Apprentissage des boules de mots avec des requêtes de correction

Leonor Becerra Bonache[1], Colin de la Higuera[2],
Jean-Christophe Janodet[2], Frédéric Tantini[2]

[1] Research Group on Mathematical Linguistics
Rovira i Virgili University
Pl. Imperial Tárraco 1, 43005 Tarragona, Spain
leonor.becerra@urv.cat

[2] Laboratoire Hubert Curien (ex-EURISE)
Université Jean Monnet
18 rue du Professeur Benoît Lauras, 42000 Saint-Étienne
{cdlh,janodet,tantini}@univ-st-etienne.fr

Résumé : Dans les années 80, Angluin a développé un paradigme d'apprentissage actif basé sur un oracle, capable de répondre à des requêtes d'appartenance et des requêtes d'équivalence. Or, si dans les différentes applications de l'apprentissage actif, les réponses aux premières sont souvent faciles à obtenir, avoir droit aux secondes n'est pas toujours réaliste. Pour contourner cette difficulté, nous proposons un nouveau type de requêtes, appelées *requêtes de correction*, que l'on étudie ici dans un contexte d'inférence grammaticale. Quand un mot est soumis à l'oracle, ce dernier le valide s'il appartient au langage cible, ou bien propose une correction de ce mot. Une telle correction est un mot du langage qui est proche de la requête (au sens de la distance d'édition). Nous introduisons ensuite une classe non triviale de langages, celle des boules topologiques de mots. Nous montrons que cette classe n'est pas apprenable dans le modèle d'Angluin, mais qu'elle l'est, avec un nombre linéaire de requêtes de correction, voire logarithmique sous certaines hypothèses. Enfin, nous étudions le bon comportement expérimental de nos algorithmes.

Mots-clés : Inférence grammaticale, apprentissage actif, requêtes de correction, distance d'édition, boules.

1 Introduction

Savez-vous combien Babylone a connu de rois qui s'appelait *Nabucodonosaur* ? Et connaissez-vous la date de naissance d'*Arnold Shwartzeneger* ? Autrefois, il n'y a pas si longtemps, vous auriez cherché des réponses en consultant un Quid ou une encyclopédie. C'était l'époque où vous participiez aux grands concours de l'été des magazines télé, mais à cause de ces questions, vous auriez sûrement raté le premier prix. Pour-

quoi ? . . . Aujourd'hui, tout est différent. Pour répondre à ces questions, quoi de plus naturel que d'utiliser le web, de lancer son moteur de recherche favori, de taper deux mots-clés, de suivre trois liens, et vous obtenez des réponses. Vous découvrez en particulier . . . qu'aucun roi ne s'est appelé *Nabucodonosaur* à Babylone, mais qu'il y en a eu deux qui se prénommaient *Nabuc***h***odonos***o***r*. De même, la date de naissance d'*Arnold Shwartzeneger* est incertaine, mais il est facile de savoir qu'*Arnold S***c***hwa***rz***ene***g***ger* est né le 30 juillet 1947.

Si vous seriez aujourd'hui en mesure de gagner les grands concours d'autrefois, c'est parce les moteurs de recherche actuels sont capables de proposer des *corrections* lorsqu'un mot-clé est peu fréquent. Ces corrections sont d'autant plus fiables que ce sont les milliards de pages web qui forment le dictionnaire de référence. Autrement dit, le web et les moteurs de recherche peuvent être vus comme des oracles puissants capables, non seulement de donner une réponse pertinente à une requête pertinente, mais aussi de détecter et de corriger une requête douteuse. En somme, nous disposons aujourd'hui d'oracles capable de résoudre ce que nous appellerons des *requêtes de correction*.

Dans l'exemple précédent, la distance qui est utilisée par le moteur de recherche pour trouver un mot proche est une *distance d'édition* qui mesure le nombre minimum d'opérations d'effacement, d'insertion et de substitution de lettres nécessaires pour transformer un mot en un autre (Levenshtein, 1965; Wagner & Fisher, 1974). Cette distance et ses variantes, où chaque opération peut avoir un poids différent, a été utilisée dans de nombreux domaines comme la reconnaissance des formes (Miclet, 1986), la bioinformatique (Durbin *et al.*, 1998) et la modélisation de langages (Amengual *et al.*, 2001). En inférence grammaticale, on la rencontre dans deux types de travaux.

D'une part, certains travaux utilisent des techniques d'inférence grammaticale probabiliste pour apprendre les poids des opérations d'édition (Bernard *et al.*, 2006). D'autre part, la distance d'édition intervient naturellement dans certains problèmes d'inférence grammaticale, en particulier quand on veut apprendre des langages à partir de données bruitées (Tantini *et al.*, 2006). Toutefois, dans ce dernier cas, on peut remarquer que les langages étudiés ne sont pas les langages traditionnels de la hiérarchie de Chomsky. En effet, même la classe la plus simple, celle des langages réguliers, n'est pas robuste au bruit d'édition, car les fonctions de parité ne sont pas apprenables en présence de bruit (Kearns & Li, 1993). De même, nous considérerons ici des langages finis, élémentaires du point de vue de la théorie des langages mais pertinent du point de vue topologique et complexe du point de vue combinatoire : les boules de mots.

Dans ce travail, nous étudions donc le problème de l'identification des boules de mots à partir de requêtes de correction. Nous commençons par introduire la distance d'édition et les boules dans la Section 2. Ensuite, nous montrons dans la Section 3 qu'elles ne sont pas apprenables à partir des requêtes d'appartenance et d'équivalence d'Angluin, ce qui justifie l'introduction des requêtes de correction. Nous montrons dans la Section 4 que les boules sont apprenables avec un nombre linéaire de telles requêtes. Dans la Section 5, nous étudions les limites de notre algorithme d'un point de vue expérimental. Puis dans la Section 6, nous proposons un nouvel algorithme utilisant un nombre logarithmique de requêtes, adapté à une distance d'édition pondérée. Enfin, nous concluons en Section 7.

2 Langage des boules de mots

Un *alphabet* Σ est un ensemble fini non vide de symboles qu'on appelle des *lettres*. Un *mot* est une séquence finie de lettres de Σ et on définit Σ^* comme l'ensemble de tous les mots. La longueur d'un mot w sera notée $|w|$, et $|w|_a$ désignera le nombre d'occurrences d'une lettre a dans w. On dira qu'un mot u est un *sous-mot* de v, noté $u \preceq v$, si $u = x_1 x_2 \dots x_n$ et il existe $u_0, u_1, \dots, u_n \in \Sigma^*$ tels que $v = u_0 x_1 u_1 \dots x_n u_n$. Toute partie de Σ^* s'appelle un *langage*. En particulier, on notera Σ^k l'ensemble des mots de longueur k et $\Sigma^{\leq k}$ l'ensemble des mots de longueur inférieure ou égale à k. $\mathbb{N}$ sera l'ensemble des entiers naturels.

