Recto / Verso
Un système de conversion automatique ancienne / nouvelle orthographe à visée linguistique et didactique

Richard Beaufort Anne Dister Hubert Naets Kévin Macé
Cédrick Fairon
CENTAL / Université Catholique de Louvain, 1 Place Blaise Pascal, B-1348
Louvain-la-Neuve, Belgique
{richard.beaufort, anne.dister, hubert.naets, kevin.mace,
cedrick.fairon}@uclouvain.be

Résumé. Cet article présente Recto / Verso, un système de traitement automatique du langage dédié à l'application des rectifications orthographiques de 1990. Ce système a été développé dans le cadre de la campagne de sensibilisation réalisée en mars dernier par le Service et le Conseil de la langue française et de la politique linguistique de la Communauté française de Belgique. Nous commençons par rappeler les motivations et le contenu de la réforme proposée, et faisons le point sur les principes didactiques retenus dans le cadre de la campagne. La plus grande partie de l'article est ensuite consacrée à l'implémentation du système. Nous terminons enfin par une première analyse de l'impact de la campagne sur les utilisateurs.

Abstract. This paper presents Recto / Verso, a natural language processing system dedicated to the application of the 1990 French spelling rectifications. This system was developed for supporting the awareness-raising campaign promoted last March by the Superior council of the French language in Belgium. We first remind the motivations and the content of the reform, and we draw up the didactic principles followed during the campaign. The most important part of this paper is then focused on the system's implementation. We finally end by a short analysis of the campaign's impact on the users.

Mots-clés : Rectifications orthographiques de 1990, conversion ancienne / nouvelle orthographe, objectifs didactiques, machines à états finis.

Keywords: 1990 French spelling rectifications, ancient / new spelling conversion, didactic purposes, finite-state machines.

1 Introduction

En 1990, le Conseil supérieur de la langue française, présidé par Michel Rocard, et réunissant des personnalités et des experts français, belge et québécois, proposait un ensemble de rectifications orthographiques, connues sous le nom de « nouvelle orthographe » (Goosse, 1991), et approuvées par toutes les instances francophones compétentes en matière de langue (en ce compris l'Académie française). Ces modifications, proposées aux usagers sans leur être imposées, entrent peu à peu dans l'usage, mais sans que leur emploi ne soit systématique. Ainsi,

on constate que ces nouvelles graphies, bien qu'intégrées dans les correcteurs orthographiques actuels (Fontenelle, 2006) et préconisées dans l'enseignement, sont encore parfois sanctionnées chez les élèves, tant en France qu'en Belgique. Fort de ce constat, le Conseil de la langue française et de la politique linguistique de la Communauté française de Belgique a réalisé en mars dernier une vaste campagne de promotion des rectifications auprès des usagers, action de sensibilisation à laquelle la presse francophone belge a participé en « passant à la nouvelle orthographe » pour la version en ligne des quotidiens. Ce « passage » a été rendu possible grâce au développement d'un système de traitement automatique du langage spécialement dédié à cette tâche : Recto / Verso. C'est ce système que nous présentons ici.

2 La campagne de sensibilisation

En Communauté française de Belgique, un dépliant reprenant « 7 règles pour nous simplifier l'orthographe » a été diffusé à 300 000 exemplaires, conjointement à la publication en octobre 2008 de quatre circulaires ministérielles officialisant le choix de la « nouvelle orthographe » pour l'enseignement du français en Belgique.

S'en est suivie une vaste campagne de sensibilisation au travers de la presse en ligne : quatre des plus grands groupes de la presse francophone belge (soit un peu plus de 90 % de la presse en ligne) ont fait le pas de passer à la nouvelle orthographe.[1]
Cette campagne a été menée durant la « Semaine de la langue » de mars 2009. Les éditions électroniques des journaux ont alors été proposées au lecteur dans deux versions : en orthographe « standard », comme d'habitude, et en orthographe rectifiée. Dans le principe, un bouton nommé « RECTO/VERSO » (RECTifications Orthographiques / VERSion Originale) permettait de basculer d'une version à l'autre.

