LoX[1] : outil polyvalent pour l'exploration de corpus annotés

Laurent AUDIBERT

DELIC – Université de Provence
29 Avenue Robert SCHUMAN
13621 Aix-en-Provence Cedex 1
laurent.audibert@up.univ-aix.fr

Résumé – Abstract

Cet article présente une application permettant d'écrire des requêtes complexes sur des corpus étiquetés et de formater librement les résultats de ces requêtes. Le formalisme des requêtes est basé sur le principe des expressions régulières bien connu de la plupart des linguistes travaillant sur des corpus écrits. Contrairement à certains logiciels, qui ne permettent que l'extraction de concordances au format relativement figé, le formatage libre du résultat des requêtes permet leur réutilisation par des programmes ultérieurs et autorise une grande diversité d'applications, s'écartant largement du cadre des simples concordanciers.

This paper describes a tool that enables complex queries on tagged corpora, and free formatting of the results. The formalism used is based on regular expressions, which are well-known from most corpus linguists. As opposed to other software, the free formatting of the results enables re-use of the query results by additional tools, and proves useful for a wide range of applications well beyond that of simple concordance programs.

Mots clés – Keywords

Corpus, Concordancier, TAL, Parser, Expression régulière.

Corpora, Concordancer, NLP, Parser, Regular expression.

1 Introduction

Il existe maintenant des applications permettant d'étiqueter de manière efficace des corpus écrits (par exemple, le logiciel Cordial, développé par la société Synapse Développement,[2]

[1] LoX est disponible sur la page Web *http://laurent.audibert.free.fr/lox.htm*

[2] Voir le site Web http://www.synapse-fr.com/

offre une lemmatisation et un étiquetage morpho-syntaxique relativement fiable). Il est alors plus pertinent pour les utilisateurs (linguistes, lexicographes, terminologues, etc.), d'interroger un corpus en écrivant des requêtes ne portant plus uniquement sur des mots mais aussi sur des lemmes ou des étiquettes morpho-syntaxiques.

Les linguistes travaillant sur des corpus écrits sont généralement familiers avec le formalisme des expressions régulières. Ces expressions se situent au niveau des caractères et permettent de décrire de manière formelle des classes de chaînes de caractères. Le langage de requête que nous présentons ici est basé sur des méta-expressions régulières. Ces expressions se situent au niveau des mots et permettent de décrire de manière formelle des classes de suites de mots, donc des portions de texte. Ainsi, ce langage de requête ne devrait pas dérouter le linguiste.

Le but premier d'un langage de requête est de diminuer de manière significative la quantité d'information à traiter manuellement. Mais il peut aussi s'avérer intéressant d'effectuer des dénombrements ou encore de générer des résultats adaptés à des besoins particuliers. Aussi, au contraire de langages précédemment proposés, par exemple par (Christ, 1994), le langage de requête que nous proposons contient un micro-langage de formatage du résultat qui permet de préciser librement la forme que prendra le résultat des requêtes.

2 Types de corpus

L'application LoX reconnaît actuellement deux types de corpus : les corpus de type texte brut et les corpus étiquetés de format tabulaire[3].

2.1 Corpus de type texte brut

Un corpus de type texte brut est un corpus au format texte contenant le texte dépourvu de mise en forme typographique, et sans aucun ajout d'étiquette ou d'annotation. Voici un exemple de texte brut :

```
La construction de centaines de digues et barrages entraînera le détournement de plusieurs fleuves, la disparition de plus de mille lacs naturels et l'abattage d'arbres centenaires.
```

La segmentation en « mots » se fait suivant le principe de la segmentation maximale, respectant ainsi les principes édictés par le projet GRACE (Adda, Mariani, Paroubek, Rajman, & Lecomte, 1999) (i.e. les ponctuations, l'apostrophe, etc. sont des mots distincts).

