin $\rightarrow e_f$
- + élagage des plus mauvais successeurs

Stack decoding : compléments

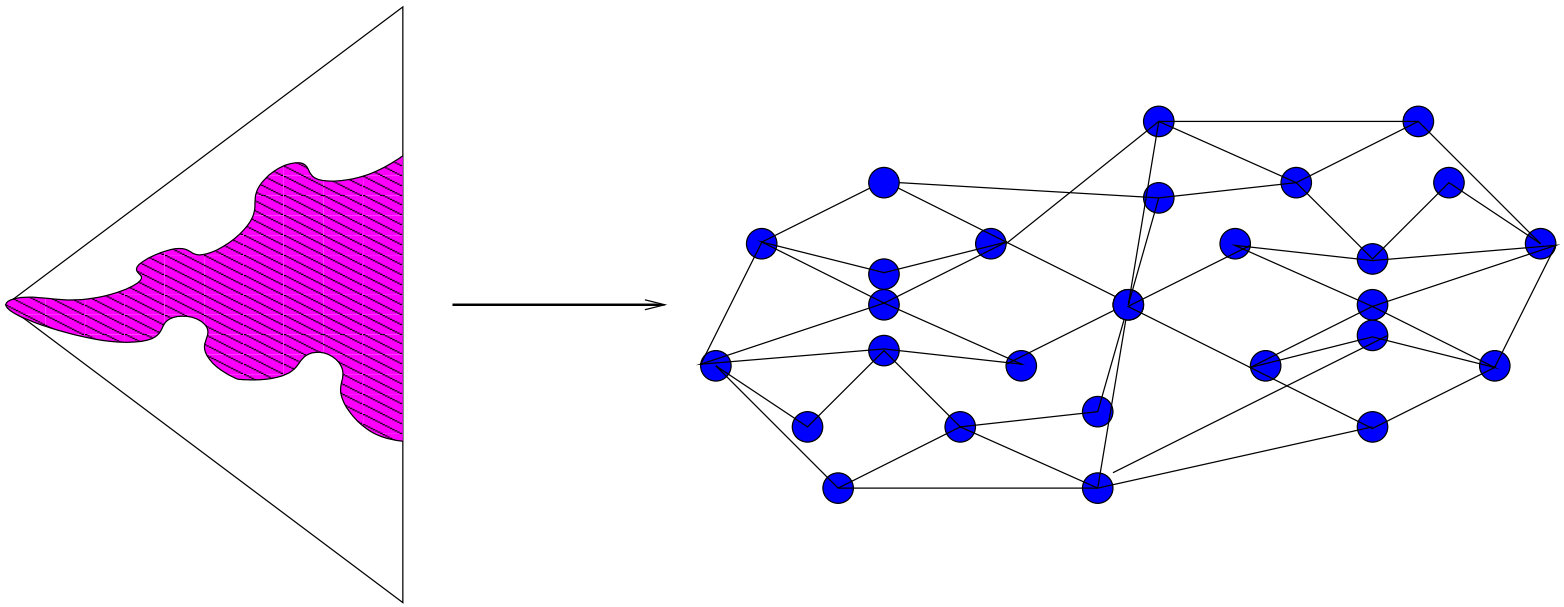
- Prop : si $h()$ est admissible ($h(e_f) = 0$, h surestime s), la recherche est sans erreur.
  - si arrêt en $[e', s']$ non optimal, $s' < s^*$, alors $\exists [e, s] \in S$ sur le chemin optimal s^*, donc $s + h(e) > s* > s'$: la recherche continue
- construction de bonnes heuristiques (regard avant, modèles simplifiés...)
- stratégies "multi-stack" (une pile par unité de temps)
- gestion pratique de l'asynchronisme

Compléments et finesses

- Les algorithmes précédents se généralisent :
  - à des modèles linguistiques d'ordre supérieur (tri-grammes...)
  - à la production de listes "*N-bests*" et de graphes de mots ($\Rightarrow$ stratégies *multi-passes*)
- Optimisation (*off-line*) du graphe $\Rightarrow$ réduction du non-déterminisme.
- Calcul rapide des scores acoustiques, *look-ahead* linguistique et/ou phonétique $\Rightarrow$ décodage (infra) temps réel

Décoder à petit pas

- L'espace de recherche initial est gigantesque
- Décoder par réduction successives de l'espace de recherche