On peut définir la distance d'édition entre deux mots w et w' en utilisant la longueur minimale des dérivations de réécriture permettant de passer de w à w'. Plus précisément, étant donnés deux mots $w, w' \in \Sigma^*$, on dit que w se réécrit en w' en un pas, noté $w \to w'$ si une des conditions suivantes est vérifiée :

- *opération d'effacement* : $w = uav$ et $w' = uv$ avec $u, v \in \Sigma^*$ et $a \in \Sigma$;
- *opération d'insertion* : $w = uv$ et $w' = uav$ avec $u, v \in \Sigma^*$ et $a \in \Sigma$;
- *opération de substitution* : $w = uav$ et $w' = ubv$ avec $u, v \in \Sigma^*$, $a, b \in \Sigma$, $a \neq b$.

Nous noterons $w \xrightarrow{k} w'$ si w peut se réécrire en w' à l'aide de k opérations d'édition.

Définition 1 (Distance d'édition)
La distance d'édition entre deux mots w et w', notée $d(w, w')$, est le nombre minimum d'opérations d'édition nécessaires pour réécrire w en w' ; c'est donc le plus petit entier $k \in \mathbb{N}$ tel que $w \xrightarrow{k} w'$.

Exemple : On a $d(abaa, aab) = 2$ car $abaa$ nécessite au moins deux pas pour être transformée en aab : $a\underline{b}aa \to aa\underline{a} \to aab$. Notons qu'on peut effectuer le calcul de $d(w, w')$ en $\mathcal{O}(|w| \cdot |w'|)$ par programmation dynamique (Wagner & Fisher, 1974). $\Diamond$

La propriété technique suivante stipule que la distance d'édition entre deux mots est plus grande que le nombre minimum d'insertions qu'il faut faire dans l'un des deux pour égaliser les longueurs :

Proposition 1
Pour tout $w, w' \in \Sigma^*$, $d(w, w') \geq \big||w| - |w'|\big|$.
De plus, $d(w, w') = \big||w| - |w'|\big|$ ssi ($w \preceq w'$ ou $w' \preceq w$).

Il n'est pas très difficile de démontrer que la distance d'édition est une vraie distance au sens mathématique du terme (Crochemore *et al.*, 2001). Aussi, quand on munit Σ^* de cette distance, on obtient un espace métrique, donc un espace *topologique*. Il est donc naturel de s'intéresser aux boules de mots dans cet espace, car d'une certaine manière, ce sont les langages les plus simples que l'on puisse considérer :

Définition 2 (Boule)
Soient $o \in \Sigma^*$ et $r \in \mathbb{N}$. On appelle boule de centre o et de rayon r l'ensemble $B_r(o)$ des mots qui sont à distance d'édition inférieure ou égale à r de o :

$$B_r(o) = \{w \in \Sigma^* | d(o, w) \leq r\}.$$

Exemple : Soit $\Sigma = \{a, b\}$. On a $B_1(ba) = \{a, b, aa, ba, bb, aba, baa, bab, bba\}$ et $B_r(\lambda) = \Sigma^{\leq r}$ pour tout rayon $r \in \mathbb{N}$. $\Diamond$

La définition des boules est simplissime, mais ne nous fions pas aux apparences : le nombre de mots qu'elles contiennent est exponentiel en fonction du rayon dès que $|\Sigma| \geq 2$. C'est le cas de la boule $B_r(\lambda) = \Sigma^{\leq r}$ contenant $(|\Sigma|^{r+1} - 1)/(|\Sigma| - 1)$ mots. Il est difficile d'estimer ce nombre dans le cas d'une boule quelconque. Toutefois, on peut se faire une idée expérimentalement (voir Table 1). On constate clairement que pour une longueur de centre fixée, le nombre moyen de mots fait plus que doubler chaque fois que le rayon augmente de 1. Compte tenu de ces chiffres, il est inenvisageable de stocker l'ensemble de tous les mots d'une boule en machine pour travailler.

Longueur du centre	Rayon					
	1	2	3	4	5	6
0	3.0	7.0	15.0	31.0	63.0	127.0
1	6.0	14.0	30.0	62.0	126.0	254.0
2	8.6	25.6	56.5	119.7	246.8	501.6
3	10.8	41.4	101.8	222.8	468.6	973.0
4	13.1	61.4	173.8	402.9	870.9	1850.8
5	16.3	91.0	285.1	698.5	1584.4	3440.9
6	17.9	125.8	441.2	1177.5	2771.3	6252.9
7	21.2	166.9	678.0	1908.8	4835.8	11233.5
8	24.3	200.2	1034.2	3209.9	8358.1	19653.6
9	26.0	265.4	1390.9	5039.6	13677.8	34013.1

TAB. 1 – Nombre moyen de mots dans une boule pour un alphabet à 2 lettres. Les centres sont tirés au hasard, chaque calcul se fait sur 20 centres.

Cette dernière remarque permet de poser la question de la représentation d'une boule. En effet, on pourrait utiliser un automate, puisque les boules sont des langages finis et donc réguliers. Cependant, si $B_r(o)$ est reconnu par un automate avec λ-transitions ayant $\mathcal{O}(|o| \cdot r)$ états, les automates finis déterministes (AFD), eux, ont souvent un nombre exponentiel d'états. En particulier, Schulz & Mihov (2002) propose une construc-

Rayon r	1	2	3	4
Nombre d'états	$5 \cdot \|o\|$	$30 \cdot \|o\|$	$180 \cdot \|o\|$	$1353 \cdot \|o\|$

TAB. 2 – Nombre d'états de l'automate de Levenshtein (Schulz & Mihov, 2002) reconnaissant $B_r(o)$ lorsque $|\Sigma| \geq 2$. L'alphabet lui-même ne joue pas sur le nombre d'états, mais seulement sur le nombre de transitions.

tion directe dont le nombre d'états, ainsi que le temps et l'espace de construction, sont linéaires en $|o|$ mais exponentiels en r (voir Table 2). Leur construction ne retourne pas des AFD minimaux, mais nous ne prenons pas beaucoup de risque en affirmant :

Conjecture : Dans le pire cas, l'AFD minimal de $B_r(o)$ contient $\Omega(2^r \cdot |o|)$ états.

En conséquence, choisir l'AFD minimal d'une boule pour la représenter n'est pas très pertinent. D'un autre côté, pour déterminer si un mot w appartient à une boule $B_r(o)$,

il suffit (1) de calculer $d(o, w)$ et (2) de vérifier si cette distance est $\leq r$. Clairement, on peut résoudre ce problème en temps $\mathcal{O}(|o| \cdot |w| + \log r)$. Donc le couple (o, r), de taille $|o|+\log r$, est un candidat pertinent pour représenter la boule $B_r(o)$. Toutefois, des boules telles que $r > 2^{|o|}$ poseront des problèmes de complexité. Par exemple, aucun mot en dehors de la boule $B_r(\lambda) = \Sigma^{\leq r}$ n'est de taille polynomial en $0 + \log r$. Donc dans la suite, nous supposerons toujours que $|o|$ et r sont du même ordre de grandeur.

Enfin, et c'est remarquable, il arrive qu'une boule admette plusieurs représentations. En particulier, s'il n'y a qu'une lettre dans l'alphabet $\Sigma = \{a\}$, alors $B_2(a) = B_3(\lambda) = \{\lambda, a, aa, aaa\}$ par exemple. Mais c'est un cas particulier :

Théorème 1
Si $|\Sigma| \geq 2$ *et* $B_{r_1}(o_1) = B_{r_2}(o_2)$, *alors* $o_1 = o_2$ *et* $r_1 = r_2$.