2.2 Corpus tabulaire

Ces corpus sont de forme tabulaire et au format texte. Chaque ligne correspondra à un élément (i.e. un mot) tandis que chaque colonne correspondra à une propriété de l'élément, c'est-à-dire un étiquetage particulier (catégorie morpho-syntaxique, lemmatisation, ou autre). La fin d'une

---

[3] D'autres formats sont envisageables et peuvent être facilement ajoutés (grâce aux mécanismes d'abstraction de la programmation orientée objet) dans des versions ultérieures, en particulier, le format SGML ou XML proposé par la *Text Encoding Initiative* (Sperberg-McQueen & Burnard, 1994).

ligne est identifiée par un délimiteur particulier (retour chariot). De même, un autre délimiteur particulier sépare les colonnes (tabulation).

Ci-dessous est représenté un extrait du journal *Le Monde* étiqueté par le logiciel Cordial.

```
La           → le           → DETDFS   → Da-fs-d ↵
construction → construction → NCFS     → Ncfs    ↵
de           → de           → PREP     → Sp      ↵
centaines    → centaine     → NCFP     → Ncfp    ↵
de           → de           → PREP     → Sp      ↵
digues       → digue        → NCFP     → Ncfp    ↵
et           → et           → COO      → Cc      ↵
barrages     → barrage      → NCMP     → Ncmp    ↵
entraînera   → entraîner    → VINDF3S  → Vmif3s  ↵
le           → le           → DETDMS   → Da-ms-d ↵
détournement → détournement → NCMS     → Ncms    ↵
de           → de           → PREP     → Sp      ↵
plusieurs    → plusieurs    → ADJIND   → Dt-.p-  ↵
fleuves      → fleuve       → NCMP     → Ncmp    ↵
,            → ,            → PCTFAIB  → Ypw     ↵
la           → le           → DETDFS   → Da-fs-d ↵
```

```
disparition  → disparition  → NCFS     → Ncfs    ↵
de           → de           → PREP     → Sp      ↵
plus         → plus         → ADV      → Rgn     ↵
de           → de           → PREP     → Sp      ↵
mille        → mille        → ADJNUM   → Mc.p    ↵
lacs         → lacs         → NCMIN    → Ncmp    ↵
naturels     → naturel      → ADJMP    → Afpmp   ↵
et           → et           → COO      → Cc      ↵
l            → le           → DETDMS   → Da-ms-d ↵
'            → //           → //       → //      ↵
abattage     → abattage     → NCMS     → Ncms    ↵
d            → de           → PREP     → Sp      ↵
'            → //           → //       → //      ↵
arbres       → arbre        → NCMP     → Ncmp    ↵
centenaires  → centenaire   → ADJPIG   → Afpmp   ↵
.            → .            → PCTFORTE → Yps     ↵
```

2.3 Nom des propriétés

Les noms de propriétés sont des intitulés qui permettent de faire référence à une colonne particulière pour un mot donné (i.e. à une propriété ou une étiquette de ce mot) dans une requête ou dans le micro-langage de formatage du résultat (masque).

Un fichier de type texte brut est considéré comme un fichier ne comportant qu'une colonne contenant le texte segmenté (un mot par ligne). On fera référence à un mot avec le mot clef « word » (nom de l'unique propriété correspondant à l'unique colonne).

Pour utiliser un corpus de type tabulaire il faudra nommer chacune des colonnes du fichier. Voici les noms de propriétés correspondant à l'exemple précédent[4] :

```
mot → lemme → typegram → codespe ↵
```

3 Syntaxe des requêtes

3.1 Requête sur les mots isolés : meta-expression régulière atomique

Une méta-expression régulière atomique (MERA) est une formule logique, sur les noms de propriétés, les chaînes de caractères et les expressions régulières. Un mot est décrit par la MERA si l'expression logique associée est « vrai » pour le mot en question. Un exemple de MERA est : *[lemme="entraîner" & typegram~"^VIND"]*. Cette MERA décrit tous les mots dont l'étiquette « lemme » est « entraîner » et l'étiquette « typegram » commence par « VIND ». Il s'agit donc d'une recherche du verbe « entraîner » conjugué à l'indicatif.

[] est une MERA spéciale décrivant n'importe quel mot. En fait *[]* est équivalent à *[true]*.

[4] « Typegram » et « codespe » sont deux étiquetages morpho-syntaxiques différents produits par le logiciel Cordial. Nous reprenons les dénominations de ce logiciel.

