Grammaires en mode avancé dans ScienQuest 1.9

La syntaxe des grammaires a été développée par Achille Falaise, à partir de la syntaxe de ConcQuest développée par Olivier Kraif.

Cette syntaxe est sensiblement différente de celle qui était utilisée avant ScienQuest 1.8.

1 Le mode recherche avancée

Il est possible de passer du mode recherche libre au mode avancé (traduction du mode recherche libre en mode avancé – cf. figures 1 et 2). C'est aussi possible en partant du mode sémantique, et cela peut permettre d'étudier (et de modifier) les requêtes précodées du mode sémantique. L'inverse (passer du mode avancé vers l'un des deux autres) n'est pas possible.

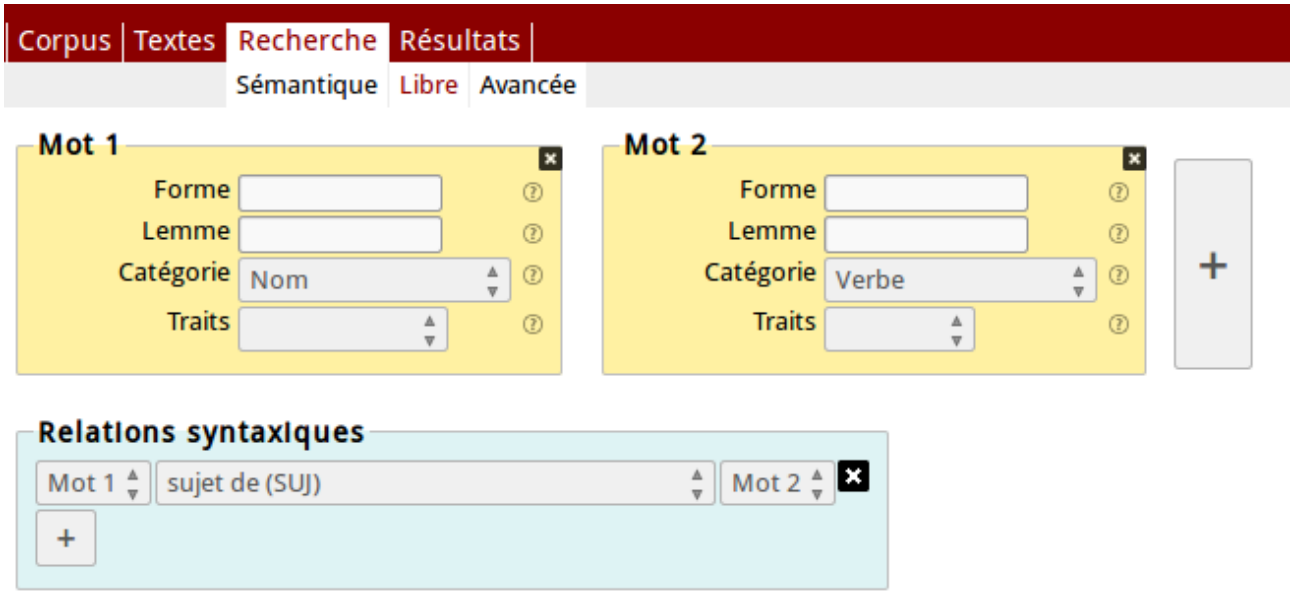

Figure 1: Recherche en mode libre...

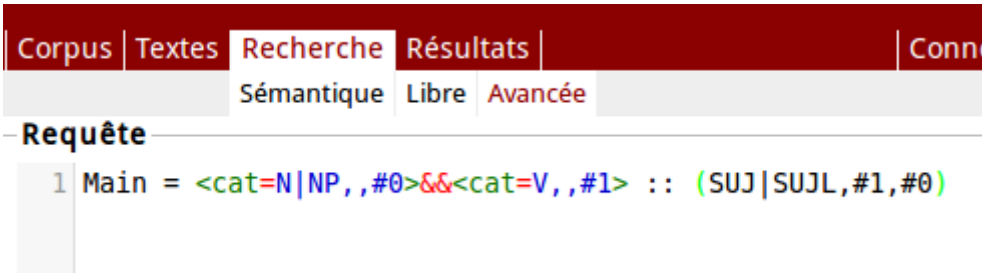

Figure 2: ... et recherche équivalente en mode avancé.