Cette action poursuivait deux objectifs complémentaires. Premièrement, elle voulait sensibiliser les grands éditeurs à la question de la nouvelle orthographe, parce qu'il est aujourd'hui évident que la nouvelle orthographe ne pourra se diffuser dans le public que si les grands éditeurs l'adoptent au préalable. Le passage temporaire de la presse à la nouvelle orthographe, dans le cadre de la campagne, est certainement un signe encourageant dans ce sens.
Deuxièmement, la campagne avait une dimension pédagogique. L'idée était d'une part de permettre au public de découvrir la nouvelle orthographe au travers de cas concrets dans des articles réels, et d'autre part de guider cette découverte, au travers de notes explicatives présentant les rectifications introduites dans le texte.

Les versions en ligne des quotidiens se présentent classiquement sous la forme de pages HTML. Dans le cadre de la campagne, nous devions donc générer des pages HTML en orthographe rectifiée au départ de textes en orthographe traditionnelle.

2.1 Les contraintes pédagogiques

Afin de donner une dimension pédagogique aux pages en nouvelle orthographe, il fallait que les formes rectifiées soient mises en évidence par rapport au reste du texte de la page, et soient expliquées à l'utilisateur.

[1] Voir `www.lesoir.be` ; `www.lalibre.be` ; `www.dhnet.be` ; `www.sudpresse.be`

Des formes rectifiées visibles. Nous avons décidé d'utiliser des couleurs, réparties en deux niveaux complémentaires : d'une part, une couleur pour l'ensemble du mot concerné, de manière à faciliter sa localisation dans la page, et d'autre part, une couleur pour la partie du mot touchée par la rectification, de manière à accélérer l'identification visuelle des modifications réalisées.
Pour ce faire, nous avons utilisé une feuille de style CSS, qui autorise entre autres l'utilisation de balises `<span>` pour formater un texte. Nous avons donc défini deux niveaux de balises `<span>` : le premier, `"recto.."`, correspond au mot rectifié dans sa globalité et indique si la forme a été modifiée (`"recto"`) ou est une exception (`"recto_ex"`) ; le second, `"rule.."`, entoure la partie du mot concernée par une règle, et indique par un chiffre de 1 à 9 le numéro de la règle concernée. Par exemple, la phrase

```
le bûcher est là                                                    (1)
```

deviendra, après rectification,

```
le <span class="recto">b<span class="rule4">u</span>cher</span> est là  (2)
```

où `"recto"` et `"rule4"` indiquent conjointement que la règle relative à la suppression de l'accent circonflexe sur *i* et *u* a bien été appliquée. Le CSS définit la couleur à appliquer pour chaque classe `"recto.."` et pour chaque classe `"rule.."`. En l'occurrence, nous avons décidé d'utiliser l'orange foncé pour `"recto"`, le vert pour `"recto_ex"` et le bleu pour toutes les classes `"rule.."`, afin de ne pas entrainer de surcharge cognitive.

Des rectifications expliquées. Après analyse, et afin de modifier le moins possible l'apparence de la page rectifiée, nous avons décidé d'opter pour l'affichage d'infobulles, n'apparaissant que lorsque l'utilisateur passe la souris sur une rectification donnée.
Pour ce faire, nous avons exploité la présence des balises `<span>` à l'aide de la librairie JavaScript jQuery.[2] Le fonctionnement de jQuery tient dans la définition de la fonction JavaScript `"ready()"` qui exécute au chargement de la page les instructions qui y sont contenues. Dans le cas présent, il y est décrit que lorsque le curseur de la souris passe au-dessus de l'une des balises ajoutées, une infobulle doit s'afficher avec un texte. Ce texte, qui diffère selon la combinaison `"recto.."` / `"rule.."` rencontrée, est conservé dans un tableau JavaScript, ce qui permet de dissocier le comportement des infobulles de leur contenu.

