Traitement informatique de l'inflexion dans le Lunaf, dictionnaire électronique du luxembourgeois

Francisca LUNA GARCIA

Laboratoire d'Informatique et d'Intelligence Artificielle – INSA Strasbourg
24, boulevard de la Victoire
67000 Strasbourg
luna@message.lu

Université Marc Bloch
22, rue René Descartes
67084 Strasbourg Cedex

Résumé – Abstract

Afin de générer les formes fléchies des noms luxembourgeois dans le dictionnaire luxembourgeois, nous utilisons un code flexionnel. Ce code s'étant révélé trop contraignant pour traiter l'inflexion (alternance vocalique/Umlaut), nous présentons ici un moyen efficace pour coder ce phénomène. La pertinence de ce type de code est double. D'une part, il correspond mieux aux besoins du linguiste qui aimerait établir des classes flexionnelles naturelles sans trop de contraintes informatiques. D'autre part, il permet de réduire significativement le nombre de classes flexionnelles. Le dictionnaire électronique luxembourgeois dispose ainsi de deux codes qui peuvent se combiner entre eux pour mieux traiter les particularités morphologiques des mots luxembourgeois.

In order to generate the inflected forms of the Luxemburgish nouns in the Luxemburgish dictionary, we use a flexional. This code having proved to be too constraining to treat the inflection (vocalic alternation/Umlaut), we present here an effective means to encode this phenomenon. The relevance of this code is two-fold. On the one hand, it corresponds better to the needs of the linguist who would like to establish natural flexional classes without too many data-processing constraints. In addition, it reduces the number of flexional classes significantly. The Luxemburgish dictionary has thus two codes which can be combined for a better processing of the morphological characteristics of the Luxemburgish words.

Mots-clefs – Keywords

inflexion vocalique, codage, génération automatique, luxembourgeois
vowel inflection, encoding, automatic generation, Luxemburgish

1 Problématique et contexte

Avec notre thèse sur l'*Étude morphologique de la langue luxembourgeoise en vue de la création d'un dictionnaire électronique des mots luxembourgeois*, le luxembourgeois fait ses premiers pas dans le monde du dictionnaire électronique. Dans une première phase, il est question de créer un dictionnaire électronique grâce à la génération automatique de mots. Dans cette communication, nous allons nous intéresser à l'inflexion, phénomène qui nous a posé problème lors de la génération. En effet, les langues germaniques ne marquent pas

seulement le pluriel à la fin du mot, mais aussi à l'intérieur du mot. Le codage que nous avons adopté pour générer les mots luxembourgeois, c'est-à-dire pour obtenir à partir du lemme la forme plurielle des noms, s'est vite révélé inefficace pour traiter l'inflexion (alternance vocalique ou Umlaut). Nous obtenions un nombre de classes flexionnelles trop élevé. Nous voulons présenter ici le codage créé afin de pallier ce problème.

A notre connaissance, ce phénomène ne bénéficie pas d'un traitement particulier dans les dictionnaires électroniques des autres langues germaniques. Dans le dictionnaire électronique luxembourgeois, le Lunaf, nous utilisons deux types de codes, l'un flexionnel qui est utilisé pour générer les mots luxembourgeois, l'autre appelé « inflexionnel », utilisé uniquement pour coder l'alternance vocalique ou Umlaut lors de la génération de la forme plurielle des noms luxembourgeois. Ce code « inflexionnel » n'est intéressant que lorsqu'il s'agit d'apporter des modifications à l'intérieur d'un mot. Nous avons programmé le code pour qu'il ne fonctionne que pour les voyelles, puisque l'inflexion est assez régulière et qu'en luxembourgeois, il n'y a pas d'autres changements morphologiques à faire à l'intérieur du mot.

Ci-dessous, nous allons présenter le code flexionnel ainsi que le code « inflexionnel », tous les deux imposés par les caractéristiques de la langue luxembourgeoise. Nous voulons montrer qu'ensemble, ils nous permettent de générer de façon optimale les mots de cette langue. Mais tout d'abord, introduisons DicoManager, le système qui accueille le Lunaf, puisque le code dont nous allons parler est celui utilisé dans les dictionnaires électroniques de ce système.