3.2 Répétiteur de MERA

Il est possible d'adjoindre un répétiteur pour répéter la MERA un certain nombre de fois. Un répétiteur est de la forme *{x, y}*, avec $x \leq y$, où *x* est le nombre minimum de fois où la MERA doit être répétée et *y* le nombre maximum de fois. Ainsi *[typegram~"^NC"]{1,3}* décrira une suite de un à trois noms communs consécutifs. Si dans le corpus se trouvent deux noms communs consécutifs, cette expression décrira trois portions : la première correspondant au premier nom commun, la seconde correspondant aux deux noms communs et la dernière correspondant au second nom commun.

Il existe aussi des répétiteurs prédéfinis :

- « * », pour dire un nombre quelconque de fois, est équivalent à *{0, LONG_MAX*[5]*}* ;
- « + » , pour dire au moins une fois, est équivalent à *{1, LONG_MAX}* ;
- « *?* » , pour dire zéro ou une fois, est équivalent à *{0, 1}*.

Un répétiteur particulier est aussi disponible, sa forme est *{x,y}* avec *x>y*. Ce répétiteur (que nous nommerons ***répétiteur préférentiel***) peut être lu comme : la MERA doit être répétée *x* fois si possible, sinon *x-1* fois, sinon *x-2* fois, … mais pas moins de *y* fois. Par exemple, supposons la séquence de mots « *x a b b b y* ». Sur cette séquence, la requête *[a][b]{1,4}* retournera trois correspondances *{ab, abb, abbb}* tandis que la requête *[a][b]{4, 1}* n'en retournera qu'une *{abbb}*.

Il est enfin possible de nommer une MERA pour faire référence ultérieurement au mot décrit par celle-ci : *cible:[lemme= "entraîner" & typegram~"^VIND"]*. Dans cet exemple, on pourra ultérieurement (ie. dans la suite de la requête ou dans un masque) faire référence au verbe décrit par la MERA en utilisant la variable « cible ».

L'ensemble constitué par la MERA, sa variable et son répétiteur associés sera appelé meta-expression régulière élémentaire (MERE).

3.3 Requête sur des groupes de mots

Une requête peut être vue comme une expression régulière sur les MERE. Il est donc possible de juxtaposer une ou plusieurs MERE pour décrire des suites de mots mais il est aussi possible d'utiliser les parenthèses « *(* » et « *)* » un opérateur de disjonction « | » un opérateur de conjonction « *&* » ainsi que les répétiteurs décrits dans la section précédente (section 3.2).

De plus, une requête peut se terminer par la spécification d'une clause stop. Cette clause permet de préciser une condition qui interrompt la recherche et rejette la tentative de correspondance en cours.

Par exemple, la requête identifiant les verbes conjugués positionnés avant un verbe à l'infinitif, au sein d'une même phrase, pourra s'écrire :

---

[5] LONG_MAX est égal au plus grand entier que la machine peut représenter.

[type_gram~"^V" & type_gram!~"^VINF|^VPARP|^VPARPRES"] [type_gram!~"^V"] [type_gram~"^VINF"] stop (type_gram="PCTFORTE")*

où la première MERA désigne un verbe qui n'est ni à l'infinitif, ni au participe présent ni au participe passé, la seconde une suite de mots sans verbe, et la dernière un verbe à l'infinitif. La clause stop précise que le résultat ne doit pas contenir de ponctuation forte.

3.4 Evaluation et optimisation d'une requête

La requête suivante décrit des n-grammes, avec $n \leq 5$, contenant le lemme « chef », ne contenant pas de ponctuation et commençant et finissant par un nom commun, un adverbe, un verbe ou un adjectif (cf. section 5.3 pour un exemple de résultat).

