Partie I. Méthodes symboliques
Alexandre DIKOVSKY

Cours 1. Méthodes Harrissiennes. Alignement.

1 Apprentissage symbolique

Les méthodes d'acquisition automatique des informations à partir de textes datent des années 50 et proviennent des méthodes générales de l'*apprentissage artificiel* (Machine learning) [T. Mitchell. Machine learning. McGraw-Hill, 1997]. Nous nous intéressons dans ce chapitre à l'*apprentissage symbolique supervisé*. D'une manière générale, il s'agit des méthodes A qui, à partir d'un certain nombre d'informations I (*description extensionnelle*) positives ou négatives par rapport à l'ensemble cible c d'informations, acquièrent une description h (*description intensionnelle*) finie et complète de c. A peut utiliser des connaissances sur le domaine $\mathcal{H}$ des descriptions h. Cet espace $\mathcal{H}$ est une classe de descriptions hypothétiques structurées (e.g., des automates abstraits, des grammaires, des formules, etc.) sélectionnée a priori.

Les méthodes qui nous intéressent dans un premier temps sont *inductives*, c'est-à-dire, celles qui élaborent les descriptions hypothétiques plus fines à partir d'ensembles croissants d'exemples. Voici le scénario classique de l'apprentissage par induction qui est une version précisée du csénario présenté dans [A. Cornuéjols, L. Miclet, Apprentissage artificiel ; concepts et algorithmes. 2002, Eyrolles]) :
(i) L'objectif de l'apprentissage est d'acquierir à partir d'un certain nombre d'exemples une *fonction cible* f qui appartient à un espace de fonctions $\mathcal{F}$.
(ii) Les échantillons des exemples $E = \{e_1, \ldots, e_n\} \in \chi$ sont tirés aléatoirement de l'environnement χ selon une distribution fixe $\mathcal{D}_\chi$ sur χ. Un "oracle" relie ces exemples E avec les valeurs de f par une relation finie R_E : $e_i R_E f(e_i), 1 \leq i \leq n$ (par exemple, $f(e_i) = u_i$ ou $f(e_j) \neq u_j$).
(iii) L'apprenant A dispose d'une connaissance initiale α (le biais) sur l'espace $\mathcal{H}$ des hypothèses. Avec cette connaissance, pour tout échantillon E, A calcule son approximation hypothétique $h = A(R_E, \alpha) \in \mathcal{H}$ de f.
(iv) Les hipothèses calculées par A convergent à la limite vers une description $c \in \mathcal{H}$ de f.
(v) Tout modèle formel particulier d'apprentissage inclut un *protocole* suivi lors de l'apprentissage, par exemple :
- comment sont fournis les exemples,
- quelle est leur nature : positive, négative ou les deux,
- teste-t-on les hypothèses et à quel moment,

quel est le *critère de stabilisation de la suite des hypothèses* qui défini le moment d'obtention de la solution).

Les méthodes priviligées dans ce chapitre sont *symboliques* au sens où :
1. la cible d'apprentissage f appartient à l'espace des hypothèses $\mathcal{H}$,
2. l'algorithme A et le critère de stabilisation sont établis sur des connaissances structurelles sur l'espace $\mathcal{H}$,
3. le critère de stabilisation définit uniquement la fonction vers laquelle converge la suite des hypothèses.
4. A ne dépend pas de la distribution $\mathcal{D}_\chi$ sur χ.

2 Contexte et Distribution

2.1 Déchiffrement des ressources linguistiques

L'idée de déchiffrement des ressources linguistiques remonte à L. Hjelmslev [Principes de grammaire générale. 1928, Kgl. danske Videnskabernes Selskab] et Z.S. Harris [Structural Linguistics. 1951, Chicago Univ. Press]. Points essentiels :
1. Toutes les ressources linguistiques (phonologiques, morphologiques, lexicales, syntaxiques, sémantiques, pragmatiques) sont complètement déterminées par l'ensemble des contextes où elles se trouvent (ainsi, toutes les deux ressources qui ont les mêmes contextes sont interchangeables).
2. On peut classer les ressources par les oppositions entre ses contextes.
3. Le principe algorithmique sous-jacent aux méthodes distributionnelles consiste à :
   - définir l'espace de recherche $\mathcal{H}$ (avec les hypothèses pondérées),
   - choisir une fonction heuristique d'estimation des hypothèses,
   - définir les solutions comme les hypothèses optimales.

