Comment allier la puissance du TAL et la simplicité d'utilisation ? L'exemple du concordancier bilingue ConcQuest

Olivier Kraif

LIDILEM - Université Stendhal Grenoble3 - France

Abstract

This paper presents the design of a multilingual concordancer, ConcQuest, which attempts to give simple access to complex technologies of NLP. Through this presentation, we try to delineate some possible solutions, in order to cope with the inherent difficulties of formal linguistic representations. These solutions involve different aspects: simplified functionalities, standardisation, graphical aids, and the like. We show that basic NLP techniques, such as bilingual aligning, lemmatizing, pos-tagging, and regular expressions, allow one to conduct advanced searches through corpora, without being a specialist. After a brief description of the functions of ConcQuest, we describe how the query language has been designed, and compare it with more widely used systems. Then we present a help interface which illustrates how a graphical representation makes it possible to handle complex queries without mastering formal operations, and how this representation may be used as an intermediate step in the process of familiarization with the the query language.

Résumé

Cette article présente l'architecture d'un concordancier multilingue nommé ConcQuest, dont l'effort de conception s'est articulé autour d'une problématique générale en TAL : comment mettre à la disposition d'un utilisateur non spécialiste des formalismes puissants, capables de décrire des phénomènes morphosyntaxiques complexes ? A travers la présentation de ce logiciel, nous verrons qu'il existe des solutions liées à différents aspects : simplicité des fonctionnalités, standardisation, aides graphiques. Nous verrons que des techniques de base du TAL (alignement, lemmatisation et étiquetage, recherche de patterns d'expressions régulières) permettent d'accéder à des fonctionnalités avancées (critères syntaxiques, recherche de cooccurrences monolingues et bilingues) sans être spécialiste du domaine. Après avoir exposé les différentes fonctionnalités offertes par ConcQuest, nous expliquerons, à travers une étude comparative, comment le langage de requête a été conçu, sur la base de méta-expressions régulières et de relations de dépendances facultatives, dans le souci d'allier puissance et simplicité. Nous décrirons ensuite une interface graphique d'aide à la construction de ces expressions, qui poursuit deux objectifs : permettre d'utiliser le langage sans aucune connaissance du formalisme d'une part, et se familiariser à celui-ci d'autre part, grâce à une construction parallèle des expressions et de leur représentation graphique.

Mots-clés : langage de requête, concordancier, biconcordance, corpus bilingue aligné, expressions régulières.

1. Introduction

Dans l'exploration d'un corpus textuel, le recours aux techniques du TAL (traitement automatiques des langues) paraît aujourd'hui incontournable. De nombreuses bases textuelles interrogeables en ligne (p. ex. Cobuild Sampler[1], SARA[2], PICLE Corpus[3], Glossanet[4],

[1] http://www.collins.co.uk/Corpus/CorpusSearch.aspx

[2] http://sara.natcorp.ox.ac.uk/lookup.html

[3] http://ifa.amu.edu.pl/~kprzemek/PICLE_search.php

Frantext[5]) permettent de forger des requêtes complexes faisant intervenir des formes lemmatisées, des traits morphosyntaxiques, des expressions régulières, et des cooccurrences dans des contextes définis par des contraintes formelles. La plupart de ses outils offrent, comme souvent les moteurs de recherche, une seconde interface permettant d'accéder à des paramétrages avancés (sensibilité à la casse, taille de l'empan de recherche, définition des contextes, contraintes morphosyntaxiques, etc.) afin d'élaborer des requêtes complexes.

Ces outils sont certes utiles lorsque l'on veut extraire d'un grand corpus des phénomènes bien ciblés (composés terminologiques, constructions grammaticales, collocations), tout en gardant un faible niveau de bruit et de silence. Pourtant, l'expérience de l'utilisation de ses interfaces par des étudiants non spécialistes - par exemple des étudiants de lettres ou de langue, amenés à faire des recherche sur des corpus - montre que les formalismes liés à ces langages de requête sont difficiles à prendre en main. Les utilisateurs novices sont bien souvent rebutés par la complexité des critères, et préférerons se limiter à des recherches simples, plutôt que d'investir du temps dans l'apprentissage de codes complexes et sans grande généralité.