2.2 Un service accessible en ligne

Afin de faciliter l'accès au système par les journaux en ligne et dans le but de simplifier les opérations de mise à jour, Recto / Verso a été conçu sous la forme d'un service web de type SOAP. Les quotidiens en ligne ont pu transmettre ainsi leurs articles à un serveur distant, à l'aide d'une API très simple, et recevoir en retour ces mêmes textes rectifiés et ponctués des balises indiquant l'emplacement et la nature des rectifications.

Les articles de presse n'ont toutefois pas été les seuls textes concernés par la campagne, puisqu'une interface web accessible au grand public a également été mise en place. Accédant elle aussi au service web, elle a permis à chacun d'employer le système de rectifications.

Du côté serveur, l'architecture reste simple : le service web reçoit le texte à rectifier ; il y applique une série de prétraitements visant à normaliser l'encodage du document ; le texte est ensuite transmis au moteur de rectifications, et, une fois traité, renvoyé au client par le service web.

[2] Voir `http://jquery.com`

3 Implémentation

3.1 Outils utilisés

L'ensemble des traitements linguistiques décrits dans cet article ont été implementés sous la forme de machines à états finis. Selon les besoins, il s'agit d'automates ou de transducteurs, pondérés ou non (Mohri, 1996; Roche & Schabes, 1997).

Nous avons utilisé la bibliothèque de machines à états finis et le compilateur présentés dans (Beaufort, 2008). La bibliothèque propose la plupart des algorithmes standard de la littérature dans leur version pondérée. Elle autorise la création et la sauvegarde de machines dynamiques (construites au vol), et permet la sauvegarde et le chargement de machines binaires (complètement précalculées), plus compactes et plus rapides à charger. Le compilateur, développé autour de la bibliothèque, convertit en machines à états finis des règles de réécriture pondérées, des langages réguliers, des dictionnaires et des n-grammes. Les règles de réécriture autorisées prennent la forme générale

$$\varphi \rightarrow \psi : : \lambda \ \_ \ \rho \ / \ \omega \tag{3}$$

où φ, lorsqu'il est entouré par λ et ρ, se réécrit ψ et se voit attribuer le poids ω. Dans cette formulation, φ, ψ, λ et ρ sont des expressions régulières (McNaughton & Yamada, 1960) et ω est défini sur le semi-anneau tropical (Kuich & Salomaa, 1986). Dans tous les cas, le résultat de la compilation est une machine à états finis au format binaire de la bibliothèque.

3.2 Architecture générale du système

Les réflexions préliminaires au développement de Recto / Verso ont été les suivantes :

1. Certaines règles de la réforme ne sont pas purement lexicales. C'est le cas, par exemple, du pluriel des noms composés ou de l'accord du participe passé *laissé* devant infinitif. Recto / Verso devait donc inclure un moteur d'analyse linguistique.
2. Dans le cadre de la campagne de sensibilisation, Recto / Verso devait traiter des pages HTML et donc être capable de masquer les éléments non linguistiques le temps du traitement, et de les réintroduire ensuite dans la version réformée, afin de respecter l'affichage prévu par le créateur de la page.

Sur la base de ces considérations, Recto / Verso a été pourvu des modules suivants :

1. Un module de prétraitement HTML, qui cache le contenu non linguistique du texte à traiter ;
2. Un module de désambigüisation linguistique, qui analyse le contenu linguistique du texte ;
3. Un module de conversion, qui applique la réforme sur le résultat de l'analyse ;
4. Un module de post-traitement HTML, qui réinsère les éléments non linguistiques dans le texte réformé.

Nous commençons en 3.3 par décrire le moteur d'analyse linguistique, qui constitue le cœur du système. Nous détaillons ensuite en 3.4 les modules de pré- et post-traitement HTML, qui travaillent de pair. Ces modules font toute l'originalité et la robustesse de l'application. Nous terminons enfin en 3.5 par le module de conversion, qui est la raison d'être de Recto / Verso.