2.2 EXEMPLES

Exemple 1. Différenciation des voyelles et des consonnes
Algorithme de B.V. Sukhotin *[Algorithmes de déchiffrement linguistique, Probl. de la linguistique structurelle, 1963 (russ.)]*
Entrée : *Un texte T non segmenté, non analysé sur un alphabet A donné de phonèmes.*
Objectif : *La décomposition $A = V \cup C$ inconnue en voyelles et consonnes. $V \cap C = \emptyset$.*
Espace de recherche : $\{(V_h, C_h) \mid A = V_h \cup C_h \ \wedge \ V_h \cap C_h = \emptyset\}$.
Fonction d'estimation : *Pour une hypothèse $A = V_h \cup C_h$,*

$$\Phi((V_h, C_h)) =_{df} \mathcal{P}(occur(vv', T) \ \vee \ occur(cc', T) \mid v, v' \in V_h, c, c' \in C_h),$$

où $\mathcal{P}(E)$ est la fréquence de E dans l'espace de recherche et

$$occur(m, T) =_{df} \exists t_1, t_2 \ (t_1 m t_2 \in T).$$

Heuristique : *La solution acceptable minimise $\Phi((V_h, C_h))$: les combinaisons vc, cv sont plus probables que les combinaisons vv', cc'.*
Explication linguistique : *Les voyelles se trouvent rarement auprès des voyelles et les consonnes se trouvent rarement auprès des consonnes, tandis que les voyelles et les consonnes sont fréquemment voisines.*

Algorithme : appliqué aux matrices $M = A \times A$

$V := \emptyset; C := Lignes$
1. **POUR TOUTE** ligne $i \in C$ dans M
   FAIRE
   $\sigma_i := \sum_j a_{ij}$
FIN_PT
2. **TANT QUE** $\exists i_0\ \sigma_{i_0} > 0$
FAIRE
   TROUVER $i_{max} := \{i_m \in C \mid \sigma_{i_m} = \max\{\sigma_i > 0 \mid i \in C\}\}$;
   $V := V \cup \{i_{max}\}; C := C - \{i_{max}\}$;
   POUR TOUTE ligne $i \in C$
      FAIRE
      $\sigma_i := \sigma_i - 2 * a_{i(i_{max})}$
   FIN_PT;
   TRIER V, C (V en avant)
FIN_TQ
3. **RENDRE** V, C

	a_1	a_2	a_3	a_4	a_5	
a_1	1	1	4	3	2	11
a_2	2	1	3	4	2	12
a_3	3	4	1	2	3	13
a_4	4	3	2	1	4	14 (max)
a_5	1	1	4	3	1	10

	a_4	a_1	a_2	a_3	a_5	
a_4	1	4	3	2	4	(v)
a_1	3	1	1	4	2	11-2*3=5
a_2	4	2	1	3	2	12-2*4=4
a_3	2	3	4	1	3	13-2*2=9 (max)
a_5	3	1	1	4	1	10-2*3=4

	a_4	a_3	a_1	a_2	a_5	
a_4	1	2	4	3	4	(v)
a_3	2	1	3	4	3	(v)
a_1	3	4	1	1	2	5-2*4=-3 (c)
a_2	4	3	2	1	2	4-2*3=-2 (c)
a_5	3	4	1	1	1	4-2*4=-4 (c)

Cet algorithme ne calcule pas toujours le maximum de la fonction d'estimation mais il donne des résultats très convaincants : 1 erreur pour le russe, 1 erreur pour le français, 3 erreurs pour l'allemand (l'erreur typique : *h* est classé comme une voyelle ; cf. les combinaisons 'ch').

Exemple 2. Extraction de morphes
Algorithme de Z. Harris *[From Phoneme to Morpheme linguistique, Language, v. 31(2), 1955]*
Entrée : *Un corpus C non segmenté, non analysé, présenté en forme phonétique sur un inventaire P de phonèmes. Par exemple, [hiysklever] correspond à* ≪He is clever≫ *en anglais.*
Objectif : *Les positions des frontières entre les morphes dans le corpus.*
Espace de recherche : *Positions entre les phonèmes dans le corpus.*
Fonction d'estimation (k-gramme) : *Pour une position hypothétique α dans le corpus C*

$$\Phi^k(\alpha) =_{df} - \sum_{occur(\alpha_{-k}...\alpha_{-1},C),\alpha\in P} \mathcal{P}(\alpha|\alpha_{-k}...\alpha_{-1}) * \log \mathcal{P}(\alpha|\alpha_{-k}...\alpha_{-1})$$

Heuristique : *La solution acceptable maximise $\Phi^k(\alpha)$.*
Explication linguistique : *Les frontières entre les morphes sont les positions où la lettre suivante est le moins prédictible (c'est-à-dire, l'entropie est maximum).*

Algorithme :

seuil := H_{max};
h_{-1} := *seuil*;
POUR TOUTE position $i \in C$ (incrémentalement)
FAIRE
SI $\Phi^k(\alpha_i) > h_{-1} \ \wedge \ \Phi^k(\alpha_i) \geq seuil$
ALORS $RENDRE(i)$
FIN_SI;
$h_{-1} := \Phi^k(\alpha_i)$
FIN_PT

h	i	y	z	k	l	e	v	e	r	
9	14	29	29	11	7	..	..	..	30	nombre des continuations
-	+	> seuil	> seuil	-	-	+	+	+	> seuil	signe de la dérivée

3 Contexte et alignement

3.1 Contextes de remplaçabilité

Idée de Z. Harris : ≪Les invariants linguistiques sont *remplaçables*≫ .

Soit un alphabet A. Un langage L : $L \subseteq A^*$.

