Projet informatique de maîtrise (LI036), 1999-2000

Organisation

L'objectif du projet est la réalisation d'un programme informatique 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 Lisp, en Prolog ou en Java, voire en Pascal, en C ou en C++, 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 trentaine 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 :	déc. 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 Repérage des « entités nommées »

On regroupe sous le terme « entités nommées » les noms de personnes, de lieux, les dates, etc. Le but du projet consiste à repérer automatiquement un sous-ensemble homogène parmi les entités (par exemple les noms de personnes ou les noms d'entreprises).

Le programme envisagé procède par étapes successives sur le même texte. Dans une première étape, grâce aux dictionnaires spécialisés (*cf. plus loin*), on étiquette les mots simples. On partira d'un corpus déjà étiqueté morpho-syntaxiquement. La seconde étape, basée sur des règles, permettra de regrouper ces mots simples pour former des entités nommées. Enfin, une troisième étape, facultative dans ce projet, pourrait tenter de relier entre elles les entités co-référentes. Les éléments seront marqués dans le texte au moyen d'un système de balisage de type SGML.

Le système repose donc, outre le programme lui-même, sur les ressources suivantes :

- des dictionnaires :
  - dictionnaire de la langue générale pour repérer les mots inconnus.
  - dictionnaires spécialisés de prénoms, de noms de sociétés connues, etc.
  - dictionnaires spécialisés d'« amorces », mots ou groupes de mots qui peuvent marquer le début d'une entité nommée.
- une « grammaire » permettant de combiner des mots repérés de natures diverses (mots inconnus, prénoms, amorces...) pour en faire une entité (un nom de personne).

Les dictionnaires et la grammaire sont stockés dans des fichiers textes. On assure ainsi une séparation nette entre les données et les traitement (le programme).

Exemple Soit le texte « *M. Jack Lang a gagné la mairie.* »

"M." est stocké dans un dictionnaire "d'amorces"
"Jack" est stocké dans un dictionnaire de prénoms
"Lang" est inconnu du système

La première étape permet d'obtenir :
<TR_PERSON>M.</TR_PERSON> <PERSON>Jack</PERSON>
<UFIRSTUNKNOWN>Lang</UFIRSTUNKNOWN> a gagné la mairie.

Une règle de la forme : TR_PERSON PERSON UFIRSTUNKNOWN ==> PERSON va pouvoir être activée.

D'où le résultat de la seconde étape : <PERSON>M.Jack Lang</PERSON> a gagné la mairie.

Une troisième étape permettrait d'identifier toutes les occurrences de Jack Lang
<PERSON id=1>M.Jack Lang</PERSON> a gagné la mairie.

Sous réserve Je devrais pouvoir fournir une partie des dictionnaires à utiliser. Il est souhaitable que le système stocke dans un fichier les mots inconnus susceptibles d'être des entités, pour pouvoir améliorer rapidement et dynamiquement la couverture.

2 Manipulation d'automates

On se propose de réaliser un programme, ou mieux, une *bibliothèque de fonctions*, qui permette de réaliser divers traitements sur des automates à nombre fini d'états.

On peut envisager une interface graphique (pour la sortie et même pour l'entrée des automates), mais on pourra aussi se contenter de travailler sur des tables de transition en ASCII, dans une notation *à la* PC-Kimmo. (Il faudra bien sûr envisager une structure de données interne pour travailler sur les automates.)

Parmi les opérations que l'on devra pouvoir réaliser avec le système, il y aura au moins :

- Repérage de la catégorie de l'automate : complet ou non, déterministe ou non, avec ou sans ε-transitions...
- Complétion d'un automate
- Déterminisation d'un automate
- Minimisation d'un automate
- Suppression des ε-transitions d'un automate

  Facultatif :
- Traduction d'une expression rationnelle en automate
- Traduction d'un automate en expression rationnelle

3 Réseaux sémantiques

À partir de définitions provenant de dictionnaires électroniques, préalablement étiquetées, il s'agit de construire un « réseau sémantique ».

Pour chaque entrée (qui peut correspondre à un des sens d'une lexie), on repèrera les mots pleins qui participent à sa définition (il faut donc (1) distinguer les mots pleins des mots « outils », (2) repérer la définition proprement dite parmi l'ensemble des informations associées à une entrée (catégorie, exemples, synonymes...)), et on construira un réseau reliant l'entrée à tous ces mots pleins.

Ce réseau pourra être stocké sous forme de fichier Ascii, dans un format du genre de celui de WordNet. On pourra aussi, éventuellement, proposer une interface graphique permettant de visualiser graphiquement le réseau, voire de le modifier.

Dans un deuxième temps, on réalisera un programme exploitant ce réseau pour désambiguïser les sens d'un mot en contexte. Principe : étant donnée une phrase contenant le mot concerné, on repère les mots pleins qui l'entourent (contexte), et on recherche ces mots dans le réseau. Alors le sens correspondant est celui qui est le plus *proche* (dans un sens qu'il faut précisément définir) des mots pleins de son contexte.

4 Génération de texte en LFG (lisp)

Le programme doit pouvoir afficher la ou les phrases correspondant à une structure fonctionnelle supposée bien formée (cohérente et complète).

Le programmeur fournira une petite grammaire LFG pour la langue de son choix.