DicoManager[1] est un système informatique qui n'a pas été créé pour une application particulière. Il offre la possibilité de créer pour chaque langue naturelle un dictionnaire électronique constitué de deux bases de données, auxquelles les applications feront appel entre autres pour la reconnaissance de mots dans un texte. En général, on considère que la fonction première d'une telle base de données est d'associer une chaîne de caractères représentant la forme des mots graphiques et sémantiques et des formes fléchies avec différentes informations rendant possible, par exemple l'étiquetage ultérieur des mots dans la phrase. Chaque dictionnaire électronique de DicoManager est donc constitué de deux bases de données lexicales, DicoCan et DicoForm. La première contient une liste de lemmes sous leur forme canonique, la deuxième, les formes fléchies des lemmes. Le Lunaf compte 325 classes flexionnelles dont 77 nominales. Ces 77 classes flexionnelles représentent chacune un code flexionnel qui permet de générer la forme fléchie, c'est-à-dire la forme plurielle du nom[2]. Voyons d'abord, en quoi consiste ce code flexionnel.

2 Le code flexionnel du Lunaf

Ce que nous codons ici, c'est la catégorie grammaticale du nombre. Les marques du pluriel des noms luxembourgeois peuvent être regroupées en cinq types flexionnels (Bruch, 1973 ; Poitou, 1987 ; Schanen, 1980 ; Newton, 1996). Deux de ces types marquent le pluriel par l'affixation et par l'inflexion. Un seul uniquement par l'inflexion.

---

[1] Système créé au sein du Laboratoire d'Informatique et d'Intelligence Artificielle (LIIA) de Strasbourg, programmé en Java et utilisant MySQL pour la création des bases de données.

[2] Les lemmes sont issus de notre corpus rassemblé à partir de textes sur Internet. Il comporte 500.000 mots et nous a permis d'obtenir 16000 mots sous leur forme lemmatisée.

Les lemmes (infinitif pour les verbes, nominatif singulier pour les noms et les adjectifs) de DicoCan vont subir la génération automatique. Nous aurons ainsi dans DicoForm (la base des formes fléchies) pour les mots variables, la conjugaison des verbes, la déclinaison des adjectifs et la forme plurielle des noms. Les mots invariables (adverbe, préposition, conjonction, etc.) sont stockés tels quels (sans subir de génération) dans les deux bases. Le code « inflexionnel », que nous allons voir plus loin ne concernant que les noms, nous ne parlerons pas dans cette communication des autres catégories lexicales.

Le code flexionnel qui permet la génération des formes fléchies se compose de deux parties, une partie qui contient les instructions pour générer les formes fléchies à partir du lemme – qui fait l'objet de cette communication – et une partie qui comporte les informations morpho-syntaxiques, notamment le genre, le nombre et le cas pour les noms. Voyons deux exemples de génération nominale : ***en/***... (type B) pour *Auto – Auto-en* (voiture), **2éi/**... (type D) pour *Schlag – Schléi* (coup).

Dans le premier exemple nous ajoutons la marque du pluriel <en> au nom, alors que dans le deuxième exemple nous déplaçons le curseur de deux lettres vers la gauche, ce qui efface les lettres <ag>, puis nous insérons <éi>.
Le codage utilisé dans les dictionnaires électroniques de DicoManager ressemble à celui utilisé dans le dictionnaire du LADL, le DELAF (dictionnaire des formes fléchies). Nous l'avons adapté à la langue luxembourgeoise et l'avons modifié pour pouvoir coder et générer les mots luxembourgeois dans le Lunaf. Pour construire le Lunaf, les lemmes peuvent subir jusqu'à quatre opérations : un nombre entier spécifie le nombre de lettres à enlever à partir de la fin du mot vers la gauche ; les caractères en minuscule ajoutent des lettres ; la commande *C* copie une lettre (il faut mettre autant de *C* qu'il a de lettres à ajouter) ; la commande *R* déplace le curseur et efface la lettre à droite (un R par lettre à effacer). Celles-ci sont réalisées par des transducteurs à états finis (Sproat, 1992) qu'il a fallu modifier pour traiter l'inflexion.