En conséquence, quand l'alphabet a au moins deux lettres, (o, r) est un représentant unique, donc *canonique*, de la boule $B_r(o)$.

Preuve : **Point 1** : si $B_{r_1}(o_1) = B_{r_2}(o_2)$, alors $|o_1| + r_1 = |o_2| + r_2$. En effet, soit $w \in \Sigma^{r_1}$. Alors $d(o_1, o_1 w) = |w| = r_1$ par la Prop. 1, donc $o_1 w \in B_{r_1}(o_1)$, donc $o_1 w \in B_{r_2}(o_2)$, donc $d(o_1 w, o_2) \leq r_2$. Or $d(o_1 w, o_2) \geq |o_1 w| - |o_2|$, d'après la Prop. 1. Aussi, on en déduit que $r_2 \geq |o_1 w| - |o_2| = |o_1| + r_1 - |o_2|$, donc $|o_2| + r_2 \geq |o_1| + r_1$. On montre de même que $|o_1| + r_1 \geq |o_2| + r_2$. **Point 2** : si $|\Sigma| \geq 2$ et que $o_2 \not\preceq o_1$, alors il existe $w \in \Sigma^*$ tel que (i) $|w| = r_1 + |o_1|$, (ii) $o_1 \preceq w$ et (iii) $o_2 \not\preceq w$. En effet, supposons que o_2 commence par un a et soit $b \in \Sigma \setminus \{a\}$; on pose $w = b^{r_1} o_1$, et on obtient le résultat. **Théorème** : supposons que $o_1 \neq o_2$. Alors soit $o_1 \not\preceq o_2$, soit $o_2 \not\preceq o_1$. Supposons le second cas, sans perte de généralité. Alors, avec le point précédent, il existe un mot w tel que (i) $|w| = r_1 + |o_1|$, (ii) $o_1 \preceq w$ et (iii) $o_2 \not\preceq w$. Comme $o_1 \preceq w$, on en déduit que $d(o_1, w) = |w| - |o_1|$, par la Prop. 1, et comme $|w| = r_1 + |o_1|$, on a $d(o_1, w) = r_1$. Donc $w \in B_{r_1}(o_1)$. Par contre, comme $o_2 \not\preceq w$, d'après la Prop. 1, on a $d(o_2, w) > ||w| - |o_2||$ et comme $|w| = r_1 + |o_1|$, on obtient $d(o_2, w) > |r_1 + |o_1| - |o_2|| = r_2$. Comme $d(o_2, w) > r_2$, on en déduit que $w \notin B_{r_2}(o_2)$, donc que $B_{r_1}(o_1) \neq B_{r_2}(o_2)$, ce qui est impossible. Par conséquent, $o_1 = o_2$, et donc $r_1 = r_2$ (puisque $|o_1| + r_1 = |o_2| + r_2$). □

3 Apprentissage des boules avec des requêtes

L'apprentissage actif est un paradigme faisant intervenir un apprenant (algorithme) et un oracle. Le but de l'apprenant est d'identifier la représentation d'un langage inconnu en posant des requêtes à l'oracle. Ce dernier connaît le langage cible et répond correctement aux requêtes (*i.e.*, l'oracle ne ment pas). En outre, l'apprenant est soumis à des contraintes d'efficacité : (1) il ne peut poser qu'un nombre polynomial de requêtes et (2) le temps global dont il dispose doit être polynomial dans la taille de la représentation cible. Depuis les travaux fondateurs d'Angluin (Angluin, 1987), deux types de requêtes sont utilisées de manière standard, les requêtes d'*appartenance* (MQ) et requêtes d'*équivalence* (EQ).

Définition 3 (Requête d'appartenance et d'équivalence)
Soit $\Re$ une classe de langages de Σ^* et $L \in \Re$. Dans le cas des requêtes d'appartenance, l'apprenant soumet un mot $w \in \Sigma^*$ à l'oracle ; sa réponse, notée $MQ(w)$ est soit OUI *si $w \in L$, soit* NON *si $w \notin L$.*

Dans le cas des requêtes d'équivalence, l'apprenant soumet (la représentation d') un langage $K \in \Re$ à l'oracle ; sa réponse, notée $EQ(K)$ est :

- *si $K = L$, alors $EQ(K) =$* OUI *;*
- *si $K \neq L$, alors l'oracle retourne un mot de $K \triangle L$, donc un mot de la différence symétrique de K et L.*

Clairement, il est raisonnable de penser que les EQ sont trop riches, trop puissantes pour exister en tant que telle ou même simplement pour être simulées. Dans certaines applications, on peut les remplacer par un tirage aléatoire de mots qui servent à poser des MQ (*sampling*), mais il est souvent difficile de définir une distribution pertinente (de la Higuera, 2006). De plus, nous n'utiliserons pas les MQ et les EQ dans la suite car en fait, les boules ne sont pas apprenables dans le modèle d'Angluin :

Théorème 2
Soient $n, m \in \mathbb{N}$ et $\mathcal{B}_{\leq n,m} = \{B_r(o) \mid o \in \Sigma^*, |o| \leq n, r \leq m\}$. N'importe quel algorithme qui identifie exactement chaque boule hypothèse de $\mathcal{B}_{\leq n,m}$ en utilisant EQ et MQ fait, dans le pire cas, au moins $2^n - 1$ requêtes.

Preuve : Il existe $N = 2^n$ boules de rayon 0 ayant un centre de longueur n. Nous décrivons ici un adversaire (à la manière d' Angluin (1988)) qui maintient un ensemble S des boules non éliminées. Initialement, S contient toutes les boules de $\mathcal{B}_{\leq n,m}$. Pour une requête d'équivalence $L = B_r(o)$, la réponse est NON et le contre-exemple est o. Ceci élimine entre autres les boules de centre o et de rayon inférieur à r, mais qu'une seule boule de rayon 0. Pour une requête d'appartenance du mot o, la réponse est NON, et l'unique boule de rayon 0, $B_0(o)$, est enlevée de l'espace hypothèse (ainsi que toutes les boules de rayon strictement supérieur à 1 contenant o). À chaque instant, les boules de rayon 0 restantes dans S sont compatibles avec les réponses aux précédentes requêtes. De plus, chaque requête enlève une et une seule boule de rayon 0 de S. Ainsi, $2^n - 1$ requêtes sont nécessaires dans le pire cas. □

Nous allons voir que les requêtes de *correction* (CQ) permettent de résoudre ces problèmes. Nous les définissons ci-dessous :

Définition 4 (Requête de correction)
Soient L un langage fixé et w un mot de Σ^* que l'apprenant soumet à l'oracle. Alors, la réponse de l'oracle, notée $CQ(w)$, est de deux types :

- *si $w \in L$, alors $CQ(w) =$* OUI *;*
- *si $w \notin L$, alors l'oracle retourne une correction de w par rapport à L, c'est-à-dire un mot w' qui appartient à L et qui est à distance d'édition minimum de w :*

$$CQ(w) \quad = \quad \textit{un mot de } \{w' \in L \mid d(w, w') \textit{ est minimum } \}$$

Remarquons tout de suite que les CQ ne présentent pas l'inconvénient des EQ : non seulement, elles peuvent facilement être simulées en connaissant le langage cible, mais

elles existent “à l’état naturel” dans certaines applications, comme nous l’avons souligné dans l’introduction. De plus, on peut remarquer qu’elles sont pertinentes d’un point de vue cognitif : il est communément admis qu’un enfant acquiert sa langue maternelle en utilisant les corrections que lui suggère son entourage (Becerra-Bonache & Yokomori, 2004).

