Projet informatique de licence (LI015), 1999-2000

Organisation

L'objectif du projet est la réalisation d'un programme relevant du Traitement Automatique des Langues. Les projets se font par **binômes** ou **trinômes**.

Sujets

Le sujet peut être choisi parmi les sujets proposés, ou il peut s'agir d'un sujet à votre initiative, auquel cas il faut en fournir une description rédigée d'une à deux pages, et obtenir l'accord de l'enseignant.

Langage

Les projets peuvent être réalisés en Pascal ou en C, exceptionnellement dans un autre langage de programmation, après accord de l'enseignant, sur la machine de votre choix, à condition toutefois qu'il soit possible d'en faire une démonstration le jour de la soutenance dans la salle machine.

Rapport

Un rapport comprenant une vingtaine de pages est demandé. Il doit être non manuscrit et doit contenir :

- une partie linguistique : description du champ d'étude, des données manipulées, des choix effectués, etc. ;
- une partie informatique : description des algorithmes mis en œuvre, des choix d'implémentation, des éventuels problèmes rencontrés et de leur solution, etc. ;
- un « manuel utilisateur » : indications pratiques pour exécuter le programme, et description de son interface (choix, menus...) ;
- en annexe (et seulement en annexe), listing du programme, éventuellement description exhaustive des données linguistiques, etc.

Soutenance

Les projets font l'objet d'une soutenance individuelle d'environ 15 mn, devant un jury d'un ou deux enseignants, au cours de laquelle l'étudiant proposera une démonstration du programme et répondra à quelques questions.

Calendrier (indicatif)

	Session de Juin	*Session de Septembre*
Début du projet :	nov. 99	*Id*
Remise du rapport :	lu. 19 juin 00, 10:00	lu. 18 sept. 00, 10:00
Soutenances :	26–30 juin 00	25–29 sept. 00

1 Correcteur orthographique (approche graphémique)

On se propose de réaliser un correcteur orthographique (lexical), qui, disposant d'un dictionnaire de formes fléchies, détecte les mots mal orthographiés, et propose si possible une correction.

Le programme prend un texte (ASCII brut) en entrée, et pour chaque forme non présente dans le dictionnaire, propose à l'utilisateur :

- de choisir un de ces remplaçants
  - pour l'occurrence, ou pour toutes les occurrences
- de fournir un remplaçant
  - pour l'occurrence, ou pour toutes les occurrences
  - si le remplaçant n'est pas dans le lexique,
    - de l'insérer dans le lexique pour ce texte,
    - de l'insérer dans le lexique stable
- d'ignorer la correction, c'est-à-dire conserver le mot initial, pour cette occurrence ou pour toutes, en l'insérant ou non dans le lexique.

Les remplaçants proposés par le programme (premier cas plus haut) seront des mots figurant dans le dictionnaire, et ayant une certaine proximité avec le mot fautif.

Pour la recherche des mots proches, on appliquera diverses heuristiques, basées sur la forme du mot :

- repérage des bigrammes ou trigrammes impossibles en français
- redoublement ou dé-doublement de consonnes
- suppression/insertion de diacritiques
- insertion de lettres
- interversion de lettres
- etc.

Ces heuristiques, qui peuvent être plus nombreuses, seront étudiées linguistiquement afin de déterminer précisément leurs conditions d'application.

2 Réaccentuation

Le programme est chargé de remettre les accents et autres signes diacritiques manquant dans un texte fourni en typographie dite pauvre.

On s'aidera d'un lexique de formes fléchies fourni. Bien sûr, certaines formes peuvent être réaccentuées directement, et d'autres sont ambiguës. Une fois que l'algorithme principal sera établi, on envisagera des heuristiques pour lever les ambiguïtés.

Exemple :
```
La ou le francais n'est pas accentue,
il y a de la gene,
mais quand le systeme m'accentue,
je suis moins gene!
```

On pourra s'inspirer du programme « Reacc » développé par le RALI à l'Université de Montréal : `http://www-rali.iro.umontreal.ca/Reacc/`.

3 Correction d'étiquettes

Disposant d'un corpus étiqueté et lemmatisé (chaque mot du corpus est accompagné de son étiquette grammaticale et du lemme lui correspondant — cf. figure 1), il s'agit d'écrire un programme permettant de corriger d'éventuelles erreurs d'étiquetage.

FIG. 1 – *Exemple de phrase du corpus*

Ce	DETRMS	ce
paragraphe	SUBSMS	paragraphe
résume	VER3B	résumer
les	DETRMP	les
problèmes	SUBMP	problème
qui	PRELMP	qui
doivent	VERB6	devoir
être	AUXE	être
résolus	PPASMP	résoudre
pour	PREP	pour
exploiter	VINF	exploiter
des	PDES	des
liaisons	SUBSFP	liaison
de	PDEA	de
télécommunication	SUBSFS	télécommunication
par	PREP	par
satellite	SUBSMS	satellite

À ce programme, l'utilisateur fournira des règles qui auront la forme suivante :

> *Changer l'étiquette Y_1 du mot X en Y_2 si les étiquettes Z_1 et Z_2 sont présentes dans le contexte de X, le contexte étant défini comme l'ensemble des cinq mots précédant et des cinq mots suivant X.*

Alors, le programme appliquera ces règles sur le corpus, en montrant à l'utilisateur les cas où la règle s'applique, de façon à ce qu'il puisse vérifier si la règle est pertinente.