Filtrage

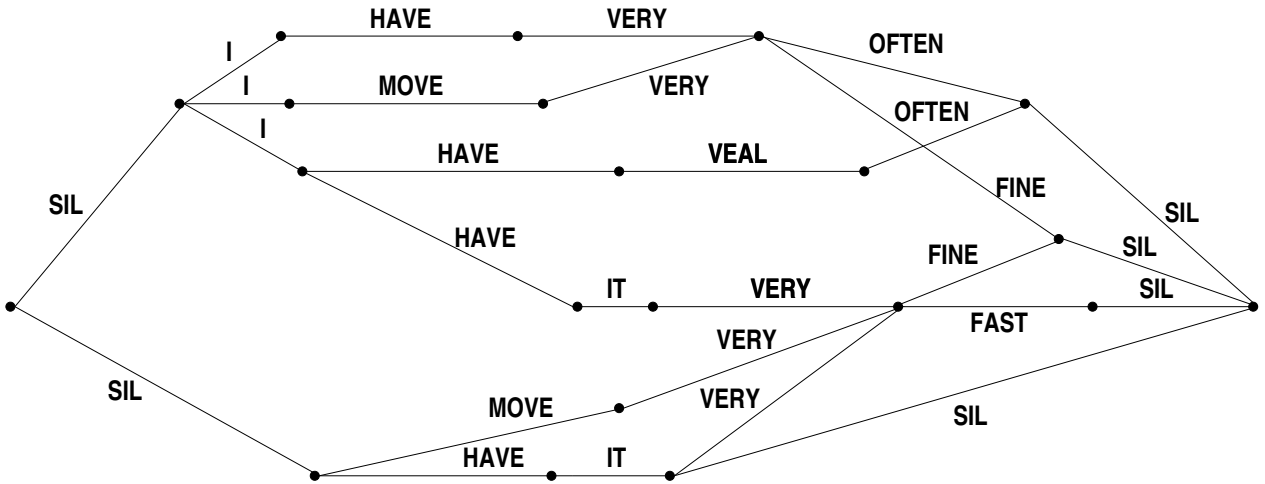
Reclassement

- Des bénéfices clairs :
  - mettre en jeu des modèles de complexité croissante
  - donner plus d'information (qui parle ? de quoi ?) au décodeur

Une interface simple : lister les n-meilleurs

- conserver les n-meilleurs solutions
- = conserver les n-meilleurs chemins (ou presque)
- = ? conserver n pointeurs arrières par hypothèse ?
- faible diversité des hypothèses, améliorations limitées

Une interface plus riche : les graphes de mots

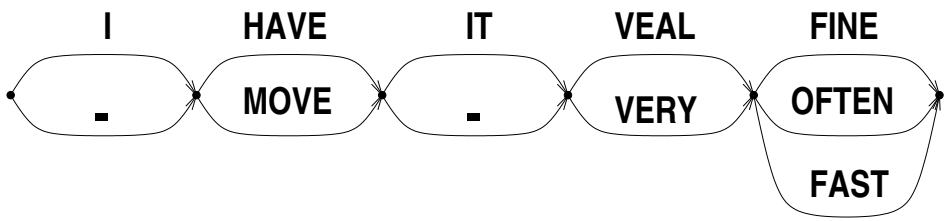


- formellement : automate acyclique valué = ensemble *fini* d'hypothèses G $\Rightarrow$ écrémage drastique de l'espace de recherche
- possibilité de ré-évaluation (acoustique et/ou linguistique) :
  - avec des modèles lus riches
  - avec des modèles plus adaptés
- une ré-évalution est une nouvelle recherche : $\text{argmax}_{w_1^n \in G}$

Construire des treillis : problèmes pratiques

- modification de la recherche : en début de mot conserver plusieurs pointeurs arrière
- les treillis réels contiennent des millions d'hypothèses, certaines très ressemblantes, certaines très improbables
  $\Rightarrow$ élagage du treillis
- la première passe détermine la "richesse" du treillis : retarder les décisions difficiles
  utiliser des modèles frustres

Une interface effective : les réseaux de consensus



- formellement : automate valué, localisation des ambiguïtés
- simplification heuristique du graphe de mots
- possibilité de ré-évaluation
- interface pour les traitements ultérieurs (par ex. indexation, traduction, etc)