Enfin, les CQ comme les boules reposent sur une distance, ce qui laisse espérer des résultats d’apprenabilité. En effet, supposons un instant qu’on travaille non pas avec des mots, mais avec la distance euclidienne et des disques du plan (voir Fig. 1).

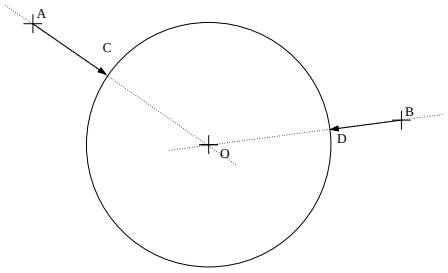

FIG. 1 – Apprentissage d’un disque de $\mathbb{R}^2$ avec des CQ.

Pour apprendre le disque de centre O et de rayon R avec des requêtes de correction, on peut procéder en trois temps. (1) On commence par chercher deux points A, B qui sont éloignés l’un de l’autre et qui sont en dehors du disque qu’on veut identifier. Pour cela, il suffit de chercher à tâtons, et de demander à l’oracle si les points qu’on considère sont dans le disque ou pas. Intuitivement, en quelques requêtes, on arrivera à trouver de tels points. (2) On demande à l’oracle de corriger les points A, B. Pour le point A, l’oracle retourne un point C qui est dans le disque, le plus près possible du point A. Clairement, ce point est à l’intersection entre le segment $[OA]$ et le cercle frontière du disque qu’on cherche. De même, soit D la correction de B. (3) A partir des points A, B et de leurs corrections C, D, il suffit de tracer les droites (AC) et (BD) avec une règle : elles se coupent en O. On peut ensuite tracer le cercle au compas. On obtient le rayon en mesurant la distance entre O et C.

Bref, il est facile d’apprendre des boules de $\mathbb{R}^2$ avec des CQ. Maintenant, quand on s’intéresse aux boules de mots, on peut penser que la démarche précédente est bonne et tenter de la reproduire.

4 Identification des boules avec des corrections

Dans cette section, nous proposons un algorithme apprenant les boules avec un nombre *linéaire* de requêtes de correction. D’une certaine manière, nous allons suivre l’algorithme précédent, visant à apprendre des disques du plan, en cherchant d’abord des points qui sont sur la frontière de la boule, puis en essayant d’en déduire le centre. Cependant, avec des boules de mots, le problème est plus compliqué.

Tout d’abord, et c’est rassurant, quand on demande à l’oracle de corriger un mot qui n’est pas dans une boule, on obtient toujours un mot du cercle qui la délimite. Autre-

ment dit, on va pouvoir circonscrire cette boule :

Exemple : Soit $B_2(bb)$ une boule. Les corrections possibles du mot $aaaa$ sont $\{aa, aab, aba, baa, aabb, abab, abba, baab, baba, bbaa\}$, tous à distance 2 de bb et $aaaa$. ◊

Cependant, contrairement à ce qui se passe dans le plan, un mot a généralement plusieurs corrections possibles. Par définition des CQ, l'oracle en choisira une au hasard. Mais aucune hypothèse ne pourra être faite sur cette correction. Nous considérerons donc toujours qu'il fait la "pire" réponse possible, selon le contexte. La proposition suivante caractérise l'ensemble de *toutes* les corrections possibles d'un mot :

Théorème 3
Soit $B_r(o)$ une boule et m un mot tel que $m \notin B_r(o)$. Alors un mot z est une correction possible de m ssi $(d(o,z) = r$ et $d(z,m) = d(o,m) - r)$.

Preuve : Posons $k = d(o,m)$ et considérons une dérivation de o en m de longueur minimale : $o \xrightarrow{k} m$. Comme $m \notin B_r(o)$, on a $k > r$, donc le long de la dérivation précédente, il existe un mot w_0 tel que $o \xrightarrow{r} w_0 \xrightarrow{k-r} m$. Définissons l'ensemble $W = \{w \in \Sigma^* \mid d(o,w) = r \text{ et } d(w,m) = k - r\}$. Clairement, $w_0 \in W$, donc $W \neq \emptyset$. De plus, $W \subseteq B_r(o)$. Enfin, notons Z l'ensemble des corrections possibles de m et montrons que $Z = W$. Soit $z \in Z$ et $w \in W$. Si $d(z,m) > d(w,m)$, alors w est un mot de $B_r(o)$ qui est strictement plus proche de m que z, ce qui contredit le fait que z soit une correction possible de m. Supposons que $d(z,m) < d(w,m)$. On a $d(o,m) \leq d(o,z) + d(z,m)$, donc $d(o,z) \geq d(o,m) - d(z,m) > d(o,m) - d(w,m)$. Or $d(o,m) = k$ et $d(w,m) = k - r$, donc $d(o,z) > r$, ce qui est impossible puisque $z \in Z \subseteq B_r(o)$. Donc $d(z,m) = d(w,m) = k - r$. En conséquence, tout mot $w \in W$ est à la même distance de m qu'une correction $z \in Z$. Donc $W \subseteq Z$. De plus, on a $d(o,m) \leq d(o,z) + d(z,m)$, donc $k \leq d(o,z) + k - r$, c'est-à-dire, $d(o,z) \geq r$. Comme $z \in B_r(o)$, on en déduit $d(o,z) = r$. Comme on a aussi $d(z,m) = k - r$, on en conclut $Z \subseteq W$. □

Remarque : On peut donner une interprétation géométrique du résultat précédent. Définissons le *segment* $[o,m] = \{w \in \Sigma^* \mid d(o,w) + d(w,m) = d(o,m)\}$ et de *cercle* $C_r(o) = \{w \in \Sigma^* \mid d(o,w) = r\}$. Le Théo. 3 stipule qu'un mot z est une correction possible de m ssi $z \in [o,m] \cap C_r(o)$. Le fait que m ait plusieurs corrections possibles montre que la géométrie de Σ^* est très différente de celle de $\mathbb{R}^2$. △

Le problème qui se pose ensuite est de construire le centre de la boule recherchée. Et là, malheureusement, le problème devient difficile, pour deux raisons. Intuitivement, d'abord, quand on munit Σ^* de la distance d'édition, on obtient un espace métrique mais on ne dispose pas d'espace vectoriel. Autrement dit, c'est comme si dans le plan, on disposait uniquement d'un compas, mais pas d'une règle … Formellement, ensuite, le problème est *difficile* (de la Higuera & Casacuberta, 2000) :

Théorème 4
Étant donné un ensemble fini de mots $W = \{w_1, \ldots, w_n\}$, trouver un mot $z \in \Sigma^*$ qui minimise $\sum_{w \in W} d(z, w)$, ou qui bien qui minimise $\max_{w \in W} d(z, w)$, sont des problèmes NP-difficiles.