([type_gram~"(^NC|^ADV|^V|^ADJ)"][type_gram!~"^PCT"]{0,3})?
[lemme="chef"]
([type_gram!~"^PCT"]{0,3}[type_gram~"(^NC|^ADV|^V|^ADJ)"])?
stop((end.index-begin.index)>4)

Cette requête sera analysée de gauche à droite. Le corpus sera également parcouru de gauche à droite. Ainsi, la requête va tout d'abord tenter de décrire une portion du corpus commençant par le premier mot du corpus, puis par le second, … La première MERE (*[type_gram~"(^NC|^ADV|^V|^ADJ)"]*) de la requête sera souvent vérifiée car de nombreux mots sont un nom commun, un adverbe, un verbe ou un adjectif. La seconde MERE (*[type_gram!~"^PCT"]{0,3}*) décrira généralement quatre portions de corpus si les trois mots suivant le mot décrit par la première MERE ne sont pas des ponctuations (le cas est également très fréquent). Ce n'est que sur la troisième MERE de la requête (*[lemme="chef"]*) que l'application de la requête va généralement échouer (un mot ayant pour lemme chef est un phénomène rare dans un corpus). Ainsi, appliquée sur un corpus de taille importante, l'évaluation de cette requête sera lente. Il serait plus pertinent de rechercher les mots dont le lemme est « chef » puis de continuer l'application de la requête autour de ces mots. Pour cela, il suffit d'encadrer par les délimiteurs « < » et « > » la partie de la requête devant être évaluée en premier. La requête devient donc :

([type_gram~"(^NC|^ADV|^V|^ADJ)"][type_gram!~"^PCT"]{0,3})?
<[lemme="chef"]>
([type_gram!~"^PCT"]{0,3}[type_gram~"(^NC|^ADV|^V|^ADJ)"])?
stop((end.index-begin.index)>4)

Dans une telle requête, la partie entre « <> » est évaluée en premier. Si l'évaluation réussit, la partie suivante est évaluée de manière classique (de gauche à droite) tandis que la partie précédente est évaluée à rebours (de droite à gauche). Le sens de l'évaluation est important dans le cas de l'utilisation du répétiteur préférentiel.

3.5 Cas particulier de l'utilisation du répétiteur préférentiel

Les deux requêtes que nous venons de voir ne s'évaluent pas dans le même intervalle de temps sur un corpus (la seconde est bien plus rapide) mais elles retournent cependant le même résultat. Il n'en va pas toujours de même lorsque l'on ajoute les délimiteurs « < » et « > » dans une expression contenant un répétiteur préférentiel *{x,y}* avec $x>y$. Par exemple, supposons la séquence de mots « *x b b b a b b y* ». Sur cette séquence, la requête *[b]{4,1} [a] [b]{4,1}*

retournera trois correspondances *{bbbabb, bbabb, babb}* tandis que la requête *[b]{4,1} <[a]> [b]{4,1}* n'en retournera qu'une *{bbbabb}* en raison de l'évaluation à rebours de la partie à gauche du délimiteur « < ».

4 Formatage des résultats (masque)

Une requête permet de décrire de manière formelle un phénomène linguistique. Une instance de ce phénomène dans un corpus, c'est à dire une portion de corpus décrite par la requête sera appelée une correspondance. A une correspondance sont associées deux positions dans le corpus (début et fin de la correspondance) ainsi qu'un certain nombre de références à des mots de la correspondance (cf. fin de la section 3.2). Les masques permettent à l'utilisateur de préciser comment la correspondance doit être utilisée pour générer une chaîne de caractères.

Dans un masque, pour produire un caractère, il suffit de taper ce caractère. On peut aussi produire un caractère en utilisant son code ASCII, pour cela, il faut saisir *[C:<code_ASCII>]*. On peut afficher un nombre grâce à *[N:<nombre>]*. On peut faire référence à une propriété d'un mot en saisissant *[P:<variable>]*. La syntaxe des masques comporte également une structure de boucle permettant de générer de manière itérative un masque : *[B:<masque>;<debut_itération>;<fin_itération>]* et une structure de choix *[?:<expression_logique>;<masque>;<masque>]*.

Par exemple le masque *[B: [P:mot][C:32] ; begin.index ; end.index]* permet de générer une chaîne contenant tous les mots de la correspondance séparés par un espace. « *begin* » et « *end* » sont deux variables prédéfinies qui pointent respectivement sur le premier et le dernier mot de la correspondance.

5 Trois utilisations possibles

Le logiciel LoX permet trois types d'exploitation des requêtes et de leurs masques associés.