Le problème est général au TAL : comment standardiser et simplifier suffisamment les entrées / sorties et paramétrages des systèmes pour les rendre utilisables par des non-initiés, sans toutefois limiter leur puissance et leur souplesse ?

C'est en plaçant cette problématique au centre de nos réflexions que nous avons conçu le concordancier bilingue ConcQuest. Bien que les problèmes soient loin d'être résolus, nous pensons qu'il est utile de montrer comment la conception d'un tel outil peut - et doit - être guidée par un soucis d'équilibre entre puissance et simplicité d'utilisation, et d'illustrer par des choix concrets quelles peuvent être les pistes de résolution. Dans la section 2 nous présenterons brièvement les fonctionnalités de ConcQuest. Nous expliciterons ensuite quels ont été nos choix dans l'élaboration du langage de requête. Un tour d'horizon des logiciels les plus répandus dans la catégorie nous permettra de positionner plus précisément ConcQuest par rapport à l'état de l'art. La section 4, enfin, sera consacrée à la description d'une interface Web conçue pour différents types d'utilisateur, avec notamment une interface graphique de construction de requêtes complexes. Nous conclurons sur les possibilités d'évolution du langage de requête et sur l'intégration de ConcQuest à diverses applications.

2. Présentation générale

ConcQuest est une application légère[6] développée en Visual Prolog, dédiée à l'extraction de concordance pour des corpus de texte monolingues, ou bilingues alignés. Les textes traités peuvent être de différents formats : texte brut, texte étiqueté et lemmatisé au format tabulé (sorties de Treetagger) ou XML.

Les fichiers d'alignement sont des fichiers XML suivant la norme Xces[7]. Les textes XML doivent être segmentés en phrases et tokenisés, et les informations morphosyntaxiques peuvent être codées au niveau d'éléments subordonnés aux tokens, ou au niveau de leurs attributs XML. ConcQuest peut reconnaître différents types de formats (xcesAna[8], sorties de

[4] http://glossa.fltr.ucl.ac.be/indexf.html

[5] http://www.frantext.fr/categ.htm

[6] Un exécutable de 330 ko environ, actuellement compilé pour les plate-formes Win32.

[7] http://www.cs.vassar.edu/XCES/dtd/xcesAlign.dtd

[8] http://www.cs.vassar.edu/XCES/dtd/xcesAna.dtd

Xelda), et possède son propre format, nommé TXS, correspondant à une version compacte du format xcesAna, valide pour des textes désambiguïsés :

```
<?xml version="1.0" encoding="utf-8"?>
  <txs file="dickens.fr.txs" num="1">
    <chunkList>
      <chunk>
        <s id="s1">
          <tok id="t1" base="passeport" ctag="NOM" msd="Masc SG
Noun">PASSEPORT</tok>
          <tok id="t2" base="pour" ctag="NOM" msd="Prep">POUR</tok>
          <tok id="t3" base="le" ctag="DET" msd="Masc SG Def Det">LE</tok>
          <tok id="t4" base="lecteur" ctag="NOM" msd="Masc SG
Noun">LECTEUR</tok>
(…)
```

Figure 1 : Exemple de format d'entrée (TXS)

Les sorties sont de deux types :

- *concordance monolingue ou bilingue*, au format KWIC, HTML ou XML.

- *lexique bilingue*, sous format hypertextuel, présentant tous les équivalents de traductions d'une même unité trouvés dans le corpus, avec accès aux couples de phrases alignées.

2.1. Concordances et biconconcordances

Pour les concordances monolingues, la recherche peut porter sur les occurrences d'une expression ou les cooccurrences (avec ET/OU) de deux expressions (à l'intérieur des phrases).