En conséquence, nous n'aurons pas d'autres choix que d'étudier plus finement les boules, pour contourner cette difficulté en utilisant les requêtes de correction. A cet effet, on commence par distinguer certains mots des boules, ceux qui ont un nombre maximum de lettres :

Définition 5 (Frontière supérieure)
On appelle frontière supérieure d'une boule $B_r(o)$, notée $B_r^{max}(o)$, l'ensemble des mots de $B_r(o)$ qui sont de longueur maximum :

$$B_r^{max}(o) = \{z \in B_r(o) | \forall w \in B_r(o), |w| \leq |z|\}.$$

Exemple : Soit $\Sigma = \{a, b\}$ et $B_1(ba) = \{a, b, aa, ba, bb, aba, baa, bab, bba\}$. On a $B_1^{max}(ba) = \{aba, baa, bab, bba\}$. ◊

$B_r^{max}(o)$ est un ensemble remarquable car tous les mots qui le composent sont construits à partir du centre o en faisant r insertions. Aussi, disposant d'un mot $w \in B_r^{max}(o)$, il "suffit" de trouver les lettres qui ont été insérées dans o pour construire w, puis de les effacer pour retrouver o. Ce faisant, on contourne les problèmes posés par le Théo. 4, puisqu'il n'est plus nécessaire de *construire* le centre o de toute pièce :

Proposition 2
$w \in B_r^{max}(o)$ ssi $o \preceq w$ et $d(o, w) = |w| - |o| = r$.

Preuve : Supposons $o \preceq w$ et $d(o, w) = |w| - |o| = r$. Alors $w \in B_r(o)$. Soit w' un mot tel que $|w'| > |w|$. Alors, par la Prop. 1, $d(o, w') \geq |w'| - |o| > |w| - |o| = r$, donc $w' \notin B_r(o)$. Par conséquent, $w \in B_r^{max}(o)$. Réciproquement, soit $w \in B_r^{max}(o)$. Posons $k = d(o, w) \leq r$. Alors il existe une dérivation de réécriture $o \xrightarrow{k} w$. Si le long de cette dérivation, une opération d'effacement est utilisée, alors ne pas faire cet effacement, conduit à un mot w' de longueur $|w| + 1$ à distance $\leq k$, donc $w \notin B_r^{max}(o)$. De même, si une opération de substitution est utilisée, disons d'un a par un b, alors laisser le a et faire, en plus, l'insertion d'un b conduit de nouveau à un mot w', de longueur $|w| + 1$, à distance $\leq k$, donc $w \notin B_r^{max}(o)$. Aussi, seules des opérations d'insertion sont utilisées le long de la dérivation, donc $o \preceq w$. Enfin, si $k < r$, il suffit d'ajouter une lettre à w pour construire un mot w' tel que $|w'| = |w| + 1$, $o \preceq w'$ et $d(o, w') = |w'| - |o| = k + 1 \leq r$, donc $w \notin B_r^{max}(o)$. Aussi $k = r$. □

En outre, parmi les mots de $B_r^{max}(o)$, certains sont jouent un rôle spécial. En effet, soit $a \in \Sigma$ une lettre arbitraire. Alors le mot $a^r o = \underbrace{a \ldots a}_{r \text{ fois}} o$ est bien obtenu à partir de o en faisant r insertions, donc il appartient à $B_r^{max}(o)$. De plus, si on connaît r, il suffit d'effacer les r premiers a pour trouver o. Nous prétendons qu'avec des requêtes de correction, on peut obtenir $a^r o$ à partir de n'importe quel mot $w \in B_r^{max}(o)$ en faisant des échanges de lettres :

Algorithm 1 EXTRAIT_CENTRE

Require: un mot $w = x_1 \dots x_n \in B_r^{max}(o)$, le rayon r
Ensure: le centre o de la boule $B_r(o)$
1: $c \leftarrow 0; i \leftarrow 1; a \leftarrow x_n$
2: **while** $c < r$ **do**
3: Assume $w = x_1 \dots x_n$ and let $w' = ax_1 \dots x_{i-1}x_{i+1} \dots x_n$
4: **if** $CQ(w') =$ OUI **then** $w \leftarrow w'; c \leftarrow c + 1$ **end if**
5: $i \leftarrow i + 1$
6: **end while**
7: Assume $w = x_1 \dots x_n$ and **return** $x_{r+1} \dots x_n$

Exemple : $B_2^{max}(bb) = \{aabb, abab, abba, abbb, baab, baba, babb, bbaa, bbab, bbba, bbbb\}$. Sur le mot $baba$, sachant $r = 2$, l'algorithme EXTRAIT_CENTRE transforme, à chaque itération, la lettre d'indice i en un a qu'il place au début du mot, puis soumet le mot obtenu à l'oracle. c compte le nombre de fois où cette transformation est acceptée.

i	w	w'	$CQ(w')$	w change	c
1	$\underline{b}aba$	$aaba$	$baba$	non	0
2	$b\underline{a}ba$	$abba$	OUI	oui	1
3	$ab\underline{b}a$	$aabb$	OUI	oui	2

Quand $c = 2 = r$, EXTRAIT_CENTRE s'arrête sur le mot $aabb$ et retourne $o = bb$. $\Diamond$

Lemme 1
L'Algo. 1 est correct : pour tout mot $w \in B_r^{max}(o)$, sachant r, EXTRAIT_CENTRE *retourne o en temps polynomial, avec $\mathcal{O}(|o| + r)$ requêtes de correction.*

Preuve : Montrons que l'opération d'échange de lettres est correcte. Soient $w = x_1 \dots x_n \in B_r^{max}(o)$ et $w' = ax_1 \dots x_{i-1}x_{i+1} \dots x_n$ pour $i \in 1..n$. S'il existe une dérivation de réécriture $o \xrightarrow{r} w$ où la lettre x_i de w est le produit d'une insertion dans o, alors supprimer x_i et faire l'insertion d'un a en tête de o conduit à un mot w' qui satisfait $o \preceq w'$ et $|w'| = |w|$. Par la Prop. 1, on a $d(o, w') = |w'| - |c| = |w| - |c| = r$, donc par la Prop. 2, $w' \in B_r^{max}(o)$ et $CQ(w') =$ OUI. Par contre, s'il n'existe aucune dérivation de réécriture où x_i est le produit d'une insertion, alors x_i est une "vraie" lettre de o. Dans ce cas, supprimer x_i et ajouter un a en tête de o conduit à un mot w' tel que $o \not\preceq w'$. Par la Prop. 1, on a $d(o, w') > |w'| - |o|$ et comme $|w'| = |w|$, on a $d(o, w') > r$. Donc $w' \notin B_r^{max}(o)$ et $CQ(w') \neq$ OUI. $\square$

Bref, nous sommes maintenant en mesure de déduire le centre de cette boule $B_r(o)$ si on dispose de son rayon et d'un mot de sa frontière extérieure. Le lemme suivant donne des pistes (sachant qu'on ne connaît ni r, ni $|o|$ pour l'instant) :

Lemme 2
Supposons que l'alphabet contienne n lettres : $\Sigma = \{a_1, \dots, a_n\}$. Alors toute correction w du mot $m = (a_1 \dots a_n)^k$ avec $k \geq |o| + r$ est un mot de $B_r^{max}(o)$.

