Gestion du braille abrégé

Eric Buist (buisteri@IRO.UMontreal.CA)

24 août 2001

Table des matières

1 Le braille

Inventé par Louis Braille, cet alphabet permet aux aveugles d'utiliser le sens du toucher pour la lecture. La version définitive de cet alphabet date de 1837. La cellule braille, représentant un caractère, est formée de six points actifs ou non, ce qui donne $2^6 = 64$ caractères. À chacune des lettres de l'alphabet correspond un agencement de points braille, ainsi qu'à chaque caractère accentué et signe de ponctuation. Malheureusement, bien que l'alphabet standard soit à peu près normalisé, les caractères accentués varient d'une langue à l'autre. Nous nous concentrerons ici sur le Français.

La représentation du braille est faite sur un papier spécial épais à l'aide d'une règle et d'un poinçon ou d'une imprimante braille. Toutefois, ce type de représentation n'est pas très commode pour afficher du braille à l'écran ou sur papier ordinaire. Dans le cas de documents au traitement de texte, tels Word ou WordPerfect, on utilise une fonte braille qui fait correspondre une cellule avec un caractère ASCII. Pour ce qui est de LaTeX, un package affichant les cellules comme des images est disponible sur le Web.

Puisque le formatage braille répond à des règles précises, le simple affichage de caractères braille ne suffit pas. Le logiciel Duxbury Braille Translation (DBT) constitue un traitement de texte spécialisé comportant des fonctionnalités braille. Il est en mesure d'importer des fichiers textes ASCII comprenant des commandes de formatage ou des documents issus d'un traitement de texte.

On peut aussi représenter une lettre braille par ses points actifs dans la cellule. Par exemple, la lettre `m` est représentée par les points 1-3-4. Le tableau 1 montre la position des six points. Le tableau 2 montre quant à lui la correspondance entre les différents caractères, les symboles braille et les caractères de la fonte braille utilisée dans la thèse de Carmen Fontaine. Dans ce tableau, certains caractères ne correspondant à aucun caractère alphabétique ont reçu des valeurs arbitraires.

1	4
2	5
3	6

Tab. 1 – Numéro des points braille dans la cellule

Série 1			Série 2			Série 3			Série 4		
a	a	⠁	k	k	⠅	u	u	⠥	â	*	⠡
b	b	⠃	l	l	⠇	v	v	⠧	ê	<	⠣
c	c	⠉	m	m	⠍	x	x	⠭	î	%	⠩
d	d	⠙	n	n	⠝	y	y	⠽	ô	?	⠹
e	e	⠑	o	o	⠕	z	z	⠵	û	:	⠱
f	f	⠋	p	p	⠏	ç	&	⠯	ë	$	⠫
g	g	⠛	q	q	⠟	é	=	⠿	ï	]	⠻
h	h	⠓	r	r	⠗	à	(	⠷	ü	|	⠳
i	i	⠊	s	s	⠎	è	!	⠮	œ	[	⠪
j	j	⠚	t	t	⠞	ù	)	⠾	w	w	⠺

Série 5			Série 6			Série 7			Série 8		
,	1	⠂	/	/	⠌	_	_	⠤			⠀
;	2	⠆	+	+	⠬	^	^	⠈			
:	3	⠒	#	#	⠼	`	`	⠘			
.	4	⠲	æ	>	⠜	ű	.	⠸			
?	5	⠢	’	’	⠄	”	"	⠐			
!	6	⠖	-	-	⠤	∂	;	⠰			
( )	7	⠶				SS	,	⠠			
«	8	⠦									
*	9	⠔									
»	0	⠴									

TAB. 2 – Alphabet braille classé par séries et incluant les symboles réels, leurs correspondants dans la fonte braille de la thèse de Carmen Fontaine et les agencements de points braille, respectivement.

Les caractères sont classés par séries de façon à faciliter l'apprentissage. La première série de dix caractères constitue la base pour toutes les autres. Les séries ne sont toutefois pas nécessaires pour appliquer les règles de l'abrégé. La lettre `w` est dissociée des autres pour des raisons historiques uniquement.

2 Le braille abrégé

L'écriture de textes en braille exige une grande quantité de papier et des coûts de production élevés. En effet, pour transcrire une page imprimée, environ trois pages braille sont nécessaires et il en coûte environ 5$ la page. De plus, le rythme de lecture en braille s'avère plus lent que celui atteint avec les yeux. C'est pourquoi un abrégé a vite été proposé afin d'économiser de l'espace et d'augmenter le rythme de la lecture. Actuellement, l'abrégé de 1955 est en vigueur, mais il exige environ quatre années d'apprentissage et comporte un grand nombre de règles et d'exceptions.

Il serait intéressant de réviser ce système afin de le simplifier et le rendre plus cohérent, ce qu'a fait Carmen Fontaine dans sa thèse de doctorat. Son système révisé comporte peu de règles et s'avère plus facile à apprendre que le système de 1955. Le problème consiste maintenant à déterminer le taux de compression du nouveau système par rapport à l'ancien, ce qui exige l'abrègement d'un certain nombre de textes. Un script est alors nécessaire pour effectuer la tâche de façon efficace.

Ce projet a pour objectif de pouvoir calculer le gain en caractères et en pages pour chacun des systèmes testés afin de savoir s'il vaut la peine de passer au système Fontaine pour ce qui est du gain d'espace. Finalement, on mettra au point un système d'abrégé complet selon les seules considérations mathématiques. Ce qui permettra de calculer la proximité des taux de compression des abrégés actuels et du taux de compression maximal.

3 Étapes de la conversion en abrégé

Pour abréger un texte, on le découpe tout d'abord en paragraphes. Une conversion primaire consistant à traiter les majuscules et les chiffres est tout d'abord effectuée. Les paragraphes sont ensuite découpés en mots et chaque mot est soumis à un traitement des locutions. Si aucune locution à abréger n'est trouvée, les mots sont abrégés selon les règles du système choisi.

Avant d'abréger un mot, on vérifie s'il doit l'être ou s'il ne faut pas plutôt l'écrire en intégral. L'abrègement d'un mot consiste tout d'abord à traiter les cas de mots dont on a déjà trouvé l'abréviation ou avec une abréviation prédéfinie dans le système d'abrégé utilisé. Si le mot ne comporte pas d'abréviation, on effectue une analyse de finales à retrancher puis une division en syllabes. On relève ensuite les séquences qui peuvent être abrégées, une séquence consistant en un groupe de lettres comportant une abréviation dans le système utilisé. Chaque séquence est testée afin de savoir quelles occurrences dans le mot seront retenues. On abrège ensuite chacune des syllabes individuellement après avoir déterminé

quelles séquences correspondent à chaque syllabe. On réunit finalement le mot à partir des syllabes, on ajoute la finale abrégée si elle a été trouvée et on effectue des corrections finales si nécessaires. Le mot et son abréviation sont ensuite mémorisés pour usage futur.

3.1 Conversion primaire

Certains programmes ou gestionnaires de clavier n'étant pas en mesure de saisir des majuscules accentuées, il arrive que de tels caractères soient écrits par c̆ suivi de la lettre en minuscule. Il faut convertir cette notation en véritable majuscule afin d'uniformiser les mots et simplifier le traitement des mots entièrement en majuscules. On réunit aussi les caractères Œ, œ, Æ et æ afin d'obtenir des abréviations uniformes selon qu'ils sont tapés ou non. Par exemple, le mot `coeur` deviendra `cœur` et `Oeuvre`, `Œuvre`. Cette conversion se doit de ne faire subir aucune modification aux commandes Duxbury qui constituent des mots précédés du signe de dollars ($).