3.3 Le module de désambigüisation linguistique

Ce module est fortement inspiré de (Beaufort, 2008), auquel nous renvoyons le lecteur pour une description détaillée. Dans ce système, la phase de désambigüisation linguistique se limite à la suite d'étapes nécessaires à segmenter un texte en unités de sens (dates, téléphones, URLs, ..., nombres, formes lexicales), et à attribuer à chaque unité et à chacune de ses parties une et une seule analyse morphologique possible : *catégorie*, *genre*, *nombre* et *personne*. Aucune structure syntaxique n'est construite, et aucune analyse sémantique n'est entreprise. Le système est organisé en trois couches : un prétraitement, une analyse morphologique et une désambigüisation syntaxique.

1. Le prétraitement est réalisé par un transducteur Pre obtenu par compilation de règles de réécriture qui décrivent les unités de sens à détecter. En cours de processus, un texte W est donc composé avec le transducteur Pre, ce qui produit un transducteur W', dont l'entrée contient le texte et la sortie contient uniquement les unités détectées, chaque unité étant systématiquement située en face du premier caractère qui lui appartient. La figure 3.3 en donne une illustration. A partir de ce résultat et en appliquant l'algorithme de segmentation de machines à états finis proposé par (Beaufort, 2008), W' est segmenté en un vecteur de machines à états finis $V_{W'}$, où chaque machine correspond à une seule unité. Ce sont les machines mémorisées dans ce vecteur qui subiront les étapes suivantes du traitement.
2. L'analyse morphologique repose principalement sur deux transducteurs pondérés. Le premier, dédié aux formes connues, représente un simple lexique de formes fléchies. Le second, dédié aux formes hors vocabulaire, permet de choisir les catégories à attribuer aux formes en fonction de leur terminaison (il correspond à l'expression régulière « $.^+ \{suf\}$ » où « $.^+$ » est une suite non nulle de caractères, et « $\{suf\}$ » est l'ensemble des suffixes du français, complété de la chaine vide ϵ). Les deux transducteurs sont pondérés par des modèles de mot distincts, de la forme $-\log p(w_i|t^i)$.
3. L'analyse syntaxique, enfin, s'organise en deux niveaux. Le premier, représenté sous la forme d'un automate pondéré, implémente un n-gramme $-\log p(t^i|t^{i-2}, t^{i-1})$ lissé par interpolation linéaire (Beaufort *et al.*, 2002). Le second est un transducteur qui regroupe des suites de catégories susceptibles de constituer des formes composées de différents types : mots composés (*e.g. nom, tiret, nom* $\rightarrow$ *nom*), formes verbales composées plus ou moins complexes (*e.g. auxiliaire, participe passé* $\rightarrow$ *verbe*), etc. On retiendra que les deux niveaux d'analyse sont conservés dans des transducteurs distincts mais alignés, de manière à donner aux traitements suivants la possibilité d'un accès à une information macro- ou micro-scopique.

On notera que la composition de l'analyse morphologie et de l'analyse syntaxique permet la reconstitution d'un modèle de langue complet sous la forme de Bayes :

$$P(T|W) = \arg\min_T \sum_i -\log p(w_i|t^i) \circ \sum_i -\log p(t^i|t^{i-2}, t^{i-1}) \tag{4}$$

Après désambigüisation, le texte se présente sous la forme de plusieurs machines à états finis alignées. Leur parcours en parallèle permet de remplir une structure de données, que nous avons baptisée OrthoML et qui présente les niveaux d'inclusion suivants :

paragraphe $\rightarrow$ *phrase* $\rightarrow$ *unité de sens* $\rightarrow$ *forme composée* $\rightarrow$ *forme simple*

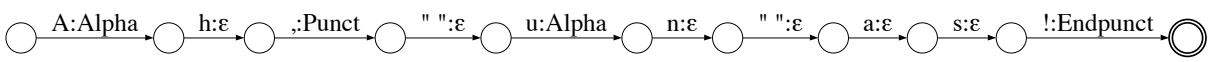

FIG. 1 – Application du prétraitement sur le texte « Ah, un as ! »

3.4 Les modules de pré- et post-traitement HTML

Une page HTML contient deux types d'éléments non linguistiques :