Preuve : Soit Z l'ensemble des corrections possibles de m. Montrons que $Z = B_r^{max}(o)$. Comme $m = (a_1 \dots a_n)^k$ avec $k \geq |o| + r$, on a $o \preceq m$, donc par la Prop. 1, $d(o,m) = |m| - |o|$. Par ailleurs, soit $w \in B_r^{max}(o)$. Par la Prop. 2, on a $o \preceq w$ et $d(o,w) = |w| - |o| = r$. De plus, comme $m = (a_1 \dots a_n)^k$ avec $k \geq |o| + r$, on a $w \preceq m$, donc par la Prop. 1, $d(w,m) = |m| - |w| = |m| - |o| - r = d(o,m) - r$. Comme $d(o,w) = r$ et $d(w,m) = d(o,m) - r$, on en déduit $B_r^{max}(o) \subseteq Z$ par le Théo. 3. Soit maintenant $z \in Z$. Par le Théo. 3, on a $d(o,z) = r$. Si $o \preceq z$, alors $z \in B_r^{max}(o)$ par le Théo. 2. Si $o \not\preceq z$, alors par la Prop. 1, $d(o,z) > |z| - |o|$, *i.e.*, $|z| < |o| + r$. Mais alors, $d(z,m) \geq |m| - |z| > |m| - |o| - r = d(w,m)$ pour tout $w \in B_r^{max}(o)$, ce qui contredit $z \in Z$. Donc $Z \subseteq B_r^{max}(o)$. □

En soumettant le mot $(a_1 \dots a_n)^k$ avec un entier k suffisamment grand, on peut trouver un mot de $B_r^{max}(o)$. Il nous reste à trouver un k intéressant, et pour ce faire, on utilise le lemme suivant. Il stipule que si on demande à l'oracle de corriger un mot fait d'une grande quantité de a, la correction qu'on obtient nous donne des indications précieuses sur le rayon et le nombre d'occurrences de a dans le centre :

Lemme 3
Considérons la boule $B_r(o)$, une lettre $a \in \Sigma$ et un entier $j \in \mathbb{N}$ tels que $a^j \notin B_r(o)$. Soit $w = CQ(a^j)$. Si $|w| < j$, alors $|w|_a = |o|_a + r$.

Preuve : Comme $a^j \notin B_r(o)$ et $w = CQ(a^j)$, par le Théo. 3, on a $d(o,w) = r$ et $d(w,a^j) = d(o,a^j) - r$. Or comme $|w| < |a^j|$, le calcul de $d(w,a^j)$ consiste à substituer toutes les lettres de w qui ne sont pas des a, puis à faire des insertions de a, donc $d(w,a^j) = j - |w|_a$. De même, $d(o,a^j) = j - |o|_a$. Par conséquent, $j - |w|_a = j - |o|_a - r$, c'est-à-dire, $|w|_a = |o|_a + r$. □

Supposons maintenant que l'alphabet soit $\Sigma = \{a_1, \dots, a_n\}$ et soient $j_1, \dots, j_n \in \mathbb{N}$ des grands entiers. Posons $k = \sum_{i=1}^n |CQ(a_i^{j_i})|_{a_i}$. Alors, par le Lemme 3, on a $k = \sum_{i=1}^n (|o|_{a_i} + r) = |o| + |\Sigma| \cdot r \geq |o| + r$. On peut donc utiliser k dans le Lemme 2 pour obtenir un mot $w = CQ\left((a_1 \dots a_n)^k\right) \in B_r^{max}(o)$. De plus, compte tenu de la Prop. 2, on a $|w| = |o| + r$. Aussi,

$$\left\{ \begin{array}{rcl} k & = & |o| + |\Sigma| \cdot r \\ |w| & = & |o| + r \end{array} \right. \Longrightarrow \left\{ \begin{array}{rcl} r & = & (k - |w|)/(|\Sigma| - 1) \\ |o| & = & (|\Sigma| \cdot |w| - k)/(|\Sigma| - 1) \end{array} \right.$$

Autrement dit, on trouve le rayon (et la longueur du centre).

Pour résumer, en supposant que $\Sigma = \{a_1, \dots, a_n\}$ et qu'on cherche à apprendre la boule $B_r(o)$. (1) Pour chaque lettre a_i, on demande la correction du mot a_i^j avec j suffisamment grand pour que la longueur de la correction soit $< j$; (2) on pose $k = \sum_{i=1}^n |CQ(a_i^{j_i})|_{a_i}$ et on demande une correction w du mot $m = (a_1 \dots a_n)^k$; (3) à partir de k et $|w|$, on en déduit r ; (4) on utilise EXTRAIT_CENTRE sur w et r pour construire c. Autrement dit, il est possible d'apprendre les boules avec des requêtes de correction (voir Algo 2).

Théorème 5
Les boules $B_r(o)$, définies à l'aide de la distance d'édition standard, sont identifiables en temps polynomial avec $\mathcal{O}(|\Sigma| + |o| + r)$ requêtes de correction.

Algorithm 2 Identification des boules avec des requêtes de correction

Require: l'alphabet $\Sigma = \{a_1, \ldots, a_n\}$
Ensure: la représentation (o, r) d'une boule $B_r(o)$
1: $j \leftarrow 1; k \leftarrow 0$
2: **for** $i = 1$ **to** n **do**
3: **while** $|CQ(a_i^j)| \geq j$ **do** $j \leftarrow 2 \cdot j$ **end while**
4: $k \leftarrow k + |CQ(a_i^j)|_{a_i}$
5: **end for**
6: $w \leftarrow CQ((a_1 a_2 \ldots a_n)^k)$
7: $r \leftarrow (k - |w|)/(|\Sigma| - 1)$
8: $o \leftarrow$ EXTRAIT_CENTRE(w, r)
9: **return** (o, r)

Preuve : On a l'apprenabilité comme conséquence des lemmes précédents. Concernant la complexité, trouver les corrections des mots a_i^j nécessite $\mathcal{O}(|\Sigma| + \log(|o| + r))$ requêtes de correction (lignes 2-5). L'utilisation de l'algorithme EXTRAIT_CENTRE nécessite $\mathcal{O}(|o| + r)$ requêtes de correction (ligne 8). D'où le résultat. □

Exemple : Considérons la boule $B_2(bb)$ sur l'alphabet $\Sigma = \{a, b\}$. Notre algorithme commence par chercher des corrections de a^j et b^j avec j suffisamment grand. On peut par exemple observer : $CQ(a) = \text{OUI}, CQ(a^2) = \text{OUI}, CQ(a^4) = aabb, CQ(a^8) = abba, CQ(b^8) = bbbb$. D'où $k = |abba|_a + |bbbb|_b = 2+4 = 6$. Ensuite, $CQ\left((ab)^6\right) = CQ(abababababab) = baba$, par exemple, et donc $r = (6-4)/(2-1) = 2$. Enfin EXTRAIT_CENTRE$(baba, 2)$ retourne bb. Donc l'Algo. 2 retourne $(bb, 2)$. ◊

5 Critique et expérimentation

Nous avons montré que les boules $B_r(o)$ ne sont pas identifiables dans le modèle d'Angluin (Théo. 2) mais qu'elles le sont en $\mathcal{O}(|\Sigma| + |o| + r)$ avec des requêtes de corrections uniquement (Théo. 5). En conséquence, ces requêtes sont novatrices, et nous pensons qu'elles méritent d'être étudiées. Pourtant, notre résultat est décevant.