Définition 1. *Pour un motif $w \in A^*$, son* contexte d'alignement *est défini par :*

$$C_L(w) =_{df} \{(u, v) \mid uwv \in L\}.$$

Equivalence contextuelle :

$$w_1 \equiv_L w_2 =_{df} C_L(w_1) = C_L(w_2).$$

Prouvez que $\equiv_L$ est une congruence par rapport à la concaténation.
Cette congruence est la plus fine parmi les congruences qui s'accordent avec L :

$$x \equiv_L y \ \wedge \ x \in L \ \Rightarrow \ y \in L.$$

Le quotient $A^*/\equiv_L$ est le *monoïde d'alignement* de L.

Exemple 3. *En anglais les phones s et $s_{fricative} = [\theta]$ ne sont pas équivalents $s \not\equiv_{ang} \theta$: $\ll lath \gg [la\theta] \neq \ll loss \gg [las]$. Mais ils le sont en français : $s \equiv_{fr} \theta$. C'est-à-dire, en français s et $s_{fricative} = [\theta]$ sont allophones du même phonème s.*

Plusieurs faits théoriques :

Théorème 1. *Si L est rationnel, alors $\equiv_L$ est finie et décidable et il existe un algorithme qui la définit à partir de l'AFD minimal pour L.*

Exemple 4. *Soit $A = \{a, b\}$. Pour $L_{ab} = \{wab \mid w \in A^*\}$ il y a cinq contextes d'alignement différents :*

$$[\varepsilon]_{L_{ab}} : C_{L_{ab}}(\varepsilon) = \{A^* \times L_{ab}, A^*a \times b, L_{ab} \times \varepsilon\}.$$
$$[a]_{L_{ab}} : C_{L_{ab}}(a) = \{A^* \times b, A^* \times L_{ab}\}.$$
$$[b]_{L_{ab}} : C_{L_{ab}}(b) = \{A^*a \times \varepsilon, A^* \times L_{ab}\}.$$
$$[ab]_{L_{ab}} : C_{L_{ab}}(ab) = \{A^* \times \varepsilon, A^* \times L_{ab}\}.$$
$$[b^2]_{L_{ab}} : C_{L_{ab}}(b^2) = \{A^* \times L_{ab}\}.$$

Par exemple, $ab^3 \equiv_{L_{ab}} b^3 \equiv_{L_{ab}} b^2$, $a^2 \equiv_{L_{ab}} ba \equiv_{L_{ab}} a$.

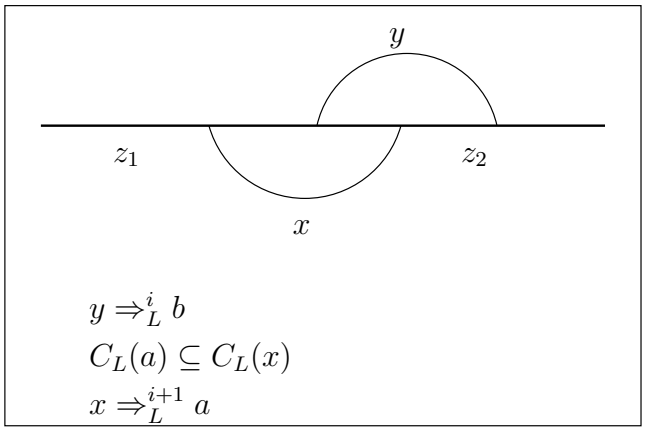
Théorème 2. *L'équivalence $\equiv_L$ n'est pas décidable dans la classe des langages hors contexte.*

3.2 Définition des structures syntagmatiques à partir d'alignement. Configurations

Définition 2 (Kulagina'58, Gladky'63).
Soit un ensemble de mots V et un langage $L \subseteq V^*$. Un motif $x \in V^*, |x| > 1$ est une configuration *du rang 1 réductible au mot $a \in V$, si $x \equiv_L a$.*

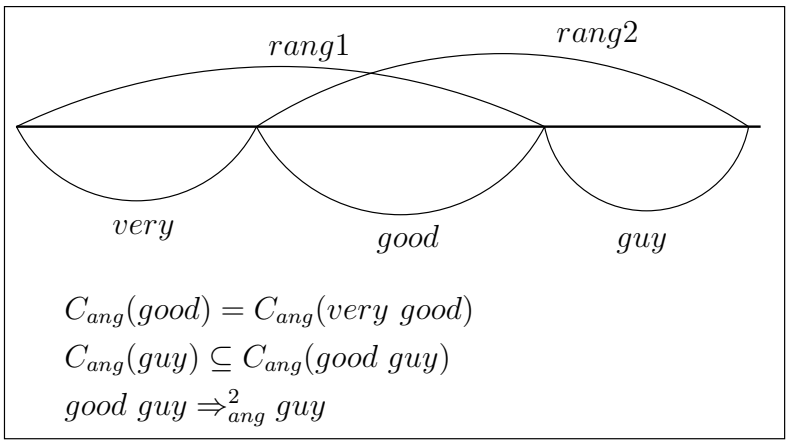
Exemple 5. very good $\equiv_{ang}$ good. *Ils représentent la même classe grammaticale des* constituants. *Mais tous les constituants ne sont pas réductibles. Par exemple,* good guy $\not\equiv_{ang}$ guy, *sinon* very good guy $\equiv_{ang}$ very guy. *D'où la définition suivante :*