On effectue ensuite la conversion des lettres majuscules en signe majuscule (ű) suivi de la lettre correspondante en minuscule. Par exemple, `A` deviendra ■`a`. Si un mot complet en majuscules est détecté, il sera converti en minuscules puis précédé de deux symboles majuscules. Par exemple, le mot `BONJOUR` deviendra ■■`bonjour` et non ■`b`■`o`■`n`■`j`■`o`■`u`■`r`. Si plus de trois mots consécutifs sont en majuscules, ils seront également traités selon le *Code de Transcription braille de l'imprimé*. Par exemple, `VOICI UN GROUPE DE MOTS EN MAJUSCULES` deviendra, après conversion, `:`■■`voici un groupe de mots en `■■`majuscules`. Des subtilités apparaissent avec la gestion des caractères spéciaux tels le trait d'union, le point, la virgule, ...

Pour ce qui est des chiffres, on les remplacera par un signe numérique (#) suivi de la lettre correspondant au chiffre. Le nombre `23` deviendra par exemple `#bc`. Il faut également gérer correctement la représentation des intervalles, des dates et des heures et ne pas précéder chacun des groupes de chiffres d'un dièse. Par exemple, on écrira `6/7/2001` `#f/g/bjja`.

3.2 Traitement des paragraphes

Ce traitement consiste à séparer la ligne en mots afin de pouvoir abréger les mots individuellement. On utilise pour ce faire des délimiteurs bien définis. Puisque les délimiteurs s'avèrent nombreux, il est plus commode de les définir comme ne constituant pas un caractère alphanumérique. L'alphabet utilisé comprend l'ensemble des lettres usuelles, majuscules et minuscules, ainsi que les lettres accentuées, les chiffres et les caractères `#`, `$` et `+`.

Système Fontaine

Le traitement des locutions n'est pas utilisé, car ce système n'admet pas d'abréviations de locutions.

Système de 1955

On vérifie si le mot commence une locution et si tel est le cas, on regarde si les mots suivants la complètent. Si tel est le cas, la locution est abrégée et on passe aux mots la suivant, sinon les mots sont abrégés comme si de rien n'était. Le système comporte un total de quarante-trois locutions de ce type. Pour les locutions se terminant par `que`, on ajoute un équivalent terminant par `qu` afin de gérer correctement les élisions.

Exemple: `parce qu'il→p`ce q'i→p_q'i→p_qu'i` [1]

Système de 1923

On effectue également le traitement des locutions dans ce système. Le nombre de locutions se limite ici à vingt-trois.

3.3 Mots ayant un ou plusieurs caractères abréviatifs

Si le mot comporte un seul caractère, il ne peut en aucun cas être abrégé et le traitement s'arrête là. Certains mots s'avèrent assez fréquents pour nécessiter une abréviation spécifique. Il faut détecter ces cas simples avant de passer aux étapes suivantes, car si le mot s'abrège ainsi, le processus est terminé pour ce mot et on peut passer au suivant. Une liste de mots doit être fournie au script, et cela dans les trois systèmes, car aucune logique ne permettrait d'abréger de façon cohérente ces mots dont il faut apprendre les abréviations par cœur.

Exemple: `elle→e→z→z`

Pour gérer correctement et sans confusion les formes féminines et/ou plurielles, des mots sont simplement ajoutés à la liste de mots. Par exemple, on ajoutera une abréviation pour `elles` dans les trois systèmes et elle pourra respecter la règle.

Exemple: `elles→es→zs→zs`

Système Fontaine

Le système comprend quarante mots ayant une abréviation d'un seul caractère, dix-sept avec une abréviation de plus d'un caractères et douze mots pouvant être abrégés seuls ou en tant que séquence de lettres dans un mot. Si on regroupe tous ces mots et qu'on ajoute les formes féminines et plurielles lorsque cela s'applique, on obtient un total de cent dix mots.

Système 1955

Avant d'abréger, il faut vérifier si le mot en question doit être abrégé ou non. En mode Duxbury, si un mot est précédé d'un astérisque (*), il sera écrit en intégral et l'astérisque sera supprimé. Sans le mode Duxbury, tous les mots sont abrégés. Le système comporte quarante-huit mots avec une abréviation d'un caractère ainsi que huit cent deux mots avec

[1] On écrit le mot non abrégé, son abréviation dans le système Fontaine, celle dans le système de 1955 et celle dans le système de 1923, respectivement.

une abréviation de plus d'un caractères. Le regroupement de ces mots ainsi que l'ajout des formes féminines et plurielles produit une liste de mille quatre cent cinquante mots.

Système 1923

Avant d'abréger, on vérifie si le mot contient les lettres `k`, `ë`, `ï`, `ü`, `w` ou `æ`. En mode Duxbury, la même règle que pour le système de 1955 s'applique. On trouve ici quarante-quatre mots ayant une abréviation d'un seul caractères et cent quatre-vingt-dix-neuf mots avec une abréviation de plus d'un caractères. Le regroupement de ces mots et l'ajout des formes féminines et plurielles, ainsi que des cas d'exception de mots devant s'écrire en intégral, comporte trois cent soixante-six mots. Ces cas d'exception regroupent les mots `an`, `eu`, `or` et `bleu`, ainsi que l'interjection `ô`.

3.4 Analyse des finales

Système Fontaine

Dans ce cas, l'analyse des finales est un peu plus élaborée. On recherche tout d'abord, à la fin du mot, une expression du type `consonnes1-i-consonnes2-ité`. Prenons l'exemple du mot `électricité`. La recherche donnera `ctr` pour le premier groupe de consonnes et `c` pour le second. Il faut vérifier si `c` constitue une consonne seule ou une séquence de consonnes ayant une abréviation dans le système Fontaine et effectuer le même traitement pour le cas de `ctr`. Dans ce dernier cas, prendre seulement le sous-ensemble `tr` permet d'obtenir une séquence qui peut s'abréger par », on doit abréger la finale `tricité` par »`ct` et non la finale `ricité` par `rct`.

Exemple: `électricité`→`élec »ct`→`élec »icité`→`élec »icité`

On vérifie ensuite la présence d'une finale `que` qui sera retirée du mot et remplacée, à la fin du traitement, par `q`.

Une règle est finalement appliquée pour la finale `consonnes+ement`. Par exemple, dans le mot `escarpement`, on détectera la finale `pement` qu'on abrègera `pm`. Lorsqu'une finale est trouvée, elle est supprimée du mot et sa version abrégée est ajoutée à la fin du traitement.

Exemple: `escarpement`→`esc`pm`→`ûcarpemê`→`esc`pem'`

Avec cet algorithme, un problème surgit dans le cas des finales plurielles. Si on tente d'abréger le mot `battements`, le système ne détectera pas de finale du type `consonnes+ement` en fin de mot en raison du `s` marquant le pluriel. Il faut donc traiter les cas des finales `consonnes-i-consonnes-ités`, `ques` et `consones-ements`.

Exemple: `battements`→`bawms`→`battms`→`battem's`

Système 1955

Dans ce système, le nombre de finales est déterminé au départ, puisque les finales ne sont pas définies par des règles. Une finale est considérée comme une séquence de lettres qui peut être abrégée mais sous la restriction *en fin de mot seulement*. On dénombre un total de vingt

finales dont certaines comportent plusieurs syllabes. Un équivalent pluriel avec `s` final doit également être prévu.

Système 1923

Ce système ne comporte qu'une seule finale, la finale `ment` qui est ajoutée parmi les séquences avec restriction *en fin de mots seulement*. On ajoutera également la finale `ments` afin de traiter le cas des pluriels.