Pour les concordances bilingues, la recherche peut porter sur les occurrences d'une expression dans une des deux langues, ou les cooccurrences (avec ET/OU) d'une expression dans chaque langue.

2.2. Critères de recherche

Les requêtes peuvent porter sur des formes, des lemmes, des traits morphosyntaxiques, ou n'importe quel attribut XML des tokens (pour un fichier au format XML).

Par exemple, si l'on cherche les verbes anglais susceptibles de traduire *poser* dans la collocation *poser un problème*, on pourra écrire la requête suivante :

```
<cat=ver> [<cat!=ver>{0,2}] %problem      AND   %poser [<>{0,2}] %problème
```

`<cat=ver>` désigne un verbe, `<cat!=ver>` désigne tout token différent d'un verbe (cette condition permet d'éviter les modaux et les auxiliaires). Le signe % permet d'indiquer un lemme. Enfin, `<>{0,2}` désigne entre 0 et 2 occurrences d'un token quelconque. Des précisions sur le langage de requête seront données dans la prochaine partie.

Le résultat, au format HTML, prendra la forme suivante :

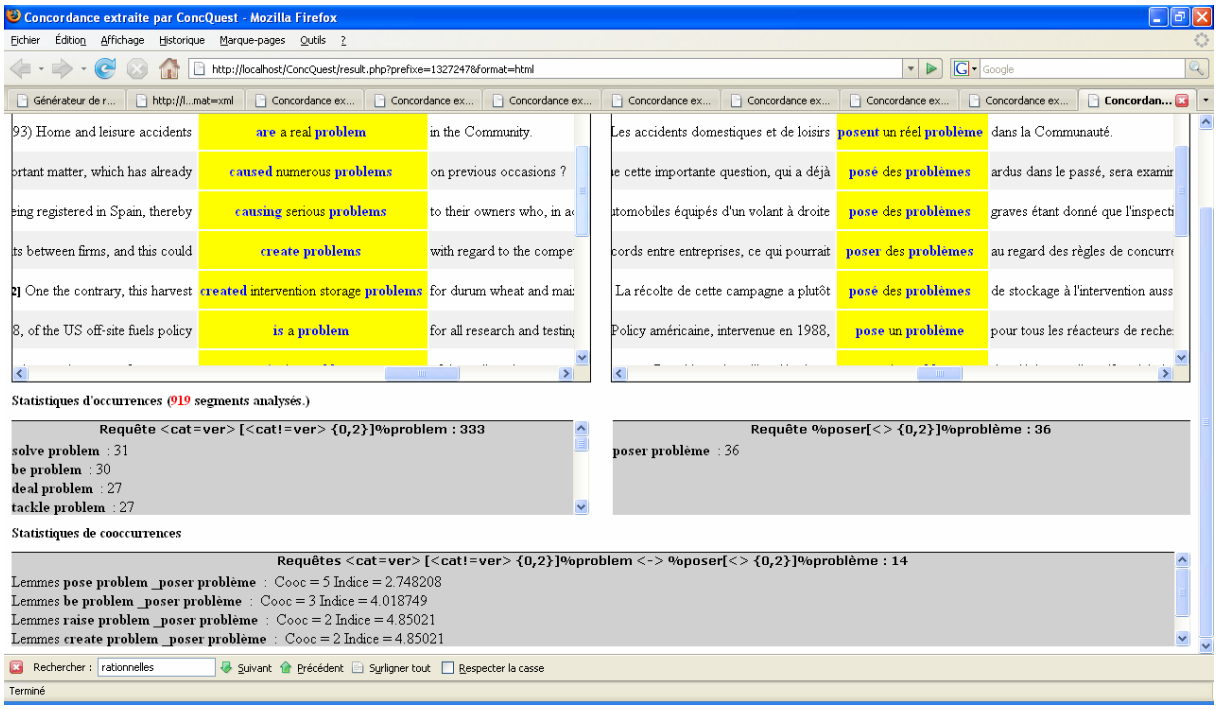

Figure 2 : Résultat d'une requête bilingue (format HTML)