Définition 3. *Supposons que les configurations des rangs $1 \leq i < r$ soient déjà définies. Alors $x \in V^*, |x| > 1$ est une* configuration du rang r réductible *à $a \in V$ (noté : $x \Rightarrow_L^r a$), si : (i) $C_L(a) \subseteq C_L(x)$ et*
(ii) quelque soit $(z_1, z_2) \in C_L(x)$, si z_1xz_2 ne contient pas de configuration $y \Rightarrow_L^i b, i < r$, en conflit avec x :

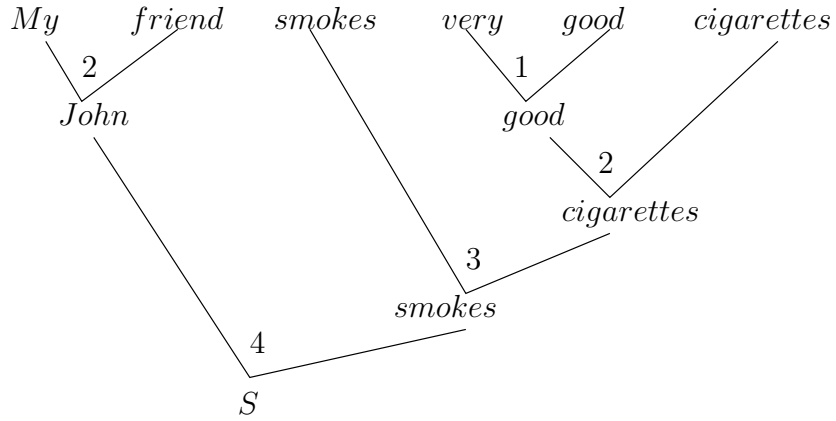


alors $(z_1, z_2) \in C_L(a)$.

Par exemple, le syntagme *good guy* est une configuration du rang 2 :



Plus généralement, $AdvAdj \equiv_{ang}^1 Adj$, $AdjN \Rightarrow_{ang}^2 N$, $VtN \Rightarrow_{ang}^3 V$, $NV \Rightarrow_{ang}^4 S$.
Ainsi, les contextes d'alignement correspondent aux classes des syntagmes et les rangs des configurations déterminent un ordre partiel sur ces classes. Par exemple :



3.3 Structure d'alignement

Définition 4. *Soit un langage* L *et* $x \in L$. *Si* $x = u_1 y u_2$ *pour une configuration* $y \Rightarrow_L^r a$, *alors* x *est dit* réductible. *Sinon* x *est* irréductible. *Une configuration* $y \Rightarrow_L^r a$ *est* irréductible

si y l'est. Nous appellons structure d'alignement *de L la paire $SA(L) = (PI(L), CI(L))$, où $PI(L)$ est l'ensemble des phrases irréductibles dans L et $CI(L)$ est l'ensemble des configurations irréductibles dans L.*

Il s'avère que la structure d'alignement d'un langage le définit complètement.

Théorème 3 (A. Gladky'1966). *Si $SA(L_1) = SA(L_2)$, alors $L_1 = L_2$.*

Exemple 6. *1. Soit $L_0 = \{b^i c b^i \mid i \geq 0\}$ et $L = a^+ L_0 a^+$. Alors $SA(L) = (PI_0, CI_0)$, est finie et $PI_0 = \{aca\}$, $CI_0 = \{a^2 \equiv_L a, bcb \equiv_L c\}$.*
2. La phrase Il pleut *est irréductible en Français.*

Théorème 4 (A. Gladky'1966). *Si $SA(L)$ est finie, alors L est hors contexte et il existe un algorithme qui construit une grammaire HC de L à partir de $SA(L)$.*

Par ailleurs, tous les langages hors contexte n'ont pas une structure d'alignement finie.

Exemple 7 (Luchkin'65). *La structure d'alignement du langage rationnel*

$$R = bab + bcb^+cb + (a^+ + cb^+c)^+$$

a $SA(R) = (PI(R), CI(R))$ infini : $PI(R) = bab + a^+$, $CI(R) = \{cb^m c \equiv_R a \mid m > 0\}$.

Et pourtant, en pratique la structure d'alignement est finie. Et pour apprendre la grammaire à partir d'exemples il suffit d'apprendre la structure d'alignement.