3.5 Division syllabique

Puisque les deux systèmes exigent que les séquences de lettres ne soient pas abrégées si elles ne se trouvent pas dans une même syllabe, diviser le mot en syllabes s'avère impératif. Cette division doit être la plus fidèle possible à celle rencontrée en grammaire française. Puisque les deux abrégés comprennent des consonnes doubles dans la liste des séquences, les groupes `ll`, `ss` et `tt` doivent être ajoutés parmi les inséparables, c'est-à-dire qu'il n'y aura pas de coupure syllabique entre ces groupes.

3.6 Recherche des séquences

La détection préliminaire des séquences consiste à dresser la liste des séquences se trouvant dans le mot à abréger et respectant les contraintes dans le cas du système de 1955. Contrairement à la syllabe qui constitue une division grammaticale d'un mot, la séquence constitue un groupe de lettres auquel est associée une abréviation dans un système donné. Toutefois, cette structure ne tient pas compte des occurrences multiples d'une séquence dans un mot. Il faut pouvoir vérifier si chaque occurrence respecte les contraintes dans le cas du système de 1955, ce qui complique l'algorithme de substitution. Par exemple, dans le mot `entendement`, on retrouve deux occurrences de la séquence `ent`, dont une seule en finale qui respecte la restriction de la séquence dans le cas du système de 1955.

Exemple: `entendement` → `?t ?dm` → `?t ?demê` → `?t ?dem'`

Système Fontaine

Il n'y a ici aucune restriction quant à l'abrègement des séquences. Si une séquence pouvant être abrégée est détectée à l'intérieur du mot, elle est répertoriée. Le système comporte quarante-deux séquences de ce type. La séquence `ex` constitue toutefois un cas particulier, car la notion de syllabes est particulière. Il est plus simple de ne l'abréger que si elle se trouve devant une consonne.

Système 1955

Il faut ici traiter les restrictions qui s'appliquent sur la plupart des séquences. Par exemple, `ll` ne se trouve qu'entre des voyelles et `an` n'est pas admise en fin de mot. Une séquence est répertoriée si au moins une occurrence respecte les restrictions. Le système comporte

quarante-et-une séquences et sept finales qui se répartissent sur plus d'une syllabe grammaticale.

Dans le cas des séquences avec la restriction *en fin de mots seulement*, on ajoute une séquence avec la lettre `s` à la fin afin de pouvoir traiter les pluriels correctement. Par exemple, dans le mot `mouvements`, la finale `vement` ne permet pas d'abréger correctement le mot, il faut ajouter la séquence `vements` abrégée `vms` afin de traiter correctement le pluriel.

Exemple: `mouvements`→`müvms`→`müvms`→`müvem's`

La séquence `ient` est ajoutée afin de traiter le cas particulier des mots se terminant par cette dernière, de même pour la séquence `ess` en début de mot qui fait exception à la règle et le cas de la séquence `ieur` qui cause ambiguïté.

Exemple: `déficient`→`déficij`→`défic.t`→`défic.t`

Exemple: `essai`→`eè/`→`ûs/`→`ess/`

Exemple: `ingénieur`→`*géni■`→`*génSSr`→`*géni■`

Les séquences `aient` et `oient` sont également ajoutées afin de traiter le cas des verbes pluriels. Sans ces séquences, `ient` serait détectée et abrégée `.t`. Cette finale serait trouvée et remplacée par `iê` si le mot était un verbe. L'ajout de la séquence permet d'abréger correctement les verbes à l'imparfait et au conditionnel.

Exemple: `aimaient`→`/m/j`→`/m/ê`→`/ma.t`

Système 1923

Les séquences ne sont soumises à aucune restriction, sauf le cas de la finale. Le système comporte vingt-huit séquences ainsi que la finale `ment` et `ments`.

3.7 Traitement individuel des séquences

Pour chaque séquence répertoriée, on établit la liste des occurrences de la séquence dans le mot que l'on représente par le début du mot jusqu'à la séquence testée, exclusivement. Par exemple, pour le mot `entendement`, la séquence `en` sera détectée et ses trois occurrences seront représentées par (chaîne vide, début du mot), `ent` et `entendem`. Pour être admise, une occurrence d'une séquence donnée doit se trouver toute entière dans une syllabe. Si la séquence testée contient un groupe inséparable ou si elle se trouve en début ou en fin de mot, toute division syllabique pouvant briser la séquence sera supprimée.

Lorsque toutes les séquences ont été testées, la division syllabique du mot est enfin définitive. On peut alors déterminer, pour chaque syllabe, quelle séquence est trouvée et qu'il faut abréger. Cet algorithme permet de gérer les cas d'occurrences multiples et peut appliquer les règles pour chacune d'elles dans le système de 1955. Il ne perturbe pas non plus le fonctionnement des systèmes de Fontaine et de 1923 qui définissent simplement des séquences sans restrictions.

Système Fontaine

La finale est retirée du mot, elle est donc déjà traitée. Seuls les cas des inséparables provoquent des exceptions dans la division syllabique.

Système 1955

Les finales font exception à la règle des syllabees. En effet, la division syllabique coupera le mot `perméabilité` `perméa-bi-lité`, ce qui coupe la finale et l'empêche de s'abréger. On supprimera donc la division entre `bi` et `lité` afin d'obtenir le découpage artificiel `perméa-bilité` qui contient la finale dans une seule syllabe.

Exemple: `perméabilité`→`p« méablt`→`p« méablt`→`p« méabilité`

Un problème similaire survient pour les séquences en début de mot seulement. Par exemple, dans le mot `inanimé`, qui se sépare `i-na-ni-mé`, le système de 1955 exige l'abrègement de la séquence `in` tandis que le système Fontaine l'interdit. Pour demeurer général, on ne peut pas définir une nouvelle séparation pour ce mot particulier et on ajoute donc les séquences en début de mot seulement comme des inséparables, considérant qu'elles constituent des préfixes. Ce qui résoud les cas de mots débutant par `ines`, `intrans`, `ess`, ...

Exemple: `inanimé`→`inanimé`→`*animé`→`inanimé`

Système 1923

Le traitement ressemble beaucoup à celui de Fontaine puisque seule la séquence `ment`, qui tient en une syllabe, est admise en fin de mots seulement.

3.8 Traitement de chaque syllabe

Pour chaque syllabe, on a trouvé un groupe de séquences à substituer. Malheureusement, il arrive souvent que l'on trouve plus d'une séquence à substituer dans une même syllabe. Par exemple, dans le cas du mot `main`, on trouvera la séquence `ai` et `in`. Une séquence doit être choisie pour la substitution, car les deux se chevauchent. Un algorithme de résolution des ambiguïtés doit être utilisé à ce moment, et cela quelque soit le système d'abrégé. L'algorithme doit s'avérer suffisamment général pour s'adapter à tous systèmes.

Résolution des ambiguïtés

Une première étape de cette résolution consiste à trier les séquences par ordre de longueur puis par position dans le mot. Ainsi, en cas de mauvaise sélection des séquences à remplacer, la plus longue séquence sera substituée la première et les autres ne pourront pas être remplacées. Le tri par position des séquences de longueurs égales dans la syllabe permet d'obtenir un comportement plus déterministe que de laisser les séquences dans l'ordre de détection qui peut dépendre du langage d'implantation de l'algorithme. L'algorithme prend ensuite chaque séquence deux à deux afin d'effectuer des tests sur chaque couple. Si les séquences

ne se chevauchent pas, on les conservera toutes les deux, sinon on effectue des tests afin de supprimer ou d'éliminer l'une des deux séquences du couple en cours de test.