1. *Des éléments structurants.* Il s'agit exclusivement des balises HTML (ouvrantes et fermantes) permettant de délimiter des portions cohérentes de texte : une phrase (`<BR>`), un paragraphe (`<P>`), une liste (`<UL`, `<OL>`, `<LI>`) ou un titre (`<H1>`, `<H2>`, etc.). Dans une page HTML, ces balises sont parfois le seul indice d'une ponctuation forte, absente du texte lui-même.
2. *Du bruit.* Il s'agit de tout le reste. Dans ce reste, nous avons décidé de placer les balises de formatage des polices (`<B>`, `<I>`, `<U>`, etc.), qui peuvent faire irruption n'importe où dans le texte, que ce soit entre deux mots d'une même phrase ou dans un mot lui-même.

Pour chaque type, nous avons construit un nouveau transducteur de prétraitement :

1. Tag, qui identifie les éléments structurants du texte et leur attribue une nouvelle unité, le Tag. Ce transducteur est obtenu et fonctionne de manière similaire à Pre.
2. Hid, qui identifie le bruit présent dans le texte et le cache à l'analyse. Dans ce cas-ci, les règles de réécriture ne projettent plus les éléments détectés sur une unité, mais sur la chaine vide ϵ. Dans le résultat de la composition $W \circ Hid$, le bruit est présent sur l'entrée, mais est absent de la sortie du transducteur.

Idéalement, nettoyer le texte, identifier les éléments structurants et détecter les unités classiques devraient être réalisés comme suit :

$$W' = \pi_2(W \circ Hid) \circ Tag \circ Pre \tag{5}$$

où $\pi_2(\cdot)$ note la projection du langage de sortie de $(\cdot)$.

Cependant, la projection fait disparaitre le bruit de la page, de manière irrécupérable. Or, l'objectif est d'appliquer la réforme sur le texte, tout en conservant l'intégralité de la page d'origine. Pour ce faire, nous avons procédé comme suit :

1. Le prétraitement est réalisé sans projection : $W' = W \circ Hid \circ Tag \circ Pre$.
2. Le résultat du prétraitement est dupliqué, et la projection est appliquée sur la copie uniquement : $W'_\pi = \pi_2(W')$.
3. L'étape de segmentation du texte en unités, sous la forme de machines à états finis conservées dans un vecteur, est appliquée aux deux versions, la bruitée et la projetée.
4. La désambigüisation morphosyntaxique et l'application des rectifications orthographiques sont réalisées sur le vecteur correspondant à la version projetée, W'_π.
5. A la fin du traitement, un alignement entre les deux versions du texte est réalisé, de manière à contenir à la fois la totalité de la page initiale, et les rectifications introduites dans le texte. Dans le principe, cet alignement est obtenu comme suit entre les deux versions du texte :

   $$W'' = Best(W' \circ W'_\pi) \tag{6}$$

   si ce n'est qu'il est réalisé au niveau des deux vecteurs d'unités :

   $$W'' = Best(V_{W'}[1] \circ V_{W'_\pi}[1]) \cdot \ldots \cdot Best(V_{W'}[n] \circ V_{W'_\pi}[n]) \tag{7}$$

3.5 Le module de conversion

3.5.1 Description de l'algorithme

L'algorithme consiste en un parcours récursif des différentes couches de la structure de données remplie par le module de désambigüisation. Toutes les opérations sont réalisées au travers de machines à états finis, par composition avec des modèles précompilés. Le niveau où tout commence est en fait celui des unités. Le processus de rectification n'y est cependant déclenché qu'en présence d'une unité alphabétique, et se déroule en deux étapes.

La première étape traite successivement les différentes formes composées (`Comp`) de l'unité. Pour rappel, à ce stade de l'analyse, une forme composée est un *niveau* de la structure de données, mais peut en réalité ne contenir qu'une forme simple. Le traitement réalisé à ce niveau diffère en fonction de la catégorie de la forme, et seuls les déterminants et les substantifs sont concernés. S'il s'agit d'un déterminant, son nombre (`Number`) est conservé pour un emploi ultérieur. S'il s'agit d'un substantif, le système tente de lui appliquer les règles de rectification, relatives aux mots composés et aux mots étrangers, qui sont contenues dans un *lexique* :

$$(Fsm(\texttt{Comp}) \ . \ Fsm(\texttt{Number})) \circ \texttt{fsmRectoLexicon} \tag{8}$$

C'est ici que le nombre du dernier déterminant rencontré intervient, puisqu'il est utilisé pour préciser au lexique si la forme traitée est au singulier ou au pluriel. Qu'il y ait eu rectification ou pas, cette étape se termine par la concaténation de la forme composée à la machine à états finis représentant l'unité complète.