4 Acquisition à base d'alignement

(ABL : Alignment Based Learning)

4.1 Le cadre d'ABL

ABL est un cadre d'inférence symbolique robuste des ressources linguistiques à partir des contextes de ces ressources dans un corpus donné. Les méthodes ABL consistent à :

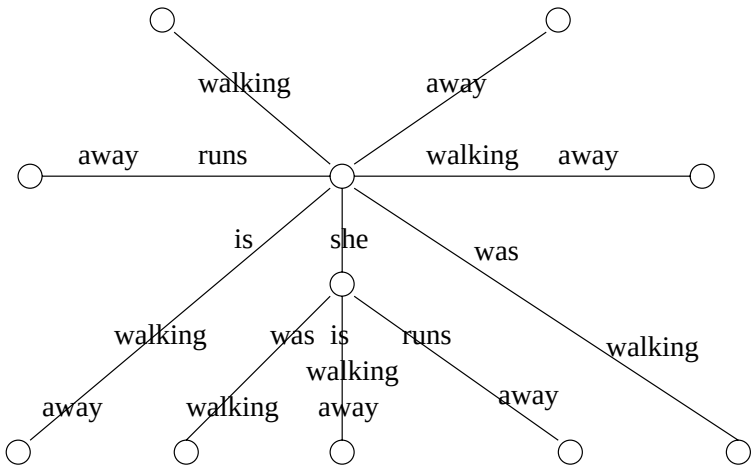
- construire les contextes d'alignement à partir du corpus,
- générer des hypothèses de ressources (les règles d'une grammaire, les morphèmes, les descriptions lexicales, etc.) en fonction des unités inter remplaçables dans les contextes générés,
- réduire l'espace des hypothèses en sélectionnant les hypothèses ≪optimales≫ (les plus compactes, les plus informatives).

Illustration : inférence grammaticale robuste par ABL [J. Geertzen, M. van Zaanen, Grammatical inference using suffix trees, *LNAI 3264*, 2004].
Méthode :
Entrée : Un corpus de phrases non-analysées
Sortie : Une grammaire hors contexte (éventuellement super-génératrice).
1. Construction des arbres des suffixes (STREE) et des préfixes des phrases.
2. Génération des constituants en fonction d'alignement des phrases dans STREE. Génération des règles hypothétiques à partir des constituants.
3. Optimisation des hypothèses des règles.

Définition 5. *L'arbre des suffixes $STREE(C)$ d'un corpus C est un arbre dont les arêtes sont étiquetées par des motifs tels que :*
- les premiers symboles des étiquettes sortant d'un même nœud sont deux-à-deux différents,
- la concaténation des motifs dans un chemin à partir de la racine vers une feuille est un suffixe d'une phrase de C,
- de cette manière les chemins de $STREE(C)$ représentent l'ensemble de tous les suffixes des phrases de C.

Exemple 8. *Soit le corpus $C = \{S_1, S_2, S_3\}$, où*
$S_1 = [she, is, walking, away]$
$S_2 = [she, was, walking]$
$S_3 = [she, runs, away]$

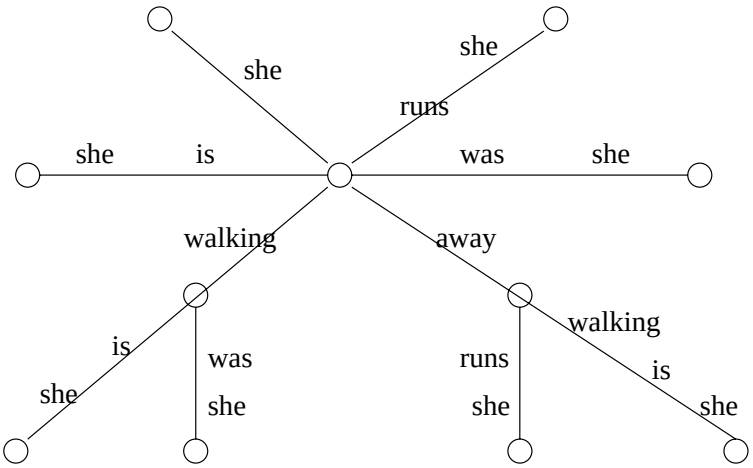
Alors l'arbre $STREE(C)$ est le suivant :



1. **Alignement aux suffixes.**
Pour tous deux constituants définis (au début : toutes deux phrases) introduire une nouvelle parenthèse ouvrante qui suit le préfixe maximum commun ; fermer cette parenthèse *à la fin des phrases* comparées.
2. Affecter les mêmes nouveaux non terminaux aux constituants alignés *aux mêmes préfixes.*

Voici le résultat de ces deux phases dans notre corpus :
she $[_a$ *is walking away* $]_a$
she $[_a$ *was walking* $]_a$
she $[_a$ *runs away* $]_a$

3. **Alignement aux préfixes.**
Construire l'arbre des préfixes $PTREE(C) =_{df} STREE(C^R)$, c'est-à-dire l'arbre des suffixes des phrases dans C avec l'ordre inversé. Voici cet arbre des préfixes :



Cette fois, pour tous deux constituants définis introduire une nouvelle parenthèse fermante qui précède le suffixe maximum commun dans cet arbre ; fermer cette parenthèse *au début des phrases comparées.*
4. Affecter les mêmes nouveaux non terminaux aux constituants alignés aux mêmes suffixes.