La suppression consiste à retirer ls séquence du tableau afin qu'elle ne soit pas substituée du tout. L'élimination consiste à réduire la priorité de substitution de la séquence en la déplaçant en dernière position dans le tableau. Ces tests ont été déterminés de façon expérimentale, ils ne sont pas inspirés de règles connues.

- **Test d'inclusion :** Si une séquence se trouve entièrement contenue dans une autre, alors elle est supprimée. Par exemple, `en` est entièrement contenue dans `ent`. Toutefois, si après le retrait de la syllabe de la séquence longue, des occurrences de la séquence courte demeurent dans la syllabe, aucune séquence ne sera supprimée. La séquence la plus courte sera toutefois éliminée pour ne pas être substituée avant la plus longue.
- **Test de longueur :** Si la longueur d'une séquence est plus longue qu'une autre, la plus courte est éliminée, maximisant ici le gain d'espace.
- **Précédence du nombre de consonnes :** Ls séquence qui a le plus petit nombre de consonnes sera éliminée. Ce test est dû au fait que les séquences avec consonnes marquent plus probablement un graphème et doivent être abrégées.
- **Doubles consonnes :** Si les deux séquences du couple comportent un même nombre de consonnes, on privilégie celle qui contient un groupe de consonnes doubles, telles `ll` ou `ss`, par exemple. Normalement, de telles séquences sont divisées par la coupure syllabique, mais on admet leur abrègement malgré ce fait. Il est donc logique d'abréger la double consonne.
- Si aucun test n'a permis d'éliminer une séquence, on les conserve toutes deux malgré la présence de l'ambiguïté. L'ordre des séquences obtenu par le tri déterminera l'ordre de la substitution.

Système 1955

Un problème survient avec les caractères consécutifs identiques. Ainsi, dans le mot `drôle`, `dr` sera détectée, l'occurrence sera représentée par la chaîne vide et puisque `dr` s'abrège `ô`, le résultat sera `ôôle`.

Exemple: `drôle`→`ôôle`→`drôle`→`drôle`

Afin de détecter ces cas et les traiter correctement, on calcule le nombre de caractères identiques consécutifs dans le mot avant et après la subsitution de la syllabe. Si ce nombre a augmenté, il faut annuler la substitution précédente. Cet algorithme permet non seulement de traiter le cas du mot précédent mais aussi le cas plus complexe du mot `bruine`. Dans ce mot, les séquences `br` et `ui` s'abrègent de façon identique et le système de 1955 interdit l'abréviation de la seconde séquence.

Exemple: `bruine`→ `;SSne`→ `;uine`→ `;uine`

3.9 Traitement final du mot

Avant de réunir les syllabes, la division est mémorisée afin de supporter la coupure des mots abrégés dans le formateur. Les syllabes sont jointes et reforment le mot abrégé. On

ajoute également le mot à la table de mots abrégés afin qu'il soit traité plus rapidement s'il est de nouveau rencontré dans le texte. Ce procédé peut être appliqué en raison de l'unicité de l'abréviation d'un mot qui est due au fait que les règles ne dépendent pas du contexte dans lequel le mot se trouve.

Système Fontaine

On ajoute la finale qui avait été retranchée au début du traitement pour terminer le processus.

Système 1955

On applique une correction finale au mot s'il est formé uniquement de signes inférieurs (symboles dont les points braille 1 et 4 ne sont pas actifs) et comporte plus d'un caractères. Cette correction consiste à remplacer le premier caractère, un signe inférieur qui est non alphabétique, par la séquence qui, abrégée, aurait donné ce caractère. Dans ce cas particulier, il est plus clair d'illustrer avec des cellules braille plutôt que des caractères.

Exemple: `entrer`→ ⠢⠗⠑ → ⠔⠞⠗⠑ → ⠢⠞⠗⠑

Si l'abréviation se termine par `.t`, ce qui équivaut à la séquence `ient`, on vérifie à l'aide d'une table si le mot constitue un verbe à la troisième personne du pluriel. Si tel est le cas, on remplace la finale `.t` par *iê*. La table des verbes pluriels qui ce terminent par `ient` comporte cent quatorze verbes et excluent ceux qui se terminent en `aient` puisqu'ils sont traités par le test d'inclusion.

Exemple: `oublient`→`ü^ij`→`ü^iê`→`ü^.t`

Exemple: `déficient`→`déficij`→`défic.t`→`défic.t`

Système 1923

Une règle semblable au cas du système de 1955 doit être appliquée pour les signes inférieurs employés seuls. Toutefois, au lieu de désabréger le premier caractère, le script devrait sélectionner un caractère le plus avantageux. Toutefois, aucun algorithme n'a pu être trouvé pour implanter cette règle complètement.

Si la finale `m'` est trouvée, ce qui correspond à la finale `ment` dans le mot initial, on vérifie à l'aide d'une table si le mot constitue un verbe à la troisième personne du pluriel. La liste des verbes comportent soixante-quinze éléments.

3.10 Conversion finale

Lorsque tous les mots du paragraphe sont convertis, on a une chaîne ASCII en braille abrégé qu'il est ensuite possible, après formatage selon les normes du *Code de transcription braille de l'imprimé*, d'afficher telle quelle à l'écran, de convertir simplement pour la rendre affichable sous LaTeX, en noir ou en braille, d'en faire du braille ASCII ou LaTeX ou finalement, de la convertir dans la fonte braille. Si cette option de conversion est exploitée, chaque

caractère est converti de façon à ce que la chaîne abrégée, si on l'affichait avec la fonte braille, donne les bons points braille.

4 Exemple complet d'abrègement

Pour des raisons d'espace, les délimiteurs de ponctuation sont joints aux mots, mais le script traite chaque délimiteur comme un mot à part entière. Puisque ce « mot » ne compte qu'un caractère, aucune tentative d'abrègement ne sera effectuée. Les tableaux suivants présentent les étapes de l'abrégé ainsi que le nombre de caractères de la phrase au cours du procédé.

4.1 Système Fontaine

Phrase initiale	Je	suis	terriblement	réjoui !	28
Conversion primaire (3.1)	ŭje				29
Mots avec abréviation (3.3)	ŭj				28
Analyse des finales (3.4)			terri blement→ˆm		21
Division syllabique (3.5)		suis	ter-ri	ré-joui !	
Séquences (3.6)		ui	er	ou, ui	
Occurrences (3.7)		ui : s	er : t	ou : réj ui : réjo	
Traitement des syllabes (3.8)		suis : ui	ter : er ri :	ré : joui : ou, ui	
Ambiguïtés (3.8)		suis : ui→SS	ter : er→« ri :	ré : joui : ou→ü, ui[2]	18
Traitement final (3.9)	ŭj	sSSs	t« riˆm	réjüi !	20

[2]Il n'y a aucune règle pour éliminer une séquence dans ce cas particulier. Seul l'odre des séquences résoud l'ambiguïté.

4.2 Système 1955

Phrase initiale	Je	suis	terriblement	réjoui !	28
Conversion primaire (3.1)	űje				29
Mots avec abréviation (3.3)	űj				28
Division syllabique (3.5)		suis	ter-ri-ble-ment	ré-joui !	
Séquences (3.6)		ui	bl,ent er,en	ou, ui	
Occurrences (3.7)		ui : s	bl : terri ent : terriblem er : t en : terriblem	ou : réj ui : réjo	
Traitement des syllabes (3.8)		suis : ui	ter : er ri : ble : bl ment : en, ent	ré : joui : ou, ui	
Ambiguïtés (3.8)		suis : ui→ ;	ter : er→« ri : ble : bl→ˆ ment : ent→ê, en[3]	ré : joui : ou→ü, ui	22
Traitement final (3.9)	űj	s ;s	t« riˆemê	réjüi !	22

[3]Le test d'inclusion permet d'éliminer la séquence qui cause ambiguïté.