2.3. Statistiques

Comme le montre la figure 2, l'extraction des concordances est accompagnée de statistiques d'occurrences concernant les requêtes prises globalement et leur différentes réalisations (comme suite de formes ou suite de lemmes, suivant le paramétrage choisi), triées par ordre de fréquences décroissantes. On obtient également des comptages de cooccurrences, accompagnés d'un indice de type information mutuelle ou *log-like* mesurant le degré d'association des expressions cooccurrentes.

2.4. Tri des sorties

Comme pour tout concordancier, les fonctionnalités de tri, au niveau des sorties, sont primordiales, afin de pouvoir regrouper les contextes analogues et effectuer des comparaisons. ConcQuest permet de définir deux clés de tri, une primaire une secondaire, sur six positions possibles : les deux expressions recherchées (s'il y en a deux) et leur contextes gauche et droit.

Dans l'exemple de la figure 2, on a trié sur l'expression recherchée en anglais, et on obtient dans l'ordre des expressions telles que :

… / caused numerous problems / causing serious problems / create problems / created intervention storage problems / is a problem / is the problem / pose any problems / …

2.5. Expressions discontinues

ConcQuest permet également de rechercher des expressions discontinues. Pour ce faire, il suffit d'encadrer les éléments intermédiaires (à ne pas sélectionner) par des crochets dans la requête. C'est ce que l'on a fait dans la figure 2 pour les tokens optionnels pris entre le verbe et le mot *problème*. Cette fonctionnalité permet de raffiner le tri (des éléments de contexte peuvent faire parti de la requête) et le calcul des statistiques d'occurrence et de cooccurrence.

Supposons que dans la collocation qui nous intéresse, nous cherchions également à intégrer la présence d'un adjectif qualificatif. La requête se complique, car en français, ce type d'adjectif connaît deux positions :

```
<cat=ver>[<cat!=ver>{0,2}]<cat=adj>%problem AND %poser[<>{0,2}](<cat=adj>
%problème|%problème<cat=adj>)
```

Dans le comptage des occurrences et des cooccurrences, les statistiques ne refléteront alors que les expressions trouvées en dehors des [] :

Occurrences en anglais (lemmatisées)	Occurrences en français (lemmatisées)
face serious problem : 6 *become serious problem : 3* *be main problem : 2* *be serious problem : 2* *cause enormous problem : 2* *cause major problem : 2* *cause serious problem : 2* *(…)*	*poser grave problème : 1* *poser nombreux problème : 1* *poser nouveau problème : 1* *poser problème ardu : 1* *poser problème grave : 1* *poser problème insoluble : 1* *poser problème juridique : 1* *(…)*

Tableau 1 : Exemples d'occurrences pour des expressions discontinues

Cooccurrences
pose major problem _poser problème majeur : 1
cause serious problem _poser problème grave : 1
cause numerous problem _poser problème ardu : 1
be real problem _poser réel problème : 1

Tableau 2 : Exemples de cooccurrences pour des expressions discontinues

De la sorte, des énoncés comme « *causing major problems* » et « *causing a major problem* » sont tous les deux comptés comme des occurrences de la forme lemmatisée *cause major problem*.

3. Conception du langage de requête

Bien qu'étant un outil de laboratoire, nous avons cherché à concevoir ConcQuest comme un outil générique susceptible de s'adresser à différents types d'usagers : étudiants, enseignants, traducteurs, linguistes, terminologues et spécialistes du TAL. Le langage de requête d'un tel outil doit satisfaire au mieux les critères suivants :

1. Permettre d'écrire des requêtes simples, sur des mots ou des expressions, sans aucune manipulation de formalisme.

2. Permettre d'accéder à des fonctionnalités avancées, en recourant à des formalismes standardisés tirant pleinement parti des fonctionnalités du TAL : étiquetage, lemmatisation, analyse syntaxique.