5.1 Extraction de sous-corpus

La première utilisation ignore les masques et génère un sous-corpus constitué de toutes les portions du corpus sources décrites par la requête. Cette utilisation permet de produire un sous-corpus dans lequel le phénomène que l'on désire observer sera dense. On ramène ainsi le corpus de grande taille sur lequel on doit travailler à un corpus de taille raisonnable.

5.2 Applications des masques

La deuxième utilisation consiste à générer un fichier constitué des chaînes générées par l'application des masques aux correspondances issues de l'application de la requête sur le corpus.

Exemple : fichier pour un étiquetage sémantique manuel

Le but est de générer un fichier permettant un étiquetage sémantique manuel aisé des occurrences du lemme « barrage » dans un grand corpus.

La requête sera donc : *cible:[lemme="barrage"]*

Le masque associé sera :

[B:[P:mot][C:32];cible.index-50;cible.index-1][C:9]
[P:cible.mot][C:9]
[C:9]
[B:[P:mot][C:32];cible.index+1;cible.index+50][C:9]
[B:[P:mot][C:32];cible.index-1;cible.index-5][C:10]

Voici un extrait du fichier généré (les contextes gauche et droit ont été tronqués) :

Contexte gauche	Mot	Eti	Contexte droit	Clef Gauche
communistes feront systématiquement	barrage		aux candidats de droite.	systématiquement " feront
gardiens sont répartis en trois "	barrages		" distants d ' une centaine	" trois en répartis
que les castors construisent des	barrages		(exemple que nous empruntons	des construisent castors
s ' adresse aux gardiens des	barrages		, aux délégués locaux , aux	des gardiens aux adresse
inquiétantes courent de barrage en	barrage		: Il s ' adresse aux	en barrage de courent
« Les castors construisent des	barrages		» fait partie du stéréotype	des construisent castors
voulu prendre le risque de faire	barrage		à ceux des rares élus locaux	faire de risque le

Pour un exemple réel de ce type, voir (Reymond, 2001).

5.3 Dénombrement des masques

L'application des masques aux correspondances, reconnues par les requêtes, produit un ensemble de chaînes. Il est possible de générer un tableau permettant de dénombrer le nombre d'occurrences de chaque chaîne dans l'ensemble des chaînes générées. De plus, lorsqu'un masque discriminant est généré, il est possible de produire un tableau de dénombrement à deux dimensions.

Exemple : recherche de figements

Le but est de dénombrer les n-grammes ($n<5$) contenant le lemme « chef » dans un grand corpus. Nous utiliserons la requête de la section 3.4, le masque associé sera :

[B:[P:mot]_;begin.index;end.index-1][P:end.mot]

Voici un extrait du fichier généré :

chef_-_d_'_œuvre	42
chefs_de_publicité	41
chef_de_file	36
premier_chef	25
chefs_d_'_entreprise	25
chef_du_gouvernement	24
rédacteur_en_chef	21

Exemple : dénombrement avec discriminant

Il est possible de générer un deuxième masque servant de discriminant, pour obtenir un dénombrement à deux dimensions. Le tableau qui suit est extrait d'un fichier résultant de l'application de plusieurs requêtes centrées sur le lemme « barrage » et appliquées sur un corpus dont le lemme barrage a été étiqueté sémantiquement. Le masque discriminant renvoie l'étiquette du sens du lemme « barrage ». Ce tableau permet d'observer la répartition des phénomènes, formalisés par les règles, qui sont en cooccurrence avec le lemme « barrage » en fonction du sens de ce lemme.

	1.1	**1.2**	**2.1**	**2.2**
Coo_Phr_castor			7	
pos(-1)_DETDMS		1	9	1
coo_faire	12		1	1
Coo_Phr_construction			9	
Coo_Phr_construire			15	
Coo_Phr_exemple			14	
pos(-1)_PREP			2	9
2-gramme(faire_barrage)	12			
3-gramme(barrage_au_développement)		7		
3-gramme(construisent_des_barrages)			9	
5-gramme(courent_de_barrage_en_barrage)				6
Coo_Phr_dommage		7		
Coo_Phr_droite	6			
Coo_Phr_développement		7		