4.3 Système 1923

Phrase initiale	Je	suis	terriblement	réjoui !	28
Conversion primaire (3.1)	ŭje				29
Mots avec abréviation (3.3)	ŭj				28
Division syllabique (3.5)		suis	ter-ri-ble-ment	ré-joui !	
Séquences (3.6)			er, bl, ment	ou	
			em, en		
Occurrences (3.7)			er : t	ou : réj	
			bl : terri		
			ment : terrible		
			em : terribl		
			en : terriblem		
Traitement des syllabes (3.8)		suis :	ter : er	ré :	
			ri :	joui : ou	
			ble : bl		
			ment : ment,ent,em		
Ambiguïtés (3.8)		suis :	ter : er→«	ré :	23
			ri :	joui : ou→ü	
			ble : bl→ˆ		
			ment : ment→m',en,em[4]		
Traitement final (3.9)	ŭj	suis	t« riˆem'	réjüi !	23

5 Gain d'espace avec l'abrègement

À partir du script, il est possible de calculer le gain d'espace en caractères et en pages pour chacun des abrégés. Pour calculer le gain en pourcentage, la formule suivante est utilisée.

$$g = 100\left(1 - \frac{newval}{oldval}\right)$$

newval constitue la valeur obtenue de l'abrégé et *oldval*, celle du texte original. Suite au calcul, la valeur de g est arrondie à l'unité pour plus de lisibilité. Pour compter les caractères, on utilise la taille du fichier abrégé ou ayant subi la conversion primaire selon le cas tandis que pour le cas des pages, on utilise le module de formatage braille pour obtenir une mise en page. Un tableau exhaustif des résultats pour les textes analysés est disponible à l'adresse `http://www.ericbuist.com/fontabui/Fontabui.xls`.

On peut ici se rendre compte, d'après le tableau 3, que les gains en caractères sont très près des gains en pages. Ce qui signifie que le formatage ne constitue pas une étape nécessaire pour le calcul des gains d'espace. Ces résultats ne sont calculés que pour les trois systèmes avec tous leurs paramètres. Il serait davantage intéressant d'observer le comportement des

[4] L'élimination de cette séquence est effectuée par le test de longueur des séquences.

	Total en car.	Gain en car	Total en pages	Gain en pages
Original	818240	0%	1312	0%
Fontaine	622383	24%	999	24%
1955	586852	28%	949	28%
1923	635821	22%	1027	22%

Tab. 3 – Gain d'espace moyen pour les huit textes analysés.

systèmes si on n'abrégeait pas les mots (**m**), les séquences (**s**), les locutions (**l**) ou les finales (**f**). La figure 1 présente une telle analyse. Les gains sont obtenus par une moyenne sur huit textes, voir annexe C pour plus d'informations.

Ces résultats démontrent clairement que les locutions n'ont aucun effet pour les systèmes où elles s'appliquent. Ce résultat est dû à la faible fréquence de tels graoupes de mots et à la présence d'abréviations de mots et/ou de séquences à l'intérieur même des locutions. Par exemple, dans le cas de `peu à peu` en 1955, on abrège `p_à_p`. Si on n'abrège pas les locutions, on obtiendra `p■ à p■`, ce qui constitue un gain de deux caractères uniquement.

Les finales Fontaine n'influent que peu non plus sur le gain d'espace pour les mêmes raisons. Par contre, les mots et les séquences ont un impact majeur bien que semblable.

6 Limitations de la procédure d'abrègement

Système Fontaine

La procédure précédente permet d'abréger la plupart des mots correctement en respectant les règles de l'abrégé du système Fontaine. Des problèmes pourraient survenir dans le cas de séquences de lettres qui se chevauchent, provoquant une ambiguïté. Toutefois, en raison du test d'inclusion effectué au début de l'algorithme de résolution des ambiguïtés, il est possible de résoudre certains problèmes en ajoutant à la liste des séquences pouvant s'abréger des séquences plus longues qui permettront de gérer l'exception correctement. Par exemple, dans le mot `battre`, les séquences `tt` et `tr` seront trouvées et `tt` sera sélectionnée par priorité des doubles consonnes. L'ajout de `ttr` associée avec son abréviation correcte, `t` » permet de lever l'ambiguïté.

D'autres problèmes émergent en raison de l'algorithme de coupure de mots qui s'avère davantage un système de division typographique que syllabique. Par exemple, par défaut, avec les règles françaises, le mot `naine` n'est pas séparé tandis qu'on devrait couper entre le `i` et le `n`. L'ajout d'une règle de coupure permet toutefois de résoudre le problème, mais des cas inconnus peuvent surgir à tout moment. Ce type de problème se caractérise par un abrègement d'une séquence qui est normalement interdit.