3. Pour les utilisateurs désireux d'aller plus loin, permettre une prise en main progressive de ces formalismes.

4. Permettre de combiner les critères sur chaque langue.

Le critère 1 est en général satisfait sur les outils grand public, comme concApp[9], AntConc[10], WordSmith[11] et nombre de concordanciers en ligne. Pour des outils de recherche plus sophistiqués, il est la plupart du temps nécessaire d'utiliser un formalisme spécialisé : requêtes en XML pour Xaira[12], contraintes sur des structures de traits pour Tiger Search (König & Lezius, 2003). La syntaxe du langage de l'IMS Workbench (Oli, 1994) quoiqu'un peu plus simple pour ses fonctions de base, impose néanmoins un usage extensif du caractère " pour encadrer les tokens. Dans cette catégorie, peu de logiciels permettent de construire des expressions simples du type "*poser problème*" : nous citerons Intex/Nooj ainsi que ConcQuest.

En ce qui concerne le critère 2, la plupart des outils grand public se contentent, au mieux d'implémenter des expressions régulières sur les chaînes (c'est le cas d'AntConc), sinon de mettre en oeuvre des caractères « génériques » (ou jokers) comme * ou ?, à l'instar de certains traitements de texte. Par exemple, chez WordSmith ou Multiconcord, on désignera par **ment* des mots se terminant par le suffixe -ment, par *trans** des mots commençant par trans- et par *trans*ment*, des mots réunissant les deux conditions. Le signe ? désigne un caractère quelconque. Ces notations ont l'avantage d'être simples, intuitives et assez répandues. Mais elles sont loin d'avoir la puissance descriptive de véritables expressions régulières : par exemple, l'expression : `aime(r(ai(s|t)?|on(s|t)|ez)|e(s|z|nt)?|ons)` permet de décrire le paradigme du verbe *aimer* au présent et au futur, ce qui n'est pas possible avec des caractères génériques. Face à un tel formalisme, un utilisateur habitué à utiliser des jokers pourra conserver ses usages, en remplaçant `*` par `.*` et `?` par `.?`. C'est pourquoi nous avons opté pour une implémentation complète des expressions régulières (syntaxe Perl).

Qui plus est, certains systèmes proposent deux niveaux d'expressions régulières : celui des caractères, et celui des tokens. Nous reprendrons le terme de « méta-expression régulière » employé par Audibert (2001). Ce type d'expression permet de définir des séquences complexes de caractères au niveau des différents attributs liés aux tokens (forme, lemme, catégorie, traits, étiquettes diverses) mais également au niveau des suites de tokens. Par exemple, dans le formalisme de l'IMS Workbench, la requête suivante permet de décrire une suite de deux noms, éventuellement séparés par la préposition *of*, sachant que la catégorie des noms correspond à différentes étiquettes commençant par 'N' :

```
[pos = "N.*"] "of"? [pos = "N.*"]
```

ConcQuest est doté d'un tel formalisme, qui présente l'avantage d'être conceptuellement simple (on reste dans le domaine des expressions régulières) et économique pour exprimer certaines contraintes. Supposons que l'on cherche à extraire des occurrences du passé composé, pour un sujet à la troisième personne du singulier, on pourra écrire :

```
<lemma=/^(être|avoir)/,cat=ver,
tags=/ipres/,tags=/pers3/,tags=/singu/><cat=adv>*<tags=ppass>
```

Plutôt que :

```
<lemma=être,cat=ver,  tags=/ipres/,tags=/pers3/,tags=/singu/>|<lemma=avoir,
cat=ver, tags=/ipres/,tags=/pers3/,tags=/singu/>)<cat=adv>*<tags=/ppass/>
```