Pour observer les relations utilisées et les catégories, l'utilisateur s'aider des arbres syntaxiques dessinés par ScienQuest (après une recherche, dans la liste des concordances, cliquer sur une concordance, puis cocher « Syntaxe – cf. figure 3).

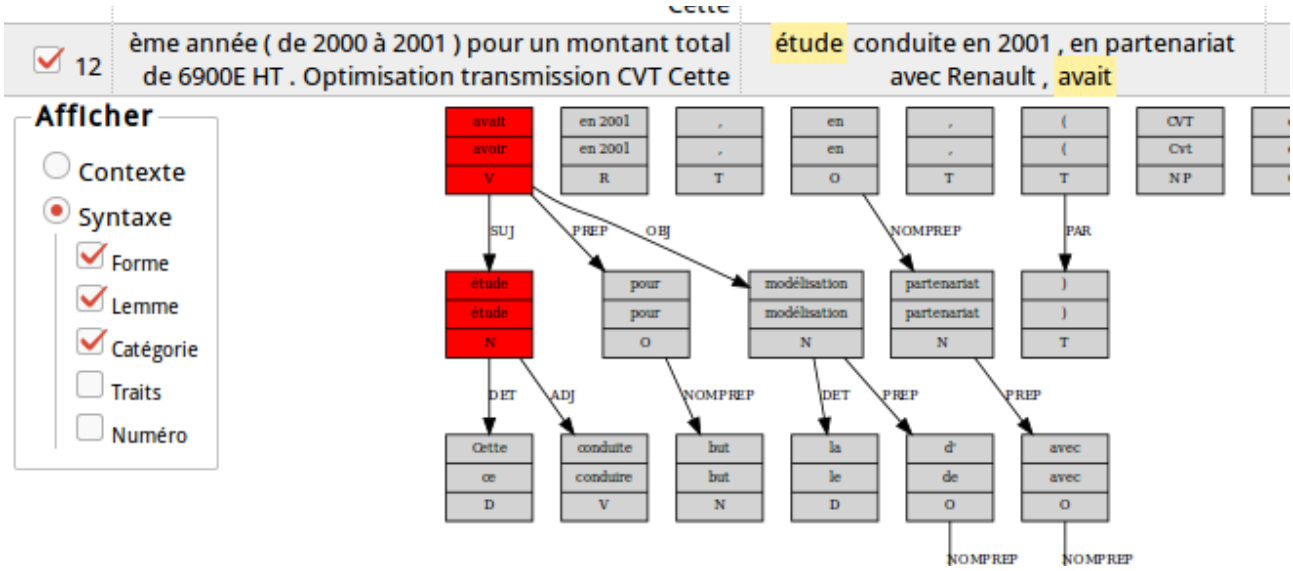

Figure 3: Arbre syntaxique d'une concordance.

2 Recherches sur des mots à la suite

Une grammaire est constituée d'expressions. Chaque expression se termine par un retour à la ligne. Une grammaire doit contenir au moins une expression de type « *Main* ». Une expression *Main* contient une liste de mots.

```
Main = <cat=V> <cat=DET> <cat=N>
```

Exemple 1: Exemple de requête ScienQuest, constituée d'une seule expression de type Main, *qui va rechercher tous les verbes suivis d'un déterminant suivis d'un nom.*

Il est possible d'avoir plusieurs expressions de type *Main*. Dans ce cas, on recherche toutes les concordances qui correspondent à au moins une des expressions.

```
Main = <cat=V> <cat=DET> <cat=N>
Main = <cat=V> <cat=N>
```

Exemple 2: Exemple de requête ScienQuest, constituée de deux expressions de type Main, *qui va rechercher tous les verbes, suivis d'un déterminant suivis d'un nom, OU directement suivis d'un nom (sans déterminant).*

Le langage utilisé est essentiellement celui de ConcQuest, développé par Olivier Kraif.