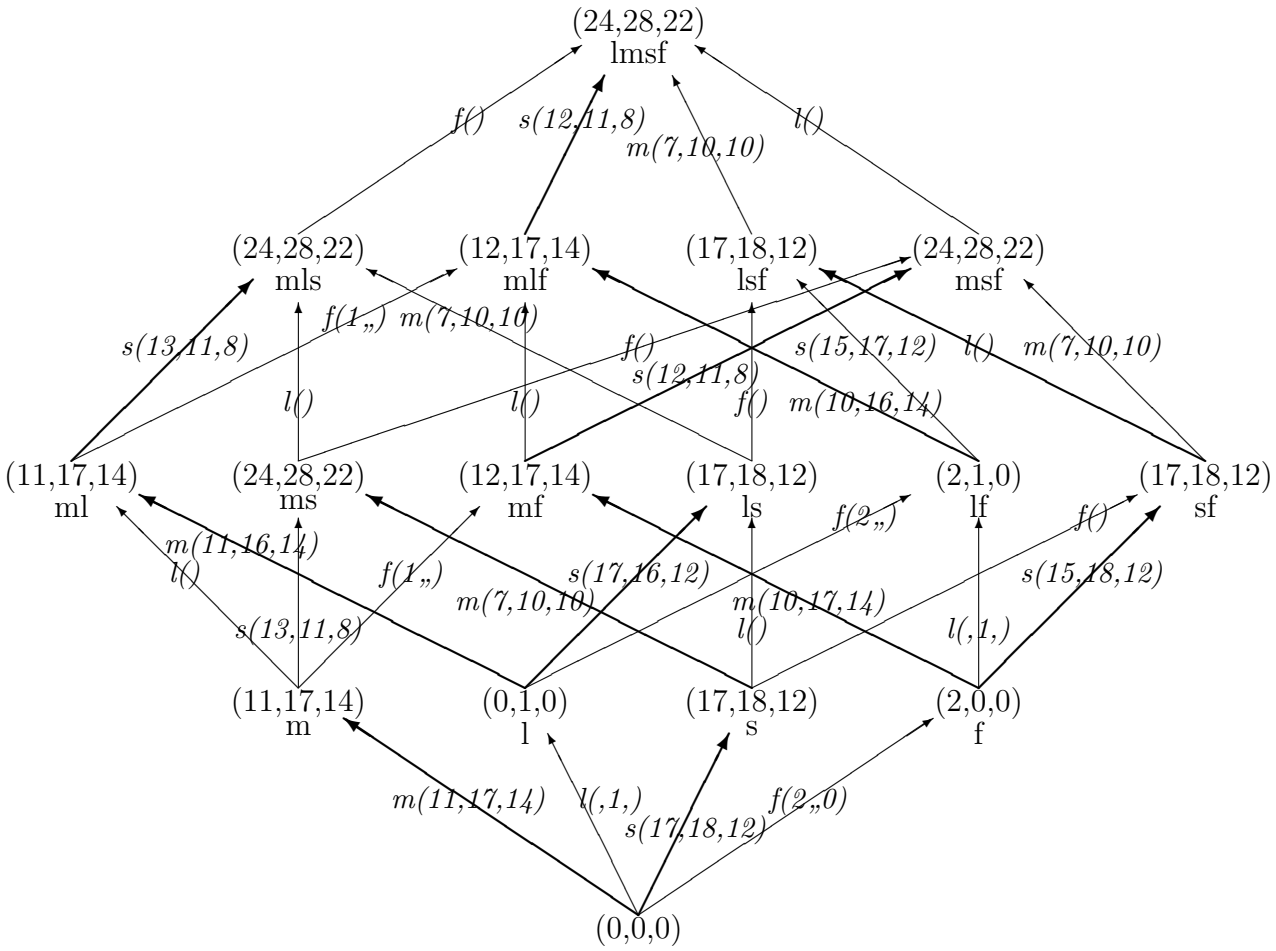

FIG. 1 – Présentation graphique des résultats des calculs de gain d'espace avec différents paramètres des systèmes. Les gains en caractères sont affichés sous forme de pourcentages et placés dans l'ordre suivant : système Fontaine, système 1955 et système 1923.

Système 1955

En plus des problèmes d'ambiguïtés du système Fontaine, les règles du système de 1955 provoquent d'autres erreurs et limitations dans la procédure. Le script ne pourra pas appliquer la règle des noms propres en intégral, car il ne peut distinguer un nom propre d'un nom de peuple ou d'un nom commun accidentellement employé comme un nom propre. Avec le mode Duxbury, on peut toutefois spécifier, en le précédant d'un astérisque, qu'un mot doit être écrit en intégral. Pour appliquer la règle des mots déjà abrégés dans la langue française, il faudrait ajouter un tableau mettant en relation les abréviations du français avec leurs correspondants en braille. La règle des mots étrangers en intégral ne peut pas non plus s'appliquer puisqu'elle exigerait une capacité à détecter la langue du texte.

Certaines règles de cet abrégé font malheureusement appel à la nature du mot à abréger. Dans le cas des mots se terminant par `ient`, le verbe `convient` s'abrège différent selon qu'il correspond au verbe `convier` ou `convenir`. On écrit `Ils convient ■is :viê`, mais `Il convient` s'abrègera `■i :v.t`. Un autre problème survient avec les homonymes. On écrira `sous la table »' t^`, mais `des sous` s'abrègera `ds süs`. Pour traiter ces exceptions, le script doit être en mesure de déterminer la nature du mot dans la phrase. Bien qu'il existe des techniques pour y parvenir, elles ne sont pas parfaites et alourdiraient inutilement le script d'abrègement.

Système 1923

Il est impossible d'appliquer correctement les règles des noms propres, car on ne peut pas faire la différence entre un véritable nom propre et un mot commençant par une majuscule en début de phrase. De plus, aucun algorithme de sélection de la lettre à désabréger dans le cas des signes inférieurs n'a été trouvé.

7 Le problème de l'expansion

Quel que soit le système employé, l'expansion constitue un autre problème. À partir du texte formaté par le script de formatage ou un logiciel de traduction braille tel que Duxbury Braille Translation, il est envisageable d'obtenir la version intégrale, possiblement même de retrouver les codes de formatage braille à l'origine du document final. Ce procédé permettrait un réabrègement dans un autre système sans avoir recours à un fichier source contenant des commandes Duxbury.

La procédure consiste à découper le texte abrégé en lignes puis les lignes en chaînes séparées par des espaces, correspondant à des mots abrégés. On désabrège ensuite chacune de ces chaînes séparément. Si la chaîne constitue une abréviation de mot, elle est remplacée par le mot en intégral. Dans le cas du système Fontaine, la fin de la chaîne est analysée pour la présence de finales. On parcourt ensuite chaque caractère de la chaîne afin de savoir s'il pourrait correspondre à une séquence abrégée et on note toutes les valeurs possibles de ce caractères. On effectue ensuite l'éliminations des valeurs abréviatives qui ne sont pas du tout

utilisables puis on détermine quels mots pourraient produire les agencements de séquences désabrégées.

7.1 Expansion des mots

Si la chaîne abrégée correspond directement à une abréviation de mot, elle sera remplacée par le mot intégral sans autre vérification. Bien qu'elle fonctionne dans la plupart des cas, cette méthode provoque des résultats incorrects pour certains cas. Par exemple, dans le système Fontaine, la phrase ■`vùci ’ let »e k.` correspondra à ■`voici la lettre au.`.

Si la chaîne débute ou se termine par un signe de ponctuation, ce signe sera retiré et une vérification sera effectuée pour un mot complet. Par exemple, dans la phrase ■`b, i l’a ".`, l'expansion du mot `bien` sera correctement effectuée malgré la présence de la virgule et du symbole de majuscule.

On découpe ensuite le mot afin de pouvoir traiter les cas des mots composés. Le découpage est effectué avec le trait d'union puis l'apostrophe. Par exemple, la chaîne `a!-midi` ne correspond à aucun mot en intégral. Toutefois, en séparant en deux groupes suivant le trait d'union, on peut obtenir l'expansion de `a!` : `après`.

L'analyse des finales Fontaine consiste à prendre les derniers caractères de la chaîne puis à déterminer s'ils constituent des consonnes ou leur expansion en est uniquement formée. Malheureusement, il faut effectuer l'expansion avec et sans finale. Par exemple, si on prend le mot `dépôt`, on peut détecter une finale erronée `pidrité` qui s'abrègera `pôt`. Un traitement et une élimination de ce mot, ainsi qu'un traitement du mot sans expansion sur la finale Fontaine, sont donc tous deux nécessaires.

7.2 Analyse des caractères

Si la chaîne ne correspond à aucun mot complet abrégé, on la découpe en caractères puis on vérifie si chacun des caractères donnés correspond à une séquence abrégée. Par exemple, la chaîne `c`` contient deux caractères. Le caractère `c` ne correspond par à une séquence abrégée, mais ` correspond à `ar`.

7.3 Élimination des valeurs inutilisées

Dans certains cas, une sélection des séquences est possible. Par exemple, l'expansion de la chaîne ■`ceci` donnera `{■\eur}ceci`, mais puisque le symbole ■ se trouve en début de mot, il constitue nécessairement un symbole de majuscule. Cette règle fonctionne bien, car aucun mot ne commence par `eur` dans une même syllabe. Malheureusement, si un mot est immédiatement précédé de guillemets ouvrants («), il sera difficile de déterminer où il commence.

Tout symbole de ponctuation en début ou en fin de mot sera lui aussi nécessairement employé comme symbole et non comme séquence. Par exemple, l'expansion de la chaîne `ceci.` sera correcte, car la sélection entre le point et la séquence `vr` sera bien effectuée. Une

subdivision s'avère nécessaire pour les signes de ponctuation en début de mot. De façon générale, aucun signe de ponctuation fermant ne se trouve en début de mot.

Le choix entre une séquence et un symbole non alphanumérique demeure évident. Par exemple, dans le cas de la chaîne `c`` dont l'expansion est `c{`\ar}`, la sélection de la séquence `ar` a plus de sens que le caractère `` ` ``.