Afin de bien comprendre les raisons pour lesquelles nous avons cherché un code qui permette de coder plus efficacement l'inflexion, nous vous présentons brièvement les modifications apportées au codage que nous venons de décrire. En fait, les mots du Lunaf sont générés à partir de deux codes différents. Nous venons d'en voir un au paragraphe précédent. Il présente une légère variante en fonction des mots qu'il s'agit de générer. Pour générer certains temps verbaux par exemple, il est nécessaire d'utiliser deux autres commandes que nous décrivons ci-dessous.

Le participe II du verbe *léieren* (étudier), c'est-à-dire *geléiert* (appris), s'obtient à partir du code flexionnel suivant : BgeE2t/....
La **commande *B*** (Beginning) renvoie le curseur en début de mot. Cette commande peut se révéler importante lorsqu'il s'agit d'ajouter des préfixes en début de mot, comme par exemple pour ajouter le préfixe <ge> du participe II. Cette commande est en étroite liaison avec la **commande *E*** (End), qui renvoie le curseur à la fin du mot. A nouveau, elle est très utile pour la formation du participe II, puisque grâce à elle, après avoir ajouté le préfixe <ge> avec la commande B, elle place le curseur à la fin du verbe et permet ainsi l'ajout de la terminaison du participe II. Voyons en détail le code de l'exemple donné plus haut :
participe II du verbe *léieren* (étudier): BgeE2t/PART.

B	déplacer le curseur en début de mot
ge	ajouter <ge>
E	déplacer le curseur en fin de mot
2	enlever deux lettres

t ajouter <t>
Ce qui nous donne la forme fléchie *geléiert* (étudié). En somme, l'opération E (End) permet l'enchaînement d'opérations de suppression, de copie et de remplacement de lettres.

Nous venons de voir comment fonctionne le code flexionnel général. Il permet de générer la conjugaison des verbes, la déclinaison des adjectifs et de la plupart des noms luxembourgeois. En revanche, il a fallu implémenter un code spécifique pour traiter le phénomène de l'inflexion.

3 Le code « inflexionnel » du Lunaf

Nous avons voulu trouver un moyen efficace de coder l'inflexion des voyelles du radical. Le codage que nous avons décrit précédemment était trop compliqué et engendrait une multiplication des classes flexionnelles lorsqu'il s'agissait de coder l'inflexion. Il a donc fallu créer ce nouveau code. Il s'agit surtout de montrer la particularité de ce code par rapport à celui que nous venons de voir, ainsi que son utilité.
En fait, le codage du Lunaf propose trois façons de déplacer le curseur dans le mot : dans le code flexionnel le déplacement du curseur se fait lettre par lettre et/ou en début ou en fin de mot, et dans le code « inflexionnel » le curseur se place sur une lettre précise à l'intérieur du mot.

Nous avons donc ajouté la possibilité de chercher une chaîne de caractères et de la remplacer par une autre chaîne de caractères. Par exemple pour passer de *Hafen* (port) à *Häfen* (ports) on va coder l'information de la façon suivante dans le Lunaf : (a)ä)/... [3].

La voyelle du radical qui subit l'inflexion se met entre parenthèses, suivie de la voyelle infléchie. L'économie de classes flexionnelles se fait surtout lorsque les voyelles de plusieurs noms subissent la même inflexion. Par exemple, les noms comme *Stad* (ville), *Schued* (dégâts) et *Daach* (toit) deviennent au pluriel *Stied, Schied, Diech*. En utilisant le code flexionnel décrit en 2) pour générer ce type de pluriel, nous aurions les trois classes flexionnelles suivantes et donc trois codes différents: 2RieE/... pour *Stad*, 3RiE/... pour *Schued,* 4RrieE/... pour *Daach*. Avec le code « inflexionnel » ces trois noms et leurs composés ne sont regroupés que dans un seul code : (a;aa;ue)ie)/..... pour ***St**a**d** → **St**ie**d**, **D**aa**ch** → **D**ie**ch**, **Sch**ue**d** → **Sch**ie**d**.*