2.1 Contraintes simples sur un mot

Il est possible de préciser plusieurs types de critères pour chaque mot :

- Rechercher uniquement une forme donnée :

```
Main = <form=chanta>
```

- Rechercher uniquement un lemme donné :

```
Main = <lemma=chanter>
```

Attention. Certains analyseurs génèrent des « mots » polylexicaux. Par exemple, pour l'analyseur Syntex, « *c'est à dire* » est un seul mot. Par conséquent, l'exemple ci-dessous fonctionnera avec cet analyseur :

```
Main = <lemma=c'est à dire>
```

Mais pas le suivant :

```
Main = <lemma=cela><lemma=être><lemma=à><lemma=dire>
```

- Rechercher uniquement une catégorie syntaxique (partie du discours) :

```
Main = <cat=V>
```

- Rechercher uniquement un trait morpho-syntaxique :

```
Main = <tags=VINF>
```

Attention. La liste des catégories syntaxiques, des traits morpho-syntaxiques, et de leurs codes, dépendent de l'analyseur syntaxique utilisé pour traiter le corpus. **Les exemples ci-dessus ne fonctionneront donc que sur certains corpus**. Si on ne connaît pas bien l'analyseur utilisé, le plus simple est de faire d'abord des recherches en mode libre, puis d'observer l'analyse syntaxique de quelques phrases (cf. figure 3).

Il est possible d'utiliser des négations en utilisant l'opérateur != à la place de = :

- Recherche tous les lemmes sauf « *être* » :

```
Main = <lemma!=être>
```

2.2 Contraintes multiples sur un mot

Il est possible de combiner plusieurs contraintes sur un mot. Par exemple, pour rechercher les occurrences de la forme « est » où il s'agit d'un nom (et non du verbe être) :

```
Main = <form=est,cat=N>
```

N.B. L'exemple ci-dessus est correct mais peu artificiel. L'exemple ci-dessous retournera le même résultat, mais est plus simple (puisqu'en français le lemme *est* n'est pas ambigu) :

```
Main = <lemma=est>
```

2.3 Listes

Pour chaque critère, il est possible d'utiliser une liste (le caractère | est utilisé comme séparateur) :

```
Main = <lemma=être|paraître|sembler,cat=V>
```

Contrairement aux autres critères, il n'est pas possible d'utiliser une négation en remplaçant l'opérateur = par l'opérateur != . Toutefois on peut utiliser une expression régulière à la place (voir ci-après).

2.4 Expressions régulières

Pour tous les critères, il est possible d'utiliser des expressions régulières. Une expression régulière est notée entre caractères / . Par exemple, tous les mots dont la forme contient « *expliq* » (*expliquer*, *expliquait*, *réexpliquer*, etc.) :

```
Main = <lemma=/expliq/>
```

Rechercher tous les verbes sauf être et avoir :

```
Main = <lemma!=/^(être|avoir)$/>
```

2.5 N’importe quel mot

L’expression <> désigne n’importe quel mot. Par exemple, pour rechercher n’importe quel mot suivant le verbe *faire* :

```
Main = <lemma=faire> <>
```

3 Recherches sur des mots éloignés

Par défaut, les recherches s'effectuent toutes sur des mots se suivant. Il est possible d'indiquer que deux mots ne se suivent pas obligatoirement (ils peuvent être éloignés l'un de l'autre dans la phrase, et dans n'importe quel ordre), en les séparant par *&&* :

```
Main = <lemma=hypothèse> && <cat=V>
```

Dans cet exemple, on cherche toutes les occurrences d'*hypothèse* et d'un verbe dans une phrase, peut importe leur position à l'intérieur de celle-ci.

