TP 9— Cadavre(s) délicieux

http://www.gallium.inria.fr/~scherer/teaching/colles/2012/tp9.pdf

Les questions en rapport avec les TP, Caml ou la programmation en général sont les bienvenues et peuvent être envoyées à gabriel.scherer@gmail.com. N'hésitez pas !

1 Cadavre exquis : structure de la phrase

Connaissez-vous le jeu du cadavre exquis ? Il peut vous sauver la vie pendant certains cours traînant un peu en longueur, mais cela demande de la discrétion car il faut plusieurs participants. Pour éviter de vous faire surpendre pendant le passage de mains en mains de la précieuse feuille, vous pouvez décider de jouer seul, avec votre ordinateur. C'est ce que vous allez (devoir) faire maintenant.

La première approche, qui consiste à choisir des mots au hasard les uns à la suite des autres, ne produit pas des phrases ressemblant à du français. On va commencer par se fixer une structure grammaticalement correcte.

Dans cette partie, on cherche à produire au hasard une liste du genre [Nom; Verbe; Nom; Adjectif] décrivant la nature des mots de la phrase. Il n'y aura plus, ensuite, qu'à remplacer chaque nature par un mot convenable pour obtenir une phrase grammaticalement correcte (ou presque). Pour prendre en compte les accords entre les différents mots (ici le dernier Nom et son Adjectif doivent être accordés), on stocke aussi les informations de genre et de nombre.

Voici le type Caml qui sera utilisé pour représenter les natures. Recopiez-le dans votre code sans changer les noms ni mettre de minuscules, ce sera important ensuite. Pensez à mettre votre code dans un éditeur de texte (WordPad ou le bloc note, mais pas Word) à part, plutôt que directement dans CamlWin où il est difficile à récuperer et modifier ensuite, ce dont vous aurez sans doute besoin aujourd'hui.

```
type genre = Masc | Fem ;;
type nombre = Sing | Plur ;;

type nature =
  | Nom of (genre * nombre)
  | Verbe of nombre
  | Adjectif of (genre * nombre)
  | Mot of string ;;
```

▷ **Question 1.** Écrire des fonctions choisir_nombre : unit -> nombre et choisir_genre : unit -> genre qui choisissent un nombre et un genre au hasard. On pourra[1] pour cela écrire et utiliser une fonction choisir : 'a vect -> 'a choisissant au hasard un élément d'un tableau. ◁

▷ **Question 2.** Écrire une fonction gn : (genre * nombre) -> nature list générant un groupe nominal, c'est-à-dire un nom suivi avec une probabilité de 0.3 par un adjectif, respectant les accords (genre et nombre) donnés en paramètres. ◁

▷ **Question 3.** Écrire une fonction structure_phrase : unit -> nature list qui génère une phrase complète selon le schéma "Sujet - Verbe - COD". Attention à l'accord sujet-verbe. ◁

2 Mots au hasard

Pour passer d'une structure à une phrase complète, il faut remplacer chaque nature par un mot qui convient. On utilisera dans cette partie l'approche la plus simple possible : une table qui associe à chaque nature un tableau de mots (par exemple Nom (Masc, Sing) -> [| "eleve"; "retardataire" |]).

▷ **Question 4.** Écrire une fonction table_mots : nature -> string array qui associe à chaque nature un tableau de mots (de la bonne nature). C'est à vous de choisir son contenu, essayez d'avoir environ 5 à 10 mots par nature. ◁

▷ **Question 5.** Écrire une fonction remplir_mot : nature -> string qui choisit un mot de la bonne nature dans la table.

[1] On devra

Adapter remplir_mot pour ajouter devant les noms un article convenablement accordé, en utilisant une fonction auxiliaire choisir_article : string -> (genre * nombre) -> string qui prend en paramètre le nom et son accord. ◁

▷ **Question 6.** Écrire une fonction phrase_hasard () : unit -> unit qui génère une phrase aléatoire (en utilisant structure_phrase et remplir_mot) et l'affiche. ◁

3 PROLONGATION

En utilisant le constructeur Mot du type nature, on peut imposer une partie de la structure de la phrase (en rajoutant par exemple "qui", "quand" ou "avec" directement dans la structure).

▷ **Question 7.** Écrire une nouvelle fonction structure_phrase qui ajoute une subordonnée relative "qui" au groupe nominal sujet. L'adapter pour que tous les groupes nominaux générés ajoutent des relatives avec une probabilité de 0.3. ◁

▷ **Question 8.** Mettre en place d'autres structures de votre choix (phrases interrogatives, compléments circonstanciels...) et tester les phrases générées. ◁

▷ **Question 9.** Adaptez votre code pour produire des phrases en anglais. ◁