Voici le résultat de ces quatre phases :
$[_b\ [_c\ she\ [_a\ is\ ]_c\ walking\ ]_b\ away\ ]_a$
$[_c\ she\ [_a\ was\ ]_c\ walking\ ]_a$
$[_b\ she\ [_a\ runs\ ]_b\ away\ ]_a$

5. Supprimer les parenthèses au début et à la fin des phrases. Apparier les parenthèses restantes :
(*i*) l'appariement est défini de l'extérieur vers l'intérieur,
(*ii*) essayer tous les appariements des parenthèses sans introduire des parenthèses croisées en ajoutant, s'il faut, les parenthèses manquantes,
(*iii*) unifier les non terminaux des constituants qui se trouvent dans un même contexte.

Voici le calcul dans cette phase :
$she\ [_a\ is\ ]_c\ walking\ ]_b\ away$
$she\ [_a\ was\ ]_c\ walking$
$she\ [_a\ runs\ ]_b\ away$

Appariement :

$$(a \leftrightarrow c) \text{ deviennent } d\ (contexte\ :\ (she..walking)\ )$$

$$(a \leftrightarrow b) \text{ deviennent } e\ (contexte\ :\ (she..away)\ )$$

Résultat :
$she\ [_e\ [_d\ is\ ]_d\ walking\ ]_e\ away$
$she\ [_d\ was\ ]_d\ walking$
$she\ [_e\ runs\ ]_e\ away$

Les méthodes d'optimisation des hypothèses sont standards : on choisit les hypothèses avec l'entropie minimale et l'information maximale vis-à-vis des contextes et on minimise le nombre des règles.

Le problème majeur d'utilisation en pratique des méthodes ABL est la complexité de construction des contextes d'alignement à partir des corpus. Les méthodes naïves ont la complexité $\mathcal{O}(|C|^2)$ (comparaison de chaque phrase contre toutes les phrases). La complexité de la méthode présentée est proportionnelle à la complexité de construction de l'arbre des suffixes $STREE(C)$ (voir l'appendice).

5 Acquisition symbolique dans les familles indexées de ressources

Définition 6. *Soit un domaine $D \neq \emptyset$ d'objets énuméré par une fonction $E_{\mathcal{L}}$ et un ensemble d'indices $\mathcal{I}$ énuméré par une fonction $I_{\mathcal{L}}$. Alors une famille $\mathcal{L} = \{L(I) \mid I \in \mathcal{I}\}$ est dite* famille indexée de ressources, *si tout son élément (une* ressource*) $L(I), I \in \mathcal{I}$, est un sous-ensemble non vide $L(I) \subseteq D$.*

La fonction

$$F_{\mathcal{L}}(I, w) = \begin{cases} 1, & si\ I \in \mathcal{I}, w \in L(I), \\ 0, & si\ I \in \mathcal{I}, w \in D - L(I). \end{cases}$$

est dite uniformement caractéristique *pour $\mathcal{L}$.*

La famille indexée de ressources $\mathcal{L}$ est récursive si les fonctions $E_{\mathcal{L}}$, $I_{\mathcal{L}}$ et $F_{\mathcal{L}}$ sont toutes récursives.

Evidemment, chaque ensemble $L(I_0)$ dans la famille indexée récursive $\mathcal{L}$ est récursif. Sa fonction caractéristique est $\lambda\ w.F_{\mathcal{L}}(I_0, w)$.

Nous ne précisons pas la nature des éléments des ensembles $L(I)$ dans $\mathcal{L}$. Les ressources à acquérir sont les **indices** I de ces ensembles qui les identifient. Le cadre général d'acquisition des ressources dans la famille indexée $\mathcal{L}$ est défini en termes d'une fonction d'apprentissage M : M s'applique à des séquences infinies *d'exemples*, c'est-à-dire, d'objets dans D munis des ≪signes d'appartenance≫

$$\sigma = (s_1, \alpha_1), (s_2, \alpha_2), (s_3, \alpha_3), \ldots$$

où $\alpha_i \in \{+, -\}, i \geq 1$. Les signes + et − sont interprétés comme ≪s_i appartient (n'appartient pas) à la ressource cible≫ . Pour tout préfixe fini $\sigma[k]$ de σ, M rend son hypothèse de ressource cible $h_k = M(\sigma[k]) \in \mathcal{I}$, ce qui correspond à l'hypothèse que la ressource cible est $L(h_k)$.

Deux schémas d'acquisition sont distingués : avec et sans tuteur.

5.1 Acquisition avec tuteur

Définition 7. *Soit une ressource cible $L(I_0) \in \mathcal{L}$ et une fonction T qui énumère une séquence*

$$\sigma = (s_1, \alpha_1), (s_2, \alpha_2), (s_3, \alpha_3), \ldots \tag{1}$$

Alors, σ est une séquence d'exemples de tuteur *T pour $L(I_0)$ si :*
(a) $\{s_i \mid \exists\ \alpha_i\ ((s_i, \alpha_i) \in \sigma)\} = D$,
(b) quel que soit un exemple $(s_i, \alpha_i) \in \sigma$, $s_i \in L(I_0)$ ssi $\alpha_i = +$.