En effet, si on reprend l'exemple de l'identification des disques de $\mathbb{R}^2$ de la Section 3, on s'aperçoit qu'il faut $\mathcal{O}(\log r)$ requêtes de correction pour apprendre. En comparaison, $\mathcal{O}(|o| + r)$ est un nombre excessif. En outre, la preuve du Théo. 5 montre que c'est l'utilisation de l'algorithme EXTRAIT_CENTRE qui coûte vraiment cher. Sans l'utiliser, on pourrait espérer une complexité logarithmique $\mathcal{O}(|\Sigma| + \log(|o| + r))$.

Cette complexité linéaire correspond toutefois au pire cas. Nous avons voulu vérifier qu'elle n'était pas atteinte en pratique. D'après le Théo. 5, le nombre de CQ nécessaires pour identifier $B_r(o)$ est $\mathcal{O}(|\Sigma| + |o| + r)$. Donc si on fixe $|o| + r = 1000$, le nombre de CQ nécessaires devrait être globalement constant. Aussi, on prend $\Sigma = \{a, b\}$, on fait varier r (donc $|o| = 1000 - r$) entre 0 et 1000 ; pour chaque valeur de r, on tire 10 centres $o \in \Sigma^{(1000-r)}$ et on fait la moyenne. On obtient la courbe de la Fig. 2.

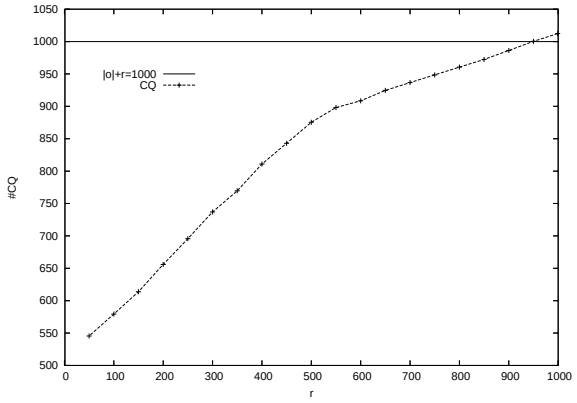

FIG. 2 – Nombre moyen de CQ nécessaires pour identifier des boules $B_r(o)$ tirées au hasard. On fixe $|o| + r = 1000$ et on fait varier r (donc $|o|$). Pour chaque r, on tire 10 centres de longueur $1000 - r$ et on fait la moyenne. Le nombre théorique de CQ est $|o| + r = 1000$. Le nombre effectif est clairement inférieur.

Clairement, on constate que le nombre de CQ n'est pas constant mais qu'il croît entre 500 et 1000 avec le rayon. Certains points de cette courbe sont faciles à justifier. Rappelons que pour trouver o, on utilise la fonction EXTRAIT_CENTRE qui parcourt un mot $w \in B_r^{max}(o)$; de plus, par la Prop. 2, $|w| = |o|+r = 1000$ et $o \preceq w$. Quand $r \simeq 1$ et $|o| \simeq 999$, EXTRAIT_CENTRE doit parcourir en moyenne la moitié du mot w pour trouver l'unique caractère inséré dans o, d'où un nombre de CQ $\simeq 500$. À l'inverse, quand $r \simeq 999$ et $|o| \simeq 1$, comme EXTRAIT_CENTRE doit faire r échanges de lettres pour trouver o, il utilise un nombre de CQ $\simeq 1000$.

Maintenant, la linéarité de la courbe, ainsi que le changement de pente qu'on observe lorsque $r \simeq |o| \simeq 500$, sont bien plus difficiles à justifier. Si on avait travaillé avec un grand alphabet Σ, alors EXTRAIT_CENTRE ferait en moyenne $1000 \cdot r/(r+1)$ CQ pour obtenir o. Mais ici, avec $|\Sigma| = 2$, o est un sous-mot de w de très nombreuses manières. Aussi, estimer de manière théorique le nombre de CQ nécessaires pour extraire o, nombre qui justifierait formellement la courbe obtenue, est un problème combinatoire d'une extrême complexité. On peut juste constater que ce nombre est souvent bien plus petit que $1000 \cdot r/(r+1) \simeq 1000$ dès que $100 \leq r \leq 900$.

Nous avons voulu vérifier que le comportement précédent restait identique pour des tailles différentes de représentations de boules. Aussi nous avons reproduit la même expérience, mais en fixant $|o| + r = t$ avec t variant entre 0 et 1000 (voir Fig. 3). La surface obtenue confirme les conclusions précédentes : le nombre de CQ moyens est largement inférieur à la théorie pour des grands centres et des petits rayons, et notre algorithme est efficace dans ces conditions.

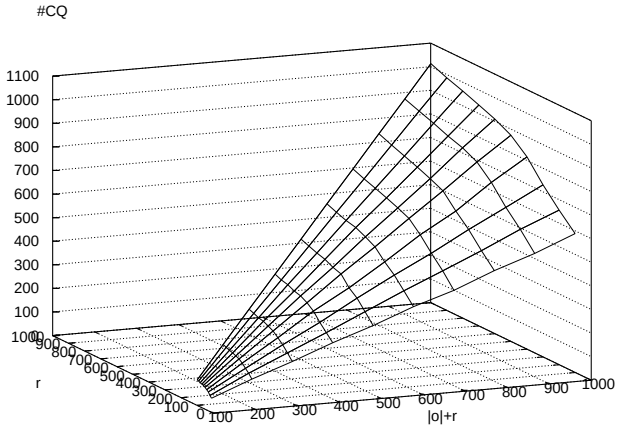

FIG. 3 – Nombre moyen de CQ nécessaires pour identifier 10 boules $B_r(o)$ lorsque $|o| + r = t$, en fonction de r et t.

6 Extension

Dans les parties précédentes, nous avons utilisé la distance d'édition non pondérée de Levenshtein (1965). Toutefois, en pratique, ses variantes pondérées sont souvent utilisées. Or nous pensons que les CQ restent pertinentes dans ce contexte, ce que nous souhaitons montrer ici. On suppose que les poids des opérations d'insertion et d'effacement restent à 1, mais que celui des substitutions est légèrement inférieur, disons $1 - \varepsilon$ avec $0 < \varepsilon < 1$. De plus, on suppose que l'alphabet contient au moins 3 lettres : $\Sigma = \{a_1, \ldots, a_n\}$ avec $n \geq 3$.

Sous ces conditions, on peut établir une version plus forte du Lemme 3. Celui-ci établissait qu'en demandant à l'oracle de corriger un mot, fait d'une grande quantité de a, n'appartenant pas à $B_r(o)$, la correction w obtenue vérifiait $|w|_a = |o|_a + r$. Ceci permettait ensuite de construire un mot de $B_r^{max}(o)$ (Lemme 2). Cette propriété reste vraie, mais en plus, le mot w lui-même appartient à $B_r^{max}(o)$:

Lemme 4
Considérons la boule $B_r(o)$, une lettre $a \in \Sigma$ et un entier $j \in \mathbb{N}$ tels que $a^j \notin B_r(o)$. Soit $w = CQ(a^j)$. Si $|w| < j$, alors $|w|_a = |o|_a + r$ et $w \in B_r^{max}(o)$.