Un tel code fonctionne de gauche à droite, c'est-à-dire, que la première voyelle recherchée est celle qui se trouve le plus à gauche dans le code. Le programme cherchera d'abord la première voyelle, le <a>. S'il ne la trouve pas, il recherchera la voyelle suivante du code, le <aa>. Et si là encore il ne trouve pas, il passera au couple suivant, le <ue>. D'après notre corpus, nous avons obtenu un maximum de trois voyelles (ou double voyelles) dans chaque code.

La recherche de la voyelle dans le nom se fait de droite à gauche et cela pour permettre la mise au pluriel des noms composés ou pour gérer les cas où un même nom comporterait deux voyelles identiques. Ce choix de recherche dans le mot se justifie pour deux raisons. La première est due au fait que si nous ne procédions pas par la fin du mot, le générateur de formes ne saurait pas quelle voyelle doit subir l'inflexion. Voyons les exemples suivants : soit les noms *Akaafsstad* (ville commerciale) et *Karnavalstad* (ville de carnaval) et

[3] Les parenthèses ont pour seul but de délimiter les opérations sur le lemme.

le code flexionnel précédent : (a;aa;u)ie)/.... Est-ce le <aa> de *Akaafs-* ou le <a> de *-stad* ou lequel des quatre <a> de *Karnavalstad* qui doit subir l'inflexion ? Il se produirait une ambiguïté et le programme génèrerait une forme fléchie incorrecte.
La deuxième raison est étroitement liée à la nature même des noms composés. Le membre principal des noms composés se trouve toujours à la fin du composé. Dans les exemples que nous venons de donner, il s'agit de *Stad* (ville). Ce qui fait une bonne raison de commencer par la fin du mot. Ce code « inflexionnel » permet ainsi de générer de façon optimale les mots simples et les mots composés. Le transducteur fonctionne lui aussi de droite à gauche. Dans l'exemple précédent, le transducteur fait que l'on passe du lemme au singulier *Stad* (ville) à la forme lexicale au pluriel *Stied* (villes). Cette façon de procéder est décrite chez Cornell (1988). Le transducteur comporte également des informations morpho-syntaxiques. Il est composé d'un arc marqué par exemple a:ie pour manipuler la transduction de *Stad* à *Stied*. En plus, cet arc est marqué [+pl], signifiant qu'il introduit la propriété morpho-syntaxique [+pluriel] en même temps qu'il effectue la transduction du /a/ de surface vers le /ie/ lexical.

Quelques exemples de codes « inflexionnels » vont nous permettre de mieux visualiser les classes nominales concernées. Par rapport au code flexionnel, le code « inflexionnel » permet de relever des régularités dans l'inflexion, c'est-à-dire les voyelles ou diphtongues qui subissent la même alternance, puisque chaque code regroupe tous les noms qui marquent le pluriel de la même façon :
(a;o)ee) → *Bam – Beem* (arbre), *Nol – Neel* (clou, ongle),
(ou;o)éi) → *Fouss – Féiss* (pied), *Stot – Stéit* (ménage).

Dans ces exemples nous voyons qu'en luxembourgeois, il est possible que plusieurs noms subissent la même inflexion lors de la mise au pluriel (Luna Garcia, 2002 ; Schanen, 2004).
L'ordre des voyelles à remplacer n'est pas aléatoire. Il est déterminé en raison de la fréquence de remplacement, c'est-à-dire qu'une voyelle ou une diphtongue qui subit plus souvent l'inflexion prendra la première place dans le code. Ceci réduit le risque d'erreur lorsque le nom contient plus d'une voyelle.

Il est possible de combiner le code flexionnel et le code « inflexionnel », notamment lorsque le nom subit l'inflexion et l'ajout de la marque du pluriel en fin de mot. Pour générer la forme plurielle des noms suivants: *Lach – Lächer* (trou), *Faass – Fässer* (tonneau), nous combinons les deux codes de la façon suivante: (a;aa)ä)er/....