[9] http://www.edict.com.hk/PUB/concapp/

[10] http://www.antlab.sci.waseda.ac.jp/software.html

[11] http://www.lexically.net/wordsmith/

[12] http://www.oucs.ox.ac.uk/rts/xaira/

La syntaxe de ces expressions est basée sur l'écriture des tokens entre chevrons < >, à l'intérieur desquels on liste des séries de contraintes portant sur les lemmes, la catégorie, les traits morphosyntaxiques ou n'importe quel attribut XML hérité du fichier d'entrée. La répétition des contraintes sur le même attribut `tags` est due au fait que l'on ne présume rien de l'ordre d'encodage des traits morphosyntaxiques dans cet attribut.

Enfin, nous avons également doté ConcQuest d'une prise en charge de relations de dépendances syntaxiques entre tokens, codées dans les fichiers au format TXS. Par exemple, pour trouver des occurrences du mot *recherche* et du verbe dont il est sujet, on adjoindra à la requête un critère (ou une liste de critères) de dépendances.

```
<lemma=recherche,$1>[<>*]<ctag=verb,$2>
(SUBJ,$1,$2)
```

De la sorte, il est possible de tirer parti des sorties d'un parseur, tout en privilégiant le niveau des tokens, plus simple, d'un point de vue opérationnel et logique, que celui de syntagmes imbriqués. On peut ainsi éviter de prendre en charge, à l'instar de Tiger Search, la complexité inhérente à des structures hiérarchiques correspondant aux constituants immédiats de la phrase.

Pour synthétiser ces options concernant le langage de requête, nous avons récapitulé les caractéristiques des différents systèmes précédemment cités dans le tableau 3.

Système	Expressions régulières sur les caractères.	Méta-expressions régulières	Recherche de lemme	Critères morpho-syntaxiques (POS tag)	Critères syntaxiques	Formalisme requis
AntConc	Oui	Non	Non	Non	Non	Non
WordSmith	Non joker*	Non	Non	Non	Non	Non
Intex/Nooj	Oui	Non	Oui	Oui	Non	Non
Xaira	Oui	Non	Oui	Oui	Oui	Oui
Tiger Search	Oui	Oui	Oui	Oui	Oui	Oui
ConcQuest	Oui	Oui	Oui	Oui	Oui	Non
Recherche bilingue						
ParaConc	Oui	Non	Non	Oui	Non	Non
TransSearch	Non	Non	Oui	Non	Non	Non
Multiconcord	Non. Joker*	Non	Non	Non	Non	Non
IMS workbench	Oui	Oui	Oui	Oui	Oui	Oui
ConcQuest	Oui	Oui	Oui	Oui	Oui	Non

Tableau 3 : Exemples de cooccurrences pour des expressions discontinues

Les trois systèmes les plus riches et les plus puissants, du point de vue syntaxique, sont Tiger Search, Xaira et IMS Workbench : ces deux derniers permettent notamment d'élargir les requêtes à l'ensemble des éléments de la hiérarchie XML, les contraintes pouvant s'appliquer, bien au-delà des phrases, aux paragraphes, chapitres, sous-corpus… Mais cette puissance a un prix, et les langages associés nécessitent une initiation spécifique. Par ailleurs les formalismes utilisés, bien que puissants, ne sont pas toujours économiques. L'expression de Xaira :

```
<element name="s"><or><seq><gap/><lemma>no</lemma><gap/><lemma>tea</lemma>
</gap></seq><seq><gap/><lemma>tea</lemma><gap/><lemma>no</lemma></gap></seq
></or></element>
```

pourra s'écrire avec ConcQuest :

```
ConcQuest    (%no <>* %tea | %tea <>* %no)
```

4. Interface

Pour la satisfaction du critère 3 précédemment cité, à savoir susciter la prise en main progressive des formalismes mis en œuvre, nous avons développé une interface Web, dotée d'un formulaire d'interrogation (construit très simplement à la manière des moteurs de recherche les plus courants) et d'un « assistant » graphique, permettant de construire ses requêtes de façon visuelle sans avoir à manipuler les différents opérateurs du langage ni les codes morphosyntaxiques correspondant aux étiquettes.