Preuve : La première partie résulte du Lemme 3 qui reste vrai. Maintenant, si on considère une dérivation de réécriture de poids minimal $o \xrightarrow{r} w \xrightarrow{k-r} a^j$, cette dérivation fait intervenir $j - |o|_a$ insertions de a, et $|o| - |o|_a$ substitutions de lettres de o par des a. Or comme le poids des substitutions est plus faible que celui des insertions, celles-ci ont toute lieu le long de la dérivation $w \xrightarrow{k-r} a^j$ pour que w soit le plus proche possible a^j

(par définition des CQ). Du coup, il n'y a que des insertions de a le long de la dérivation $o \xrightarrow{r} w$. Donc $w \in B_r^{max}(o)$. □

En conséquence du Lemme 4, il n'est plus utile de construire un mot de $B_r^{max}(o)$ puisqu'on l'obtient directement. Mais il y a mieux... Notons $w_a = CQ(a^j)$ avec $|w_a| < j$. Compte tenu du Lemme 4, w_a est obtenue à partir de la o en faisant r insertions de a dans o. Aussi, les autres lettres de w_a sont les mêmes que celles de o, et elles interviennent dans le même ordre. Plus formellement, introduisons la fonction E_a qui efface toutes les occurrences de la lettre a dans un mot : (1) $E_a(\lambda) = \lambda$, (2) $E_a(a.u) = E_a(u)$ et (3) $E_a(b.u) = b.E_a(u)$ pour toute lettre $b \neq a$. Alors, pour tout $a \in \Sigma$, on a $E_a(w_a) = E_a(o)$.

Exemple : Soit $\Sigma = \{a, b, c\}$. On considère le boule $B_1(o)$ avec $o = baac$. Supposons qu'en corrigeant les mots a^j, b^j et c^j avec j suffisamment grand, on obtienne, comme corrections, $w_a = baaca$, $w_b = babac$ et $w_c = bacac$. On a bien $E_a(w_a) = E_a(o) = bc$, $E_b(w_b) = E_b(o) = aac$ et $E_c(w_c) = E_c(o) = baa$. ♢

Maintenant, si l'on dispose d'un alphabet avec au moins trois lettres a, b, c, la connaissance de $E_a(o)$, $E_b(o)$ et $E_c(o)$ permet facilement de construire o. Il suffit d'*aligner* les trois mots précédents :

$E_a(o)$	b	$\cdot$	$\cdot$	c
$E_b(o)$	$\cdot$	a	a	c
$E_c(o)$	b	a	a	$\cdot$
o	b	a	a	c

La procédure ALIGNE qui réalise cette construction n'utilise pas de CQ et s'exécute en temps $\mathcal{O}(|o|)$. Elle est donc clairement plus efficace qu'EXTRAIT_CENTRE. On obtient finalement l'Algo. 3 et le Théo. 6.

Algorithm 3 Identification des boules définies avec la distance d'édition pondérée

Require: l'alphabet $\Sigma = \{a_1, \ldots, a_n\}$ avec $n \geq 3$
Ensure: la représentation (o, r) d'une boule $B_r(o)$
1: $j \leftarrow 1$
2: **for** $i = 1$ to 3 **do**
3: **while** $|CQ(a_i^j)| \geq j$ **do** $j \leftarrow j \cdot 2$ **end while**
4: $w_i \leftarrow CQ(a_i^j); e_i \leftarrow E_{a_i}(w_i)$
5: **end for**
6: $o \leftarrow$ ALIGNE(e_1, e_2, e_3)
7: $r \leftarrow |w_1| - |o|$
8: **return** (o, r)

Théorème 6
Quand $|\Sigma| \geq 3$, les boules $B_r(o)$, définies à l'aide de la distance d'édition pondérée, sont identifiables avec $\mathcal{O}(\log(|o| + r))$ requêtes de correction, en temps $\mathcal{O}(|o| + r)$.

7 Conclusion

Dans ce travail, nous avons introduit un nouveau type de requêtes, dites de correction (CQ). Des oracles capables de répondre à celles-ci existent déjà de manière naturelle sur le web et sont faciles à implémenter. Nous nous sommes ensuite intéressés aux boules de mots définies avec la distance d'édition. Bien qu'elles semblent élémentaires, ces boules sont extrêmement complexes d'un point de vue combinatoire. Les étudier nous a permis de toucher du doigt la *géométrie* de Σ^*, très différente de la géométrie euclidienne. Enfin, nous avons montré qu'elles n'étaient pas apprenables avec les MQ et EQ d'Angluin, mais qu'elles l'étaient avec un nombre linéaire, voire logarithmique de CQ. Nous continuons de travailler dans ce domaine, en particulier pour développer une application.

Références

AMENGUAL J.-C., SANCHIS A., VIDAL E. & BENEDÍ J.-M. (2001). Language simplification through error-correcting and grammatical inference techniques. *Machine Learning Journal*, **44**(1), 143–159.

ANGLUIN D. (1987). Learning regular sets from queries and counterexamples. *Information and Computation*, **75**, 87–106.

ANGLUIN D. (1988). Queries and concept learning. *Machine Learning*, **2**, 319–342.

BECERRA-BONACHE L. & YOKOMORI T. (2004). Learning mild context-sensitiveness : Towards understanding children's language learning. In *Proc. of the 7th Int. Coll. in Grammatical Inference (ICGI'04)*, p. 53–64 : LNCS 3264.

BERNARD M., JANODET J.-C. & SEBBAN M. (2006). A discriminative model of stochastic edit distance in the form of a conditional transducer. In *Proc. of the 8th Int. Coll. in Grammatical Inference (ICGI'06)*, p. 240–252 : LNCS 4201.

CROCHEMORE M., HANCART C. & LECROQ T. (2001). *Algorithmique du texte*. Vuibert.

DE LA HIGUERA C. (2006). Data complexity issues in grammatical inference. In *Data Complexity in Pattern Recognition*, p. 153–172. Springer-Verlag.

DE LA HIGUERA C. & CASACUBERTA F. (2000). Topology of strings : median string is NP-complete. *Theoretical Computer Science*, **230**, 39–48.

DURBIN R., EDDY S., KROGH A. & MITCHISON G. (1998). *Biological sequence analysis*. Cambridge University Press.

KEARNS M. J. & LI M. (1993). Learning in the presence of malicious errors. *SIAM Journal of Computing*, **22**(4), 807–837.

LEVENSHTEIN V. I. (1965). Binary codes capable of correcting deletions, insertions, and reversals. *Doklady Akademii Nauk SSSR*, **163**(4), 845–848.

MICLET L. (1986). Grammatical inference. In *Syntactic and Structural Pattern Recognition - Theory and Applications*, p. 237–290. Word Scientific.

SCHULZ K. U. & MIHOV S. (2002). Fast string correction with levenshtein automata. *Int. Journal on Document Analysis and Recognition*, **5**(1), 67–85.

TANTINI F., DE LA HIGUERA C. & JANODET J. C. (2006). Identification in the limit of systematic-noisy languages. In *Proc. of the 8th Int. Coll. in Grammatical Inference (ICGI'06)*, p. 19–31 : LNCS 4201.

WAGNER R. & FISHER M. (1974). The string-to-string correction problem. *Journal of the ACM*, **21**, 168–178.