7.4 Détermination des agencements valides

Malgré l'élimination, certaines ambiguïtés demeurent, comme par exemple la valeur du caractère `û` dans la chaîne `tût`. On détermine les mots pouvant être formés par l'agencement des séquences, dont le nombre dépend du nombre d'ambiguïtés à résoudre. Dans ce cas, deux mots peuvent être formés : `tût` et `test`. En utilisant le dictionnaire des mots du Français fondamental, on peut éliminer les mots inexistants dans la langue, mais dans cet exemple, aucune élimination n'est possible. Dans le cas de la chaîne `m`àe`, les mots suivants sont possibles : `maràe` et `marche`. Puisque seul le mot `marche` existe, il sera conservé et l'autre, éliminé. Si plus d'un mot existe, on gardera l'ensemble de ces mots et on les notera comme un groupe avec l'utilisation des acolades. Si aucun des mots n'existe, par exemple si un mot en langue étrangère ou un mot inventé avait été abrégé, tous les mots seront gardés.

Les mots sont ensuite triés en fonction du nombre de signes spéciaux qu'ils contiennent. La plupart du temps, le mot comportant le plus de symboles alphabétiques constitue le véritable mot abrégé.

7.5 Limitations de la procédure

Comme on peut facilement s'en rendre compte, l'expansion mène à plusieurs ambiguïtés que le lecteur est normalement capable de résoudre en utilisant le contexte. Malheureusement, ici, ces ambiguïtés demeurent présentes dans le résultat qui devra être entièrement révisé à la main pour fins de correction.

La vitesse de la procédure varie en fonction du nombre de mots qu'il faut chercher dans le dictionnaire. Si on parvient à définir des règles d'élimination suffisamment précise, il est possible d'accélérer le programme. Dans le cas contraire, l'expansion est plus lente et ne fonctionne bien qu'avec des mots du dictionnaire, et non avec des mots du registre courant.

A Fonctionnement du script

Le langage Perl se prête tout particulièrement à ce genre de script en raison de sa portabilité et de sa capacité à manipuler les chaînes de caractères et à effectuer des substitutions de façon efficace. Grâce à ce langage, il a été possible d'écrire un script capable d'abréger un texte dans le système de Carmen Fontaine, dans celui de 1955 ainsi que dans celui de 1923. Le programme ainsi que sa documentation se trouve à l'adresse `http://www.ericbuist.com/fontabui/`.

A.1 Fichiers programme

fontabui.pl **:** Contient l'interface en ligne de commande et effectue l'appel des autres modules afin de fournir un système unifié et cohérent à l'utilisateur.

abrege.pm **:** Contient les routines permettant la conversion effective en abrégé.

brl.pm **:** Package qui contient l'alphabet braille et des routines de conversion en fonte braille, en LaTeX et en LaTeX braille.

chunk.pm **:** Classe permettant la gestion d'un bout de texte ou d'une commande Duxbury issus du document à formater.

duxcmd.pm **:** Permet l'interprétation des différentes commandes Duxbury supportées par le formateur. Ce fichier comporte une routine pour chacune des commandes traitées et interagit avec la classe *formatter* afin que les commandes produisent un effet sur le document final.

formatter.pm **:** Classe permettant la gestion d'un ensemble d'objets de type *chunk*. Effectue le formatage proprement dit du texte et retourne, par le biais de la méthode *get*, le texte sous forme d'une chaîne de caractères.

tab.pm **:** Classe contenant les données nécessaires à la gestion d'une tabulation, c'est-à-dire une portion de ligne caractérisée par une position dans la ligne, un alignement, ainsi que d'autres paramètres. Le formateur produit les lignes sous forme d'un ensemble de tabulations avant de les convertir en texte ordinaire.

Hyphen.pm **:** Module Perl permettant la coupure de mots avec un algorithme basé sur celui de LaTeX, provient du Web.

Ce programme s'avère portable et fonctionne sous toute plateforme pour laquelle Perl a été porté. Pour simplifier le travail sous Windows, un fichier exécutable a été créé et peut fonctionner sans le besoin de l'interpréteur Perl.

A.2 Fichiers de données

s*`Système`*.abr **:** Contient la liste des mots avec leurs abréviations pour un système donné.

s*`Système`*.seq **:** Contient la liste des séquences pour un système donné.

s*`Système`*.loc **:** Contient la liste des locutions pour un système donné.

frhyphen.tex **:** Fichier TeX contenant des informations au sujet de la coupure des mots. Des informations ont été ajoutées à ce fichier pour traiter certains cas d'exception.

vient.txt **:** Liste des verbes à la troisième personne du pluriel se terminant par `ient` utilisée pour l'application de l'exception reliée à la règle de la séquence `ient` dans l'abrégé de 1955.

vment.txt **:** Liste des verbes à la troisième personne du pluriel se terminant par `ment` utilisée pour traiter la règle de la séquence `ment` dans l'abrégé de 1923. règle

A.3 Format des séquences

Le langage Perl permet de manipuler facilement les chaînes de caractères et effectuer des recherches et/ou des substitutions de façon simple en utilisant des expressions régulières. Les tables de séquences et de mots sont stockées sous forme de tableaux associatifs, ce qui permet une recherche rapide de l'abréviation correspondant à une séquence. Les clés de la table constituent des expressions régulières permettant de chercher la séquence dans le mot, en tenant compte des contraintes, et les valeurs, les abréviations. Ces tables sont générées par la lecture de fichiers de données dans lesquels figurent les informations, voir section A.5 pour plus d'informations sur le format de ces fichiers. Les expressions régulières utilisables par Perl sont obtenues par la conversion des données du fichier.

Dans le cas des systèmes de 1955 et 1923, on fait appel aux références pour les locutions. Chaque clé du tableau associatif racine constitue un mot, le début d'une locution. Si à ce mot correspond un autre tableau associatif, la locution comprend un mot additionnel. On a trouvé l'abréviation lorsque la valeur de la table constitue une abréviation. La figure 2 montre un exemple d'une branche de cette structure. Cet arbre est créé lors de l'exécution du programme à partir d'un fichier de données.

```
"par" => {"conséquent" => "p_c",
          "exemple" => "p_e",
          "suite" => "p_s",
          "-" => {"dessus" => "p-d",
                  "dessous" => "p-ou"}},
```

Fig. 2 – Exemple d'une branche de la structure arborescente permettant le traitement des locutions

A.4 Découpage syllabique

Le découpage en syllabes des mots est effectué grâce à un package trouvé sur le Web et basé sur le système de coupure de mots de LaTeX. Ce qui donne des découpages typographiques représentant bien les syllabes des mots. Le découpage donne un tableau de positions duquel il est facile de retirer un élément afin de supprimer une division syllabique lorsque nécessaire.

Malheureusement, le découpage typographique est légèrement différent du découpage syllabique. Par exemple, le mot `ananas` comprendra les syllabes `a-na-nas`. Toutefois, la grammaire française interdit la coupure si une seule lettre se trouve avant le trait d'union. Ainsi, le découpage typographique deviendra `ana-nas`. Les erreurs de découpage auront bien entendu une répercussion importante sur l'abrègement.

Il existe deux façons de solutionner ce problème. Il est possible d'ajouter les mots posant problème dans la liste des exceptions et donner la coupure syllabique exacte que l'on souhaite. Malheureusement, on constate après quelques tests que la liste est longue. On peut toutefois

ajouter des patrons de coupure qui regrouperont plus d'un mot. Par exemple, on peut indiquer au système de coupure de séparer entre en et ce, permettant le découpage correct de tout mot se terminant par `ence`.

A.5 Format des fichiers de données

Le script utilise pour son fonctionnement un certain nombre de fichiers de données textes éditables manuellement, le format du fichier dépendant de l'extension de ce dernier.