Une fonction M apprend *$L(I_0)$* sur la séquence de tuteur *(1), si :*
(c) pour tout k, l'hypothèse $h_k = M(\sigma[k])$ est compatible *avec $\sigma[k]$, c'est-à-dire,*

$$\{s_i \mid (s_i, +) \in \sigma[k]\} \subseteq L(h_k),$$
$$\{s_i \mid (s_i, -) \in \sigma[k]\} \subseteq D - L(h_k) \text{ et}$$

(*d*) *il existe un préfixe* $\sigma[K]$ *de* σ *tel que* $M(\sigma[j]) = h_K$ *pour tous* $j \geq K$ *et* $L(h_K) = L(I_0)$. K *s'appelle* point de stabilisation *de* M.

M apprend $L(I_0)$ avec tuteur *si elle l'apprend sur toutes les séquences de tuteur.*

M apprend avec tuteur la famille indexée $\mathcal{L}$ *si elle apprend avec tuteur toute sa ressource* $L \in \mathcal{L}$. *Si une telle fonction* M *existe, alors on dit que* $\mathcal{L}$ *est* apprenable avec tuteur.

Proposition 1. *1. Si* $\sigma = (s_1, \alpha_1), (s_2, \alpha_2), (s_3, \alpha_3), \ldots$ *est une séquence d'exemples de tuteur pour* $L \in \mathcal{L}$, *alors* $\{s_i \mid (s_i, +) \in \sigma\} = L$, *et* $\{s_i \mid (s_i, -) \in \sigma\} = D - L$.
2. Toute famille indexée $\mathcal{L}$ *est apprenable avec tuteur.*
3. Si la famille $\mathcal{L}$ *est récursive, alors elle est apprenable avec tuteur par un algorithme (une fonction recursive)* $M_{\mathcal{L}}$.

Voici un algorithme $M_{\mathcal{L}}$ qui apprend $\mathcal{L}$ avec tuteur si la famille $\mathcal{L}$ est récursive.
Algorithme d'apprentissage avec tuteur
Entrée : Le préfixe $\sigma[k] = (s_1, \alpha_1), \ldots, (s_k, \alpha_k)$ d'une longueur $k > 0$ d'une séquence de tuteur σ.
Sortie : $h_k \in \mathcal{I}$ (une hypothèse)

DANS LA BOUCLE d'énumération de $\mathcal{I}$ par $I_{\mathcal{L}}$
  Calculer l'indice minimum $h_k \in \mathcal{I}$ **tel que**
  POUR TOUT $1 \leq i \leq k$
    SI $\alpha_i = +$
    ALORS $F_{\mathcal{L}}(h_k, s_i) = 1$
    SINON $F_{\mathcal{L}}(h_k, s_i) = 0$
    FIN_SI
  FIN_PT;
  RENDRE h_k
FIN_BOUCLE

Exemple 9. *Familles indexées de langages :*
1. Famille indexée des langages hors contexte. Les indexes $\mathcal{I}_{HC}$ *sont les grammaires HC elles-mêmes (*$\mathcal{I}_{HC}$ *est énumérable),* $\mathcal{L}_{HC} = \{L(G) \mid G \in \mathcal{I}_{HC}\}$. $F_{\mathcal{L}_{HC}}$ *est récursive : par exemple, c'est l'algorithme de parsing de Cock & Kasami & Younger.*

2. Famille indexée $\mathcal{L}_{n^6}$ *des langages acceptés par des machines de Turing en temps* $\mathcal{O}(n^6)$. $\mathcal{I}_{n^6}$ *est une famille enumérable des MTD qui acceptent tous ces langages. Certes, il existe une fonction récursive* $F_{\mathcal{L}_{n^6}}$.

3. Des autres exemples : les langages rationnels, linéaires, contextuels, etc.

Selon la Proposition 1, toutes ces familles sont apprenables avec tuteur. Par ailleurs, selon les témoignages des cognitivistes, les enfants en bas âge apprennent la langue maternelle pratiquemment sans tenir compte des corrections, c'est-à-dire à partir d'exemples positifs. D'où l'intérêt de l'acquisition sans tuteur.

5.2 Acquisition sans tuteur

Définition 8 (E. Gold, 1965). *Soit une famille indexée de ressources $\mathcal{L}$ et une ressource cible $L_c \in \mathcal{L}$. Une séquence (finie ou infinie)*

$$\sigma = s_1, s_2, s_3, \ldots \tag{2}$$

est une séquence d'exemples positifs *pour L_c si $\{s_i \mid s_i \in \sigma\} = L_c$ (c'est-à-dire, σ est une énumération de L_c).*

Une fonction M apprend *L_c* à la limite sur *σ, s'il existe un préfixe $\sigma[K]$ de σ tel que $M(\sigma[j]) = h_K$ pour tous $j \geq K$ et $L(h_K) = L_c$. K s'appelle* point de stabilisation *de M sur σ.*

M apprend *L_c* à partir d'exemples positifs *si elle l'apprend à la limite sur toutes les séquences d'exemples positifs pour L_c.*

M apprend *la famille indexée $\mathcal{L}$* à partir d'exemples positifs *si elle apprend à partir d'exemples positifs toute ressource $L \in \mathcal{L}$. Si une telle fonction M existe, alors on dit que $\mathcal{L}$ est* apprenable à partir d'exemples positifs (P-apprenable). *Si, de plus, cette fonction est récursive, alors $\mathcal{L}$ est* récursivement apprenable.