La seconde étape concerne l'unité complète (`fsmUnit`) et est réalisée sous la forme d'une cascade de deux compositions :

$$\texttt{fsmUnit} \circ \texttt{fsmRectoRule} \circ \texttt{fsmMarkRule} \tag{9}$$

A ce stade, les mots étrangers et les mots composés concernés par la réforme ont déjà été traités. Toutes les autres rectifications sont encore à appliquer, ce qui est fait dans la première composition de la cascade. La deuxième composition de la cascade n'applique plus de règles de rectification, mais insère, autour des formes modifiées, les balises `<span>` qui permettront le formatage du texte et l'affichage des bulles informatives lors de la lecture de la page dans un navigateur. Ces balises s'insèrent autour de *toutes* les rectifications, y compris celles réalisées sur les mots étrangers et les mots composés.

L'algorithme se termine par la création et l'impression (dans un fichier de sortie) de la chaine de caractères correspondant à l'unité. C'est à ce moment qu'est réalisé le post-traitement HTML décrit en 3.4, où un alignement permet de combiner dans le texte de sortie à la fois les rectifications réalisées et les informations non linguistiques qui appartenaient à la page initiale.

3.5.2 Deux modes de gestion des rectifications

On le constate dans l'algorithme, les mots composés et les mots étrangers à rectifier sont gérés à l'aide d'un lexique, alors que les autres rectifications sont gérées par des règles de réécriture. Voici les justifications de ce choix.

Utilisation d'un lexique. Qu'il s'agisse des mots étrangers ou des mots composés, il est possible d'en dresser une liste exhaustive. En ce qui concerne les mots étrangers, il est bien évident

que leur nombre est *a priori* infini. Cependant, les seuls mots étrangers qui seront reconnus comme tels et sans erreur sont ceux recensés dans les lexiques utilisés. Dans le cadre de la campagne, ces lexiques ont bien sûr été complétés.
En ce qui concerne les mots composés, ceux concernés par les rectifications sont de la forme « porte-avion » (verbe + nom) et « sans-papier » (préposition + nom). Il est de ce fait possible d'exprimer l'ensemble des mots composés que le système peut rencontrer, sous la forme de l'expression régulière suivante :

$$(\{\texttt{PREP}\}|\{\texttt{VERBE}\}) \cdot \texttt{"-"} \cdot \{\texttt{NOM}\} \tag{10}$$

où {PREP} représente l'ensemble des prépositions du lexique, {VERBE} correspond aux formes verbales qui peuvent participer à la création d'un mot composé, et {NOM} fait référence à l'ensemble des substantifs du lexique. Grâce au compilateur utilisé, il est possible de décrire des lexiques dont les formes acceptées en entrée sont réécrites en sortie. En voici un exemple :

$$\begin{aligned} &\texttt{garde-chasses<SG>} \longrightarrow \texttt{garde-chasse} \\ &\texttt{garde-chasse<PL>} \longrightarrow \texttt{garde-chasses} \end{aligned} \tag{11}$$

où <SG> et <PL> indiquent respectivement que le mot est singulier ou pluriel. Ceci permet d'éviter de devoir décrire des règles de réécriture plus ou moins lourdes, dont chacune ne s'appliquerait, au final, qu'à une forme du lexique. Les différentes combinaisons possibles autorisent plus de 3 millions de mots composés potentiels (354 verbes / prépositions $\times$ 92 778 noms), compilés en 25,7s et représentés sous une machine à états finis compacte de 466 Ko.