Exemple : au vu de l'étiquetage fautif

l	l	DETRMS
'	'	x
usager	usager	SUBSMS
désire	désirer	VERB3
établir	établir	VINF
des	des	PDES
liaisons	liaison	SUBSFP

on pourra envisager une règle : L'étiquette PDES (=Y_1) du mot **des** (=X) devient DETRMP (=Y_2) si VERB3 (=Z_1) et SUBSFP (=Z_2) se trouvent dans le contexte du mot.

On envisagera la possibilité d'appliquer plusieurs règles lors d'une passe, et on choisira de préférence de stocker ces règles dans un fichier. On notera que l'ordre d'application des règles est important.

À titre d'exemple, vous proposerez un ensemble de règles qui permette de corriger les erreurs d'étiquetage sur "*des*".

On pourra envisager d'autres formes de règles.

4 Calcul de collocations

On veut réaliser un programme de calcul des collocations dans un corpus étiqueté et lemmatisé.

Le but est de repérer et de compter les blocs de 2 ou 3 mots juxtaposés qui se retrouvent au moins deux fois dans le corpus. On s'intéressera aussi aux groupes « N à N », « N de N », etc., dont on calculera aussi les fréquences.

La sortie du programme consiste donc en une liste, classée par fréquence, de toutes les collocations qui y apparaissent au moins deux fois.

On attachera un soin particulier au choix de la structure de données, et à l'algorithme, qui doit faire le calcul en un temps raisonnable.

5 Extraction de sigles

L'objectif est d'être capable de construire, semi-automatiquement, la liste des sigles (acronymes) (*par exemple EDF*), et de leur forme développée (*par exemple Electricité de France*), utilisés dans un texte donné. Le programme aura trois fonctions :

1. Création (extraction) de la liste (triée, normalisée) des sigles du texte donné. Attention aux formes diverses d'un même sigle (Inalf, I.N.A.L.F, INaLF).
2. Proposition (interactive) de syntagmes candidats pour la forme développée, quand on en trouve dans le texte.

Le texte fourni est en ASCII brut.

6 Justification et « hyphénation »

Il s'agit de réaliser un programme qui justifie (au sens des traitements de texte) un texte fourni en ASCII, sur un nombre de colonnes donné, en découpant les mots selon les règles en usage pour le français.

On supposera que tous les caractères ont la même dimension (« fonte proportionnelle »).

Pour le découpage (éventuel) des mots (*hyphenation*), on prendra bien garde de distinguer les règles, qui seront stockées dans un fichier, du programme lui-même. On s'autorisera à utiliser un dictionnaire d'exceptions.

7 Concordancier

Réalisation d'un « concordancier » : disposant d'un corpus étiqueté et lemmatisé (chaque mot du corpus est accompagné de son étiquette grammaticale et du lemme lui correspondant — cf. figure 1), il s'agit d'extraire les phrases de ce corpus comportant des séquences d'étiquettes et/ou de lemmes données.

Les séquences

Les séquences sont des suites d'étiquettes et/ou de lemmes, et comportant éventuellement des symboles ayant une interprétation particulière (cf. expressions rationnelles) :

- `*` signifie zéro ou plusieurs occurrences de l'entité (étiquette grammaticale ou lemme) précédente,
- `+` signifie une ou plusieurs occurrences de l'entité précédente.

Voici quelques exemples de séquences accompagnées de leur interprétation :

`SUBS ADJ`	un substantif, suivi d'un adjectif
`SUBS ADJ* PREP`	un substantif, suivi d'un nombre quelconque d'adjectifs, suivi d'une préposition
`SUBS ADJ+ PREP`	un substantif, suivi d'au moins un adjectif, suivi d'une préposition.

Exemple

Considérons la phrase de la figure 1 (p. 3). Cette phrase doit être extraite si la séquence recherchée est `VINF des SUBS` mais ne doit pas l'être si la séquence recherchée est `SUBS ADJ des SUBS`.

Extraction des phrases

Le programme propose à l'utilisateur de saisir une séquence (on rappelle que cette séquence peut comprendre des étiquettes et des lemmes), extrait les phrases qui comprennent cette séquence, et les stocke dans un fichier (on envisagera aussi leur visualisation).

À titre d'exemple, si l'on s'intéresse au lemme *des*, on pourra demander au programme d'extraire du corpus les phrases correspondant aux séquences suivantes :

1. `VINF des SUBS`
2. `PREP VINF des ADJ* SUBS`
3. `PREP des ADV* ADJ* SUBS`
4. `SUBS ADJ+ des ADV* ADJ* SUBS`

Remarques

- les étiquettes grammaticales utilisées dans les séquences sont plus générales que celles utilisées dans le corpus. En particulier, aucune information de genre et de nombre n'est donnée dans ces séquences,
- les prépositions "à" et "de" sont étiquetées *PDEA* dans le corpus sur lequel vous travaillerez mais doivent être considérées comme des prépositions.

Remerciements

Merci à Lionel Clément, Marcel Cori, Laurence Danlos, Eric Gaussier, et Thierry Poibeau pour leurs propositions et/ou relectures des sujets de projets.

1.2, 11/99