Il est en outre possible de définir un intervalle facultatif entre deux mots, grâce à l’expression <>{*intervalle*}, où *intervalle* est la taille maximum de cet intervalle. Par exemple, pour rechercher *hypothèse* suivi d’un verbe, et séparés par deux mots au maximum (donc séparés par 0, 1 ou 2 mots) :

```
Main = <lemma=hypothèse> <>{2} <cat=V>
```

L’expression définit un intervalle <>* de taille indéterminée (entre zéro et l’infini). Par exemple, pour rechercher *hypothèse* suivi d’un verbe, et séparés par n’importe quel nombre de mots :

```
Main = <lemma=hypothèse> <>* <cat=V>
```

Attention, les requêtes comportant des intervalles peuvent retourner un nombre colossal de résultats, et donc prendre longtemps à s’exécuter.

4 Relations syntaxiques

Le mode avancé permet d’effectuer des requêtes sur toutes les relations syntaxiques du corpus, alors que l’assistant du mode libre ne donne accès qu’aux relations les plus utiles.

Il est possible d'indiquer une relation syntaxique entre deux mots. Les relations syntaxiques sont regroupées en fin d'expression. Par exemple, *hypothèse*, suivi d'un verbe, et sujet de ce verbe :

```
Main = <lemma=hypothèse,#1> <cat=V,#2> :: (SUJ,#2,#1)
```

```
Main = <cat=D,#0> <lemma=hypothèse,#1> <cat=V,#2> :: (SUJ,#2,#1) (DET,#1,#0)
```

Il est généralement plus intéressant de définir des relations syntaxiques sur des mots ne se suivant pas nécessairement :

```
Main = <lemma=hypothèse,#1> && <cat=V,#2> :: (SUJ,#2,#1)
```

```
Main = <cat=D,#0> && <lemma=hypothèse,#1> && <cat=V,#2> :: (SUJ,#2,#1) (DET,#1,#0)
```

Il est possible d'avoir des listes de relations syntaxiques (ne fonctionne pas avec les relations virtuelles) :

```
Main = <lemma=hypothèse,#1> && <cat=V,#2> :: (SUJ|SUJL|OBJ,#2,#1)
```

Il est aussi possible de sous-spécifier une relation syntaxique pour chercher *n'importe quelle* relation :

```
Main = <lemma=hypothèse,#1> && <cat=V,#2> :: (,#2,#1)
```

5 Variables

Par commodité, il est possible d'utiliser des variables pour définir des listes de formes et de lemmes (le caractère , est utilisé comme séparateur). Le nom d'une variable commence toujours par le caractère $:

```
$formul=avancer,émettre,effectuer,faire,formuler,poser,prendre,proposer

$hypo=hypothèse,conjecture

Main = <lemma=$formul,#1> && <lemma=$hypo,#2> :: (OBJ,#1,#2)

Main = <lemma=$formul,#1> && <lemma=$hypo,#2> :: (SUJ,#1,#2)
```

6 Relations virtuelles

Par commodité, il est possible de définir une relation « virtuelle » qui combine des relations normales. Par exemple :

```
(OBJPASSIF,#1,#2) = (SUJ,#3,#2)(AUX,#3,#1)

$formul=avancer,émettre,effectuer,faire,formuler,poser,prendre,proposer

$hypo=hypothèse,conjecture

Main = <lemma=$formul,#1> && <lemma=$hypo,#2> :: (OBJ,#1,#2)

Main = <lemma=$formul,#1> && <lemma=$hypo,#2> :: (SUJ,#1,#2)

Main = <lemma=$formul,#1> && <lemma=$hypo,#2> :: (OBJPASSIF,#1,#2)
```