Utilisation de règles. Les autres rectifications nécessitent des règles de réécriture pour l'une des deux raisons suivantes. Soit le nombre de formes concernées est potentiellement infini. C'est le cas des règles concernant l'accent circonflexe. Soit la règle nécessite la connaissance d'un contexte postérieur à la forme à modifier, et indescriptible sous la forme d'un lexique. C'est le cas, par exemple, de la règle « laissé + infinitif », qui implique de repérer, à l'aide d'une expression régulière, l'infinitif qui serait présent dans le contexte droit du participe.

3.5.3 Insertion des balises

Si les machines à états finis ont de nombreux avantages, elles présentent cependant un inconvénient majeur : elles ne facilitent pas l'identification d'un symbole de l'alphabet qui aurait été réécrit par l'un ou l'autre règle. Dans une machine, tous les symboles sont égaux. Il est dès lors difficile de repérer qu'une règle de réécriture a été appliquée.

Le compilateur que nous avons utilisé (cf. point 3.1) propose un mécanisme permettant de résoudre cette difficulté. Il autorise l'utilisation de *marqueurs*, symboles extérieurs à l'alphabet utilisé, que l'on peut insérer dans une règle de réécriture afin d'identifier un phénomène et d'en suivre l'évolution.

Dans notre cas, les marqueurs ont été exploités comme suit. Nous avons défini un couple de marqueurs par règle de rectification. Par exemple, [RU1] et [RU1END] pour la règle concernant le pluriel des mots composés, le premier indiquant le début d'une rectification, et le second en indiquant la fin. Ceci étant fait, voici comment se réécrit une ligne du lexique :

$$\texttt{garde-chasse<PL>} \longrightarrow \texttt{garde-chasse[RU1]s[RU1END]} \tag{12}$$

Grâce à la présence de ces marqueurs, l'insertion des balises a été fortement simplifiée. En effet, les règles précisent maintenant qu'en présence de marqueurs de ce type, il faut d'une

part introduire une balise globale autour de la forme complète :

```
<span class="recto">garde-chasse[RU1]s[RU1END]</span>          (13)
```

et d'autre part réécrire les marqueurs, en les replaçant par le nom de la règle correspondante :

```
<span class="recto">garde-chasse<span class="rule1">s</span></span>    (14)
```

4 Evaluation du système

Robustesse et nombre d'articles traités. Lors du lancement de l'application, quelques bugs ont causé plusieurs fois l'arrêt brutal du système. Une rapide analyse nous a permis de les trouver et de les corriger, de sorte que l'application a traité, sans arrêt du système, plus de 1 000 000 de textes au cours de la Semaine de la langue française en fête (mi-mars 2009). L'action, depuis, continue. En effet, la presse partenaire, convaincue par la stabilité de l'application et ayant surtout constaté une augmentation de la consultation des articles en ligne, a souhaité continuer de proposer ce service à ses lecteurs. C'est ainsi qu'à l'heure où ces lignes sont écrites, Recto / Verso a traité plus de 7 000 000 d'articles, soit environ 3,2 articles/seconde.

Performances et taille des articles traités. L'application a été installée sur 4 serveurs pourvus d'une distribution Ubuntu Server JeOS 8.04.1, d'un processeur 3 GHz et de 2 048 Mo de RAM. Elle traite 20 000 caractères/seconde lorsque le traitement peut rester déterministe (absence ou faible présence d'éléments non linguistiques), et tombe au minimum à 10 000 caractères/seconde en cas d'indéterminisme fort (forte présence d'éléments non linguistiques).

Les articles traités ont une taille moyenne de 3 500 caractères, ce qui nous donne un traitement moyen de 170 ms. La plupart des textes oscillent entre 200 et 4 500 caractères, 2% des textes se situent entre 15 000 et 30 000 caractères, et 1 texte sur 50 000 environ dépasse les 60 000 caractères.

Qualité de l'analyse. Toutes les formes concernées par les Rectifications publiées au Journal officiel sont traitées par notre logiciel, hormis *punch* (la boisson) qui s'écrit désormais *ponch* conformément à la prononciation. En effet, notre système ne propose pas de désambigüisation sémantique, et il ne nous était pas possible de distinguer la boisson de son homonyme (au sens de « dynamisme »).

