Référence rapide CQL pour la BFM

Table des révisions

Date	Nom	Modification
20/07/11	Bénédicte Pincemin (BP)	Création du document à partir de la section 7 du tutoriel TXM pour la BFM.
27/03/12	Alexei Lavrentiev (AL) et Céline Guillot (CG)	Création d'une version de référence rapide
11/05/12	AL	Suppression des exemples de l'opérateur %d actuellement indisponible
15/05/19	AL et CG	Mise à jour pour le corpus BFM2019

Table des matières

1 Introduction

1.1 CQP, CQL

La Base de français médiéval est accessible en ligne grâce au logiciel « portail TXM », elle peut aussi être téléchargée et interrogée localement avec le logiciel TXM pour poste. Dans les deux cas, il est possible d'utuliser des requête CQL.

CQL est l'acronyme de *Corpus Query Language*, c'est un langage d'expression de requêtes. Une expression CQL est une chaîne de caractères exprimant un motif linguistique (un mot, ou une suite de mots) à partir des valeurs de leurs propriétés (comme la catégorie grammaticale, le lemme, la forme graphique).

CQP est l'acronyme de *Corpus Query Processor*, c'est un module logiciel qui traite des requêtes : c'est un moteur de recherche qui permet de trouver toutes les occurrences correspondant à une équation CQL dans un corpus donné. Le moteur CQP a été développé à l'université de Stuttgart. Il est intégré à TXM où il assure les recherches d'occurrences et d'une façon générale toutes les opérations de sélection à l'intérieur du corpus.

1.2 Les requêtes dans TXM : requêtes simples, requêtes avancées

CQL est donc un langage formel, avec un lexique et une syntaxe d'opérateurs, qui forment un métalangage permettant de combiner des éléments pour la recherche de motifs structurés.

L'apprentissage du langage CQL n'est pas un passage obligé pour utiliser TXM (et donc la BFM), mais c'est en langage CQL qu'on a le mode d'expression de motifs le plus riche.

Si l'on saisit un mot dans la zone de requête, c'est interprété comme la recherche des mots présentant exactement cette graphie dans le corpus. Cela permet déjà un certain nombre de recherches simples. Mais on perçoit assez vite deux limites : d'une part, on reste à la « surface » du texte, on ne tire aucun parti des autres informations linguistiques encodées dans le corpus (lemme, catégorie grammaticale, etc). D'autre part, on est rivé à l'empan exact d'un mot : la formulation de la recherche ne peut se faire ni sur une partie du mot (son début par exemple), ni sur des expressions en plusieurs mots – alors que cela devient possible en utilisant CQL.

1.3 Dynamique de la construction d'une requête

Une requête se met au point : entre ce qu'on veut repérer (que l'on pense avoir exprimé dans la requête), et ce qu'on trouve effectivement dans le corpus, il y a souvent un écart qui demande à être corrigé. Il est de toutes façons toujours sage de vérifier la portée effective, dans le corpus choisi, de la requête utilisée, avant de l'utiliser pour un calcul statistique.

L'apprentissage et l'utilisation de CQL font donc un usage central de la fonctionnalité Index de TXM. La fonctionnalité Index permet de lister toutes les formes correspondant au motif dans le corpus. On peut les parcourir soit par importance quantitative décroissante (tri par fréquence décroissante, qui est la manière dont se présente le résultat par défaut), soit par ordre alphabétique, ce qui peut faciliter la lecture en regroupant les réalisations de forme proche.

Le parcours de cette liste des configurations trouvées met en évidence les formes indésirables ; en revanche il ne dit rien des formes qui seraient pertinentes mais qui, ne correspondant pas formellement à la requête, n'ont pas été repérées. Méthodiquement, on recommande donc toujours, quand on a un motif linguistique à recherche, de commencer par l'exprimer de façon très ouverte, de veiller à minimiser les *a priori* qui pourraient être réducteurs. L'examen des occurrences

correspondantes trouvées guide alors sur la manière d'ajouter alors peu à peu des contraintes permettant de cibler les « bonnes » formes et d'écarter les « mauvaises ».

2 Exemples de recherches

2.1 Les niveaux d'occurrences

Une requête CQL peut s'exprimer sur deux niveaux différents. Le premier niveau concerne les caractères constituant l'expression de recherche. Le second niveau concerne les successions d'occurrences.

Exemples :

```
"cist"%c
```

Cette requête cherche toutes les formes de « cist », avec ou sans majuscule. Il s'agit du premier niveau de l'occurrence, qui porte sur la forme graphique de l'occurrence recherchée.

```
"cist"%c "chevalier"
```

Cette requête porte sur une succession de deux occurrences. Nous nous plaçons alors au second niveau.

Les exemples présentés dans ce tutoriel seront répartis en fonction de ces deux niveaux.