6 Perspectives

Les informations générées par un traitement adéquat, sur un corpus préalablement étiqueté sémantiquement, peuvent permettre de rechercher quels sont les phénomènes, en cooccurrence avec un mot donné, les plus indicateurs du sens de ce mot. Un traitement statistique subséquent pourrait alors permettre une désambiguïsation sémantique automatique. C'est dans cette perspective que s'inscriront les développements ultérieurs de l'application. Un développement fort intéressant, notamment dans la perspective de désambiguïsation, serait de pouvoir utiliser des bases de connaissance externes. En effet des sources d'informations sont actuellement disponibles, comme WordNet (Miller, Beckwith, Fellbaum, Gross, & Miller, 1993), EuroWordNet (Vossen et al., 1998). Elles permettent d'augmenter considérablement l'information disponible sur les mots. Il serait intéressant de pouvoir utiliser de telles sources d'informations dans les requêtes.

L'outil développé est extrêmement ouvert et permet une grande diversité d'applications en raison du grand pouvoir expressif des requêtes et de leur libre interprétation au travers des masques. Il est ainsi possible, dans le cadre de projets d'étudiants par exemple, de développer des frontaux graphiques permettant une utilisation plus simple, plus intuitive, mais aussi plus limitée des possibilités de l'application (par exemple une utilisation de type concordancier, dénombrement de phénomènes bien particuliers, …).

7 Conclusions

Nous avons présenté une application permettant d'écrire des requêtes complexes sur des corpus étiquetés et de formater librement les résultats de ces requêtes. Le formalisme des requêtes est basé sur le principe des expressions régulières bien connu de la plupart des linguistes travaillant sur des corpus écrits. Contrairement à certains logiciels, qui ne permettent que l'extraction de concordances au format relativement figé, le formatage libre du résultat des requêtes permet leur réutilisation par des programmes ultérieurs et autorise une grande diversité d'applications, s'écartant largement du cadre des simples concordanciers. Cette application se rapproche du logiciel INTEX développé au LADL (Silberztein, 2000) mais est plus orienté vers le traitement de corpus étiquetés (notamment morpho-syntaxiquement désambiguïsé). LoX existe dans une version ligne de commande pouvant être compilée sur n'importe quelle plate-forme et dans une version disposant d'une interface graphique pour une utilisation plus souple sous Windows.

Du fait du grand pouvoir expressif des requêtes et de leur libre interprétation au travers des masques LoX constitue un outil puissant et polyvalent pour tout linguiste ou lexicographe travaillant sur des corpus écrits. Nous avons pris dans l'article l'exemple de l'étiquetage morpho-syntaxique, mais le logiciel n'est pas dépendant du type particulier d'étiquetage utilisé, et peut être utilisé dans des domaines variés de la recherche basée sur les corpus.

Références

Adda, G., Mariani, J., Paroubek, P., Rajman, M., & Lecomte, J. (1999). L'action GRACE d'évaluation de l'assignation des parties du discours pour le français. *Langues, 2*(1), 119-129.

Christ, O. (1994). A Modular and Flexible Architecture for an Integrated Corpus Query System. *Proceedings of COMPLEX'94 3rd Conference on Computational Lexicography and Text Research*, 23-32.

Miller, G. A., Beckwith, R., Fellbaum, C., Gross, D., & Miller, K. (1993). *Introduction to WordNet : An on-line lexical database* (Technical Report). Princeton University: Cognitive Science Laboratory.

Reymond, D. (2001, 2-5 juillet). *Dictionnaires distributionnels et étiquetage lexical de corpus.* Paper presented at the RECITAL (TALN) 2001, Tours.

Silberztein, M. (2000). INTEX [Linguistic software]. Champs-sur-Marne: Association pour le traitement informatique des langues (ASSTRIL).

Sperberg-McQueen, C. M., & Burnard, L. (1994). Guidelines for Electronic Text Encoding and Interchange. *Text Encoding Initiative.*

Vossen, P., Bloksma, L., Rodriguez, H., Climent, S., Calzolari, N., Roventini, A., Bertagna, F., Alonge, A., & Peters, W. (1998). *The EuroWordNet Base Concepts and Top Ontology* (LE2-4003.): EuroWordNet project.