Lors d'une révision manuelle de 1 000 articles, nous avons constaté à deux reprises que les noms propres (*Nivelles*, *Capelle*), homographes de formes concernées par la réforme (*tu nivelles*, *il capelle*), avaient été mal analysés et modifiés à tort. Ce sont les seules erreurs que nous ayons eu l'occasion d'observer.

5 Conclusions

Dans cet article, nous avons présenté Recto / Verso, un système dédié à l'application de la nouvelle orthographe, spécialement développé dans le cadre de la Semaine de la langue française en Belgique. Pour être acceptée par la presse et bien accueillie par le public, l'application devait être rapide, simple et intégrer une dimension pédagogique. Nos développements ont tâché d'atteindre ces objectifs. L'application, réalisée à l'aide de machines à états finis, traite entre 10 000

et 20 000 caractères/seconde selon le taux d'éléments non linguistiques contenus dans le texte. Le service web, de type SOAP, accepte des pages HTML complexes et y modifie les formes concernées par la réforme, sans altérer le reste de la page. Les feuilles de style et les scripts Javascript, enfin, facilitent l'identification des rectifications réalisées et permettent l'affichage d'infobulles explicatives.

Ceci ayant été mis en place, il restait à attendre le retour du public, dans le cadre de la campagne de sensibilisation. Recto / Verso a intéressé les lecteurs des quotidiens en ligne, qui l'ont testé massivement. La demande des journaux de maintenir le système sur leur site est d'ailleurs la preuve la plus tangible de cet intérêt du public pour la « nouvelle orthographe ».

Remerciements

Cette recherche a été partiellement financée par le projet « Vocalise », dans le cadre du programme FIRST Post-Doc de la Région wallonne (convention 716619), ainsi que par le Service de la langue de la Communauté française de Belgique.

Nous remercions Noémi Boubel, Bernadette Dehottay et Sophie Roekhaut, pour leur intervention dans la constitution de certaines ressources linguistiques, et Olivier Blanc, pour les conseils qu'il nous a donnés lors de l'établissement des principes de l'architecture client / serveur.

Références

BEAUFORT R. (2008). *Application des Machines à Etats Finis en Synthèse de la Parole. Sélection d'unités non uniformes et Correction orthographique*. PhD thesis, Faculté d'Informatique, Facultés Universitaires Notre-Dame de la Paix, Namur, Belgique. 605 pages.

BEAUFORT R., DUTOIT T. & PAGEL V. (2002). Analyse syntaxique du français. Pondération par trigrammes lissés et classes d'ambiguïtés lexicales. In *Actes des XXIV^{e} Journées d'Etudes sur la Parole (JEP'02)*, p. 133–136.

FONTENELLE T. (2006). Les nouveaux outils de correction linguistique de microsoft. In P. MERTENS, C. FAIRON, A. DISTER & G. PURNELLE, Eds., *Actes de la 13^{e} conférence sur le traitement automatique des langues naturelles (TALN'06)*, volume 1, p. 3–19, Louvain-la-Neuve : Presses Universitaires de Louvain.

GOOSSE A. (1991). *La « nouvelle » orthographe, Exposé et commentaires*. Louvain-la-Neuve : Duculot.

KUICH W. & SALOMAA A. (1986). *Semirings, Automata, Languages*, volume 5 of *EATCS Monographs on Theoretical Computer Science*. Berlin, Germany : Springer-Verlag.

MCNAUGHTON R. & YAMADA H. (1960). Regular expressions and state graphs for automata. *IRE Transactions on Electronic Computers EC*, **9**(1), 39–47.

MOHRI M. (1996). On some applications of finite-state automata theory to natural language processing. *Journal of Natural Language Engineering*, **2**, 1–20.

E. ROCHE & Y. SCHABES, Eds. (1997). *Finite-State Language Processing*. Cambridge, Massachusetts : MIT Press.