Pour ce faire, nous avons défini un jeu d'étiquettes unique (nommé *tagset0*), possédant une formulation explicite (ver->"verbe", ipres->"indicatif présent"), que ConcQuest traduit vers le *tagset* adapté au corpus, via des tables de conversions accessibles par l'utilisateur (les équivalences entre traits pouvant elles-mêmes être définies avec des expressions régulières).

Lors de la construction graphique de la requête, l'utilisateur peut définir, au moyen de listes déroulantes, l'ensemble des contraintes qu'il veut appliquer. L'adjonction d'un token en séquence, ou en disjonction, se fait selon les deux axes syntagmatique et paradigmatique (horizontal et vertical) pour aboutir à une représentation voisine de celle d'un automate.

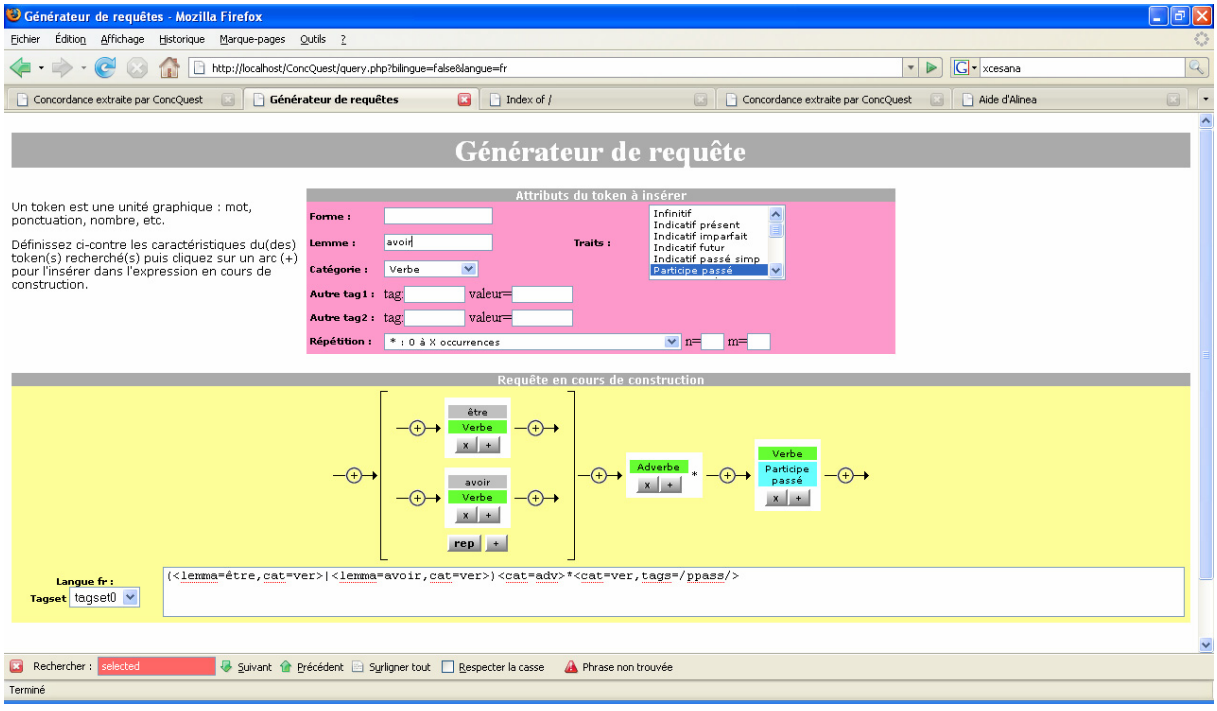

Figure 3 : Assistant pour la construction des requêtes

Parallèlement à la construction graphique, l'utilisateur voit la requête résultante s'afficher dans un champ texte, ce qui lui permet de mieux comprendre la signification de certains opérateurs, et de se familiariser avec les étiquettes du jeu de base.