Les avantages du code « inflexionnel » sont donc :

- une génération similaire des mots simples et des mots composés,
- le déplacement du curseur sur une lettre ou un couple de lettres précises à l'intérieur d'un mot,
- la possibilité d'établir un ordre de fréquence,
- la possibilité de repérer les régularités flexionnelles
- la possibilité de le combiner au code flexionnel.

Ce code peut néanmoins présenter un désavantage, qui va dépendre des caractéristiques morphologiques de la langue traitée. Selon le nombre de voyelles contenues dans le mot, cette façon de procéder peut se révéler handicapante. En effet, lorsque le nom (surtout les noms composés) contient plusieurs voyelles et que le code se compose de deux ou trois voyelles à infléchir, il peut y avoir un problème. En luxembourgeois, nous n'avons pas encore rencontré ce type de problème, mais nous ne l'excluons pas. Nous avons fabriqué l'exemple suivant pour illustrer ce cas. Soit un nom comme **Karnevalstad* (ville de carnaval) et le code

suivant : *(e;a)ie)/…. La voyelle qui subit le plus souvent l'inflexion, le <e> dans ce cas, se retrouve en première position et sera donc remplacée en premier. Cette situation va donc produire une forme incorrecte étant donné que c'est le <a> de *Stad* (ville) qui doit être infléchi. Mais heureusement, cet inconvénient ne devrait pas être très fréquent pour les noms simples en luxembourgeois, puisque du moins d'après notre corpus, les noms au singulier ne contiennent qu'une voyelle ou une diphtongue. En revanche, ce problème pourrait se poser pour les noms composés, même si c'est peu probable en luxembourgeois.

4 Résultats

Dans le Lunaf, nous utilisons donc deux codes qui traitent chacun une caractéristique particulière de la morphologie des noms luxembourgeois. Dans cette communication, nous avons voulu présenter le code qui permet de traiter l'inflexion des noms luxembourgeois et qui de ce fait est appelé « inflexionnel ». Le codage utilisé dans notre dictionnaire électronique propose donc trois façons de déplacer le curseur dans le mot : lettre par lettre, en début ou en fin de mot et sur une lettre précise à l'intérieur du mot.

Grâce au code « inflexionnel » nous avons effectivement réduit le nombre de classes. De 92 classes flexionnelles nominales nous sommes passés à 77. Bien que les exemples de codes donnés plus haut ne se composent que de voyelles et de diphtongues, nous n'excluons pas le remplacement de consonnes à l'intérieur du mot lorsque les caractéristiques de la langue traitée s'y prêtent.

Ce code n'a pas été créé pour traiter exclusivement l'inflexion des noms de la langue luxembourgeoise. Il est tout à fait possible de l'utiliser dans d'autres langues qui présentent des alternances vocaliques du même genre comme par exemple les langues germaniques.

Références

CORNELL T. (1988), « IceParse : A model of inflectional parsing and word recognition for Icelandic ablauting verbs ». In : *Morphology as a Computational Problem*, Department of Linguistics, University of California, Los Angeles.

LUNA GARCIA F. (2002), « Particularités morphologiques du mot luxembourgeois et leur implémentation pour la génération automatique des mots ». In : *Bulletin Universitaire de Linguistique Générale et Appliquée* (BULAG), n°27, pp.151-166.

NEWTON G. (1996), *Luxembourg and Lëtzebuergesch*, Oxford, Clarendon Press.

POITOU J. (1987), « La flexion des substantifs en allemand II ». In : *Nouveaux Cahiers d'Allemand*, n°3, pp.299-314.

SCHANEN F. (1980), *Recherches sur la syntaxe du luxembourgeois de Schengen*, thèse soutenue à Paris IV.

SILBERZTEIN M. (1996), *INTEX 3.4 reference manual*, LADL, Université Paris 7.

SPROAT R. (1992), *Morphology and Computation*, Cambridge, The MIT Press.