1. Première étape : Écrire un programme qui génère le langage correspondant à une grammaire CF quelconque. Le programme pourra «bogger» en cas de récursion gauche. On pourra produire par exemple un réseau d'automates.
2. Seconde étape : Compléter ce programme pour tenir compte des équations fonctionnelles. Seul le langage correspondant à la structure fonctionnelle devra être produit. Procéder par étapes :
   - analyser les équations du type $\downarrow = (\uparrow \text{ obj})$
   - ensuite les equations du type $\downarrow = (\uparrow (\downarrow \text{ pcas })\text{-obj})$
   - ensuite les équation contraintes
   - ensuite les equations existentielles
3. Troisième étape : Retrouver les formes fléchies dans un dictionnaire à partir des lemmes et informations morphologiques des mots.

5 « Complétion » automatique

On se propose de réaliser un programme qui accélère la vitesse de frappe d'un utilisateur en proposant — de façon intelligente — la fin des mots en cours de frappe.

Un tel système, pour être vraiment performant, devrait faire de la prédiction syntaxique en temps réel. On se contentera dans ce projet de faire un système qui propose la suite la plus probable des mots.

La méthode repose sur un calcul de probabilité basé sur la fréquence d'apparition de chaque mot dans un corpus préalablement fourni. On construit un graphe dont chaque chemin correspond à un mot (une lettre par arc), et dont chaque arc porte une probabilité.

Par exemple, la probabilité de la prochaine lettre de la suite "v-o-l-u" est respectivement de 0/9, 8/9 ($= \frac{3+5}{9}$), 1/9, 0/9 pour les lettres 'b', 'm', 'p' et 't' si nous avons dans le corpus :

0	occurrences du mot	volubile
3	"	volume
5	"	volumineux
1	"	voluptueux
0	"	volute

La lettre la plus probable, ici "m", sera proposée à l'utilisateur, et un nouveau calcul est effectué pour la suite de lettres "v-o-l-u-m".

Un soin particulier sera apporté à l'interface de ce programme : l'utilisateur doit pouvoir accepter ou revenir facilement sur les propositions du système, etc.

Un langage particulièrement adapté au projet serait Emacs Lisp, ce qui permettrait d'intégrer la complétion à l'éditeur Emacs.

6 Désambiguisation par automates

Partant d'un texte dont chaque mot est étiqueté avec toutes ses catégories morpho-syntaxiques possibles, on va s'attacher à lever certaines ambiguïtés non pertinentes, c'est-à-dire à supprimer certaines des étiquettes attachées à certains mots.

Pour mener à bien cette tâche, on extraiera du corpus donné les éléments pertinents, ce qui permettra d'établir des « règles contextuelles », que l'on appliquera ensuite au corpus.

1. Étude linguistique

On s'intéressera à une catégorie donnée, par exemple « verbe conjugué ». On extraira du texte une première liste de mots qui peuvent être des verbes conjugués (ceux dont la liste d'étiquettes comprend la catégorie V), puis, après désambiguisation, une deuxième liste comprenant seulement les mots qui sont effectivement des verbes conjugués dans le texte.

On relèvera alors les contextes locaux (catégories précédentes, catégories suivantes) qui permettent de choisir la catégorie retenue pour un mot donné en français.

Par exemple, si X est marqué N et V, et si X est précédé d'un Det, alors X doit être marqué N (l'étiquette V doit être ôtée).

Etablir ainsi des séquences de 2 ou 3 catégories impossibles en français.

On aura intérêt à subdiviser certaines catégories du DELAF, par exemple mettre à part parmi les pronoms, les formes clitiques (*je*, *tu*, *il(s)*, *elle(s)*, *nous*, *vous*, *se*, *on*, *ce*, *y*, *en*, *le*, *la*, *les*, *lui*, *leur*).

Attention au cas où les mots suivants ou précédents ont plusieurs catégories possibles.

2. Programmation et implémentation

Une fois les règles dégagées, on réalisera un programme qui procédera à la désambiguisation du corpus en appliquant les règles. On conseille d'implémenter les règles sous forme de transducteur.

On fournit deux textes étiquetés (d'une dizaine de pages) à l'aide du DELAF, qui serviront de base à l'élaboration des règles de désambiguisation.

Notations du DELAF :

V : verbe
N : nom
DET : déterminant (article)
DETP : déterminant possessif
DETQ : déterminant numéral
INTJ : interjection
CNJS : conjonction de subordination
CNJC : conjonction de coordination
A : adjectif
PRON : pronom
ADV : adverbe
Xin : préfixe

Pour les verbes :
P : présent indicatif
S : subjonctif présent
Y : impératif
K : participe passé
F : futur
W : infinitif
I : imparfait
C : conditionnel
G : participe présent
T : subjonctif imparfait
Inv : invariable
1, 2 ou 3 : personne
m, f, s, p : genre, nombre

Exemple Forme générale : *mot lemme cat:morpho ...*

```
vient  venir V33:P3s
livre  livre N1:ms  livre N21:fs  livrer V3:P1s:P3s:S1s:S3s:Y2s
```

7 Compression de texte

On doit réaliser un couple de programmes qui permette de compresser-décompresser un texte.

Le principe de l'algorithme de compression est le suivant : sur des bases statistiques, en fonction du texte en entrée, on détermine des chaînes les plus longues possible (en particulier préfixes et suffixes) présentes avec une fréquence raisonnable, et on remplace dans le texte toutes les occurrences de ces chaînes par des non-mots — ou des mots absents du texte — (p. ex. « aa »), plus courts.

Bien sûr, le fichier compressé contiendra aussi une table de conversion, qui permettra au programme de décompression de reconstituer le fichier original.

Remerciements

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

v1.2 12/99