Par ailleurs l'interface d'interrogation comporte des fonctionnalités originales, comme la possibilité de sauvegarder ses propres requêtes (si l'on a créé un compte) afin de se constituer une collection d'exemples réutilisables rapidement (par exemple en cours). Les paramétrages

accessibles concernent : le choix du corpus, le choix du format de sortie, le tri des sorties, l'indexation des statistiques d'occurrence par forme ou par lemme, le nombre maximum de phrases analysées, le nombre maximum de phrases trouvées, la possibilité d'effectuer un tirage aléatoire des expressions trouvées.

5. Conclusion et perspective

ConcQuest a l'ambition de proposer des fonctionnalités complètes de recherche d'expressions (expressions régulières sur les caractères et sur les traits, contraintes syntaxiques) avec une grande économie de moyen. Bien entendu, le langage des méta-expressions régulières peut encore être raffiné. Nous pensons ajouter prochainement une prise en charge de « parenthèses capturantes », comme dans Perl, qui permettrait d'enregistrer certaines chaînes pour les réutiliser dans des contraintes. On pourrait ainsi, par exemple, vérifier si deux tokens sont accordés en genre et en nombre...

ConcQuest pouvant être utilisé comme une simple fonction, en mode ligne de commande, et travailler sur des fichiers XML, en entrée et en sortie, il est facilement intégrable à d'autres applications. Un générateur d'activités autocorrectives (de type lecture d'exemples et exercices lacunaires) pour des apprenants de FLE, est en cours d'expérimentation (Kraif & Ponton, 2007). Comme nous l'avons montré par ailleurs (Kraif & Tutin, 2007), un module de recherche tel que ConcQuest peut également être utilisé dans une perspective d'aide à la rédaction.

Les potentialités d'utilisation des corpus étiquetés, et tout spécialement des corpus bilingues alignés, sont immenses, tant pour les étudiants que pour les enseignants, traducteurs ou terminologues. Pourtant, nous sommes convaincu qu'elles sont encore sous-exploitées, du fait de la complexité inhérente aux technologies du TAL qui y sont mises en œuvre. Avec un outil comme ConcQuest, relativement simple mais suffisamment puissant, nous espérons avoir fait un pas, aussi modeste soit-il, dans la direction des utilisateurs, afin de les rapprocher de la technologie.

Références

Audibert L. (2001). LoX : outil polyvalent pour l'exploration de corspus annoté, *Actes de TALN/RECITAL*, Tours, 2-5 juillet 2001, Atala. (cf. http://www.up.univ-mrs.fr/delic/papiers/Audibert-2001recital.pdf).

Christ O. (1994). A modular and flexible architecture for an integrated corpus query system. *COMPLEX'94, Budapest.*

König E. and Lezius W. (2003). *The TIGER language - A Description Language for Syntax Graphs, Formal Definition. Technical report IMS*, Universität Stuttgart, Germany.

Kraif O. et Ponton C. (2007). Du bruit, du silence et des ambiguïtés : que faire du TAL pour l'apprentissage des langues ?*, Actes de TALN 2007*, Toulouse, 12-15 juin 2007.

Kraif O. and Tutin A. (2007). Looking for Semi-Frozen Expressions using an Aligned Corpus : an Application for Academic Writing for EFL Learners, *7e Conférence Teaching and Language Corpora : TaLC2006*, Université Paris 7, Paris.

Logiciels

ConcQuest est distribué gratuitement par son auteur. La version actuelle fonctionne sous Win32. Une version Linux sera prochainement disponible. Plus d'informations à l'adresse : http://w3.u-grenoble3.fr/kraif