Caractéristiques communes des fichiers `.abr`, `.seq` et `.loc`

Chacun de ces fichiers textes comprennent des informations utiles pour l'abrègement. Chaque ligne du fichier contient une seule information. Si une ligne vide ou contenant seulement des espaces est rencontrée lors de la lecture du fichier, elle sera ignorée. Dans une ligne, la chaîne `//` et les caractères qui la suivent sont ignorés, étant considérés comme des commentaires.

L'information d'une ligne non ignorée se divise en deux parties séparées par un ou des espaces. La partie droite de la ligne se trouve dans la fonte braille et ne doit contenir aucun caractère d'espacement tandis que la partie gauche est en ASCII ordinaire, majuscules ou minuscules. Ces deux parties seront converties en minuscules lors de la lecture du fichier.

Les fichiers `.abr`

Ce type de fichier contient l'ensemble des mots abrégés par un ou plusieurs symboles dans un système donné. La partie gauche contient le mot tandis que celle de droite, son abréviation dans la fonte braille. Le mot à abréger ne doit contenir aucun caractère non alphabétique, tel que l'espace ou le trait d'union.

```
// Cette ligne et la suivante seront ignorées

ai /
aie /e
aies /:
ait [
an 1
ans 1s
après a6
au k
```

FIG. 3 – Extrait du fichier *sFontaine.abr*

Les fichiers `.seq`

Ces fichiers comprennent les séquences avec leurs abréviations. Ces séquences constituent des chaînes de caractères alphabétiques avec des règles possibles pour restreindre l'emploi de la séquence dans les mots. La syntaxe des règles constitue une simplification des expressions régulières utilisées par le script.

Si le symbole ^ est rencontré en début de séquence, la séquence ne sera abrégée qu'en début de mot. Si le symbole $ est rencontré en fin de séquence, elle ne sera autorisée qu'en fin de mot. Les crochets, quant à eux, définissent un groupe de lettres dont un membre devra être voisin de la séquence dans le mot pour que cette dernière soit abrégée. Si un groupe quelconque `[g]` se trouve en début de séquence, celle-ci devra être précédée, dans le mot, d'un élément du groupe pour être abrégée. Une règle similaire est définie pour la fin de mot.

Les groupes sont formés soit d'un ensemble de lettres, comme par exemple `[bmp]` ou d'un caractère spécial. Le groupe `$v` correspond à l'ensemble des voyelles, le groupe `$c`, à l'ensemble des consonnes tandis que `*` représente tout caractère.

```
ez$ z         // ez en fin de mot -> z
ex[$c] x      // ex suivie d'une consonne -> x
au k          // au -> k
[$v]ss[$v] ! // ss entre des voyelles -> !
^es :         // es en début de mot -> :
^im[bmp] [    // im en début de mot, suivie de b, m ou p -> [
an[*] 1       // an suivie d'un caractère (sauf en fin de mot) -> 1
```

FIG. 4 – Extrait du fichier *s1955.seq*

Les fichiers `.loc`

Ces fichiers comprennent l'ensemble des locutions admises dans un système donné. Contrairement au fichier de mots, des espaces, des apostrophes et des traits d'union sont admis dans la partie de gauche.

```
à cause (_c
à présent (_6
à peu près (_p_6
au contraire k_c
aujourd'hui k'h
c'est-à-dire c'e-(-d
d'abord d'a
```

FIG. 5 – Extrait du fichier *s1923.loc*

Le fichier *frhyphen.tex*

Ce fichier TEX possède un format différent des précédents, car il est manipulé par le système de coupure de mots, obtenu sur le Web. Dans une ligne, ce qui suit le symbole % est considéré comme un commentaire. Le fichier comporte deux parties : les règles de césure et les cas d'exception.

Dans la section débutant après la ligne `\patterns{` et se terminant à la ligne avant le symbole `}` se trouvent les règles de coupure permettant un recoupement de plusieurs mots causant problème. On retrouve une seule règle par ligne et elle est formée d'une chaîne de caractères. Si un point se trouve en début de la règle, elle ne sera applicable qu'en début de mot tandis que si un point se trouve à la fin, elle ne s'appliquera qu'en fin de mot. Le groupe de lettres définit une séquence qui, si rencontrée dans le mot, sera soumise à la règle de césure. Pour indiquer une coupure, il suffit d'indiquer un poids à l'endroit de césure dans la séquence. Le poids constitue un chiffre de 0 à 9 indiquant la priorité de la règle sur les autres règles. Un chiffre impair autorise la coupure tandis qu'un chiffre pair l'interdit. Plus le chiffre est élevé, plus la règle sera prioritaire. Par exemple, la règle `iè5re` permet la coupure entre `iè` et `re` dans les mots tels que `première`, `dernière`, `derrière`, `arrières`, ...

La partie des exceptions débute à la ligne contenant `\hyphenation` et contient un mot par ligne. On insère un trait d'union là où une coupure est autorisée. Par exemple, on peut ajouter `chro-no-mé-treur`. Cette possibilité permet de couper correctement des mots pour lesquels il n'est pas possible de définir une règle de césure ou que la définition d'une telle règle perturberait le comportement sur les autres mots.

B Formatage braille

Afin de permettre une estimation correcte du nombre de pages braille d'un document abrégé, il s'avère nécessaire de formater le document selon les conventions du *Code de transcription braille de l'imprimé*. De façon simple, l'opération consiste à placer trente caractères par ligne et vingt-cinq lignes par page. En bas de page à droite, des numéros de page braille doivent être insérés et en haut à droite doivent figurer les numéros de page de l'imprimé.

Pour permettre une meilleure estimation du nombre de pages, il faut reproduire le plus fidèlement possible le formatage engendré par le logiciel Duxbury. Ce qui consiste à interpréter les codes dollars et à effectuer la mise en forme adéquate. Ces codes indiquent les sauts de paragraphes, les alignements centrés, les numéros de page de l'imprimé ainsi que les tabulations s'il y a lieu.

Le formateur interprète la plupart des commandes disponibles et produit un fichier semblable au résultat de Duxbury. Il insère des caractères form feed au début de chacune des nouvelles pages et effectue la coupure de mots en fin de ligne. Certaines commandes Duxbury ne sont toutefois pas implantées.

- Commandes relatives aux Computer BANA Code, codage braille pour l'informatique
- Commandes relatives aux codes Nemeth, notation braille pour les mathématiques
- Sauvegarde et restauration de la position du curseur

- Protection de blocs
- Numéros de page en chiffres romains
- La règle des signes inférieurs, pour les systèmes 1955 et 1923, n'est pas appliquée dans le cas d'un mot ayant subi une coupure.
- Certaines commandes du formateur, qui ne sont pas apparues dans les documents de travail, n'ont pas pu être complètement testées.

C Textes utilisés

Lors des tests de comparaison entre les systèmes, huit textes ont été utilisés afin d'établir un gain moyen.

- Goleman, Daniel, *L'intelligence émotionnelle*, p. 1–47.
- Goleman, Daniel, *L'intelligence émotionnelle*, p. 48–105.
- Morency, Pierre, *Lumières des oiseaux, histoire naturelle du Nouveau Monde*, p. 1–114
- Bourgault, Pierre, *Maintenant ou jamais*, p. 1–102.
- McCullough, Colleen, *Les maîtres de Rome*, p. 1–70.
- Jardin, Alexandre, *Le petit sauvage*, p. 1–56.
- Werbowski, Tecia, *L'Oblomova*, p. 1–61.
- Atwood, Margaret, *La voleuse d'hommes*, p. 1–36.