5.3 APPENDICE : Construction rapide des arbres des suffixes

Il existe des méthodes de construction de $STREE(C)$ en temps linéaire (par exemple, l'algorithme de E. Ukkonen (Information Processing, 1992, 92(1))

Nous présentons un algorithme moins détaillé de E.M. McGreight (JACM, 1976, 32(2)) dont la complexité moyenne est $\mathcal{O}(|C| \log |C|)$. Tous les algorithmes utilisent les notations et les notions suivantes utilisées dans l'algorithme de McGreight :

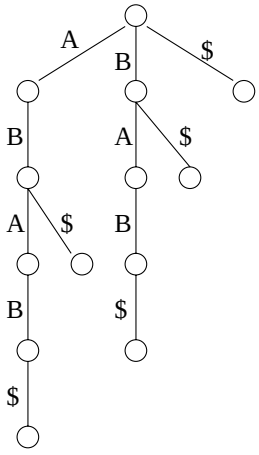
Définition 9. *L'arbre des suffixes Y est défini en n pas où n est la longueur du texte donné y. Y_j dénote l'arbre après le pas j. $y(i,n)$ est le suffixe de y à partir de la position $i, 1 \leq i \leq n$ et jusqu'à sa fin.*
Position *d'un motif u dans Y est le nœud de Y qui représente u.*
Extension *d'un motif u est tout motif qui a le préfixe u.*
Position étendue *d'un motif u dans Y est la position de l'extension la plus courte de u qui est présente dans Y.*
Position contractée *d'un motif u dans Y est la position du préfixe le plus long de u présent dans Y.*
$head_i$ est le préfixe maximum de $y(i,n)$ qui est en même temps un préfixe de $y(j,n)$, $j < i$. C'est-à-dire, *$head_i$* est le préfixe maximum de $y(i,n)$ dont la position étendue existe dans Y_{i-1}.
$tail_i$ est le motif tel que : $y(i,n) = head_i tail_i$.

Algorithme

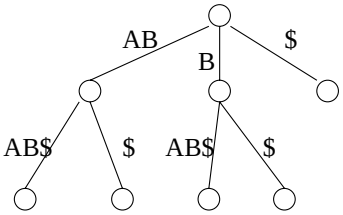
Y_0 est l'arbre singleton (à un nœud)
POUR TOUT i, $1 \leq i \leq n$ (incrémentalement)
  Y_i := insérer$(Y_{i-1}, y(i,n))$
FIN_PT

insérer$(Y_{i-1}, y(i,n))$:
1. Trouver la position étendue p de $head_i$ dans Y_{i-1}. Soit le motif $s(p)$ représenté par p.

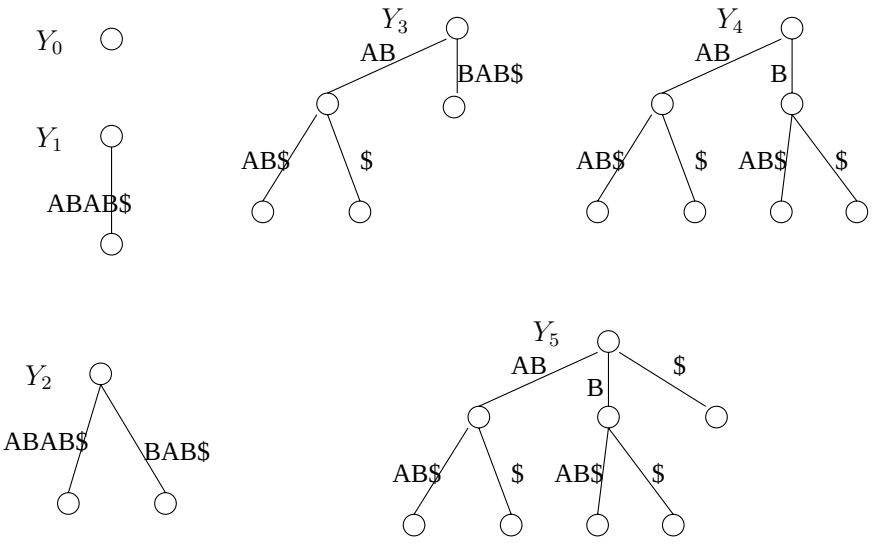
2. Si $s(p) \neq head_i$, c'est-à-dire, $s(p) = head_i\ v, v \neq \varepsilon$, alors l'arête qui entre dans p est décomposée en deux par un nouveau nœud q : la première arête est étiquetée par $head_i$ et la deuxième est étiquetée par v.
3. La nouvelle arête est rajoutée à q et elle est étiquetée par $tail_i$.
Voici le calcul de l'arbre des suffixes pour $y = ABAB$:



STRIE de ABAB$



STREE de ABAB$



Construction du STREE de ABAB$