2.2 Premier niveau : recherche un seul ensemble de formes

2.2.1 Mots

- Recherche littérale sur une seule forme graphique :

```
fortune
```

La recherche porte sur la forme graphique du mot ci-dessus.

Remarque 1 : la définition du mot dépend de l'indexation qui a été faite du corpus, il faut savoir comment ont été définies les unités lexicales (critères linguistiques, typographiques, etc.), ou a minima savoir qu'une recherche infructueuse peut être liée à une méconnaissance des limites de mot (relancer alors la recherche de façon assouplie, en utilisant la troncature, cf. plus loin). Dans la BFM, l'indexation a détaché les articles élidés et la requête suivante ne fournit aucun résultat :

```
l'ami
```

En revanche on peut chercher :

```
l'
```

Recherche des mots qui ont comme forme graphique « l' ».

```
ami
```

Recherche des mots qui ont comme forme graphique « ami ».

Pour savoir comment rechercher la succession des deux (*l'ami*), se reporter à la section 2.3

Remarque 2 : Les marques de ponctuation sont traitées de la même façon que les mots, elles sont donc intégrées dans les indexes et calculs de dimensions des textes.

Remarque 3 : ce qu'on recherche ainsi est la graphie complète du mot, et non tout mot contenant la chaîne de caractères donnée. Si l'on veut chercher des mots qui contiennent une certaine chaîne de caractères, c'est possible en indiquant qu'il y a une troncature (cf. plus loin).

```
fortun
```

Recherche des mots qui ont comme forme graphique « fortun » (et non pas des mots qui contiennent la chaîne de caractères « fortun »). Dans notre corpus nous n'en trouvons aucun, cette chaîne n'existe pas en tant que mot complet.

Remarque 4 : la saisie directe de la graphie du mot est une forme de requête simplifiée. Les trois requêtes suivantes ont la même signification, la troisième étant la plus explicite mais aussi la plus complexe à saisir.

```
fortune
```

```
"fortune"
```

```
[word="fortune"]
```

Recherche des mots qui ont comme forme graphique « fortune ».

Nous verrons par la suite des cas où seule la forme la plus complète convient. Il est utile de s'habituer à la lire et à l'écrire, c'est pourquoi nous l'avons indiquée systématiquement dans les exemples qui suivent.

Attention, il ne faut pas mettre d'espace entre les guillemets et la chaîne de caractères cherchée, **à l'intérieur des guillemets tous les espaces « comptent »** (à moins que l'on ne veuille justement rechercher une forme contenant un espace, par exemple un mot composé qui aurait été identifié comme tel au moment de l'import du corpus).

- Recherche non sensible à la casse :

```
"fortune"%c
```

La recherche porte sur la forme graphique du mot placé entre guillemets, en ignorant la casse « %c ».

- Recherche d'un mot commençant par « for » :

```
"for.*"
```

*Le « . » signifie « n'importe quel caractère », le « * » signifie « facultatif ou répété n fois ».*

– Recherche d'un mot finissant par « une » :

```
".*une"
```

– Recherche d'un mot commençant par « f » et finissant par « e » :

```
"f.*e"
```

– Recherche d'un mot contenant un « u » :

```
".*u.*"
```

– Recherche du mot « fortune » au singulier et au pluriel :

```
"fortunes?"
```

Le « ? » signifie que le caractère qui le précède n'est pas obligatoire.

– Recherche du mot « fortune », au singulier et au pluriel, ignorant la casse :

```
"fortunes?"%c
```

L'opérateur « %c » se situe en dehors des guillemets, ce qui signifie qu'il porte sur toute la chaîne de caractères.

– Recherche du mot « hasard » dans toutes ses graphies :

```
"h?a[sz]ar[dt]s?"%c
```

Cette expression signifie qu'on cherche un mot qui commence peut-être par « h », suivi de « a », suivi de « s » ou « z », suivi de « ar », suivi de « d » ou « t », peut-être suivi de « s », en ignorant la casse.

– Recherche portant sur la racine lexicale « hasard » et ses dérivés :

```
"h?a[sz]ar[dt].*"%c
```

Cette expression signifie qu'on cherche un mot qui commence peut-être par « h », suivi de « a », suivi de « s » ou « z », suivi de « ard » ou de « art », et de n'importe quelle terminaison, en ignorant

la casse.

- Recherche des noms en « isme » :

```
".+ismes?"
```

« . » signifie « n'importe quel caractère », « + » signifie « répété n fois », suivi de « isme », suivi peut-être de « s ».

- Recherche combinée du mot « fortune » ou « hasard » :

```
"(fortunes?|h?a[sz]r[dt]s?)"%c
```

Cette expression signifie que l'on cherche le mot « fortune » au singulier et peut-être au pluriel, ou bien un mot qui commence peut-être par « h », suivi de « a », suivi de « s » ou « z », suivi de « r », suivi de « d » ou « t », peut-être suivi de « s », en ignorant la casse (ce qui équivaut à toutes les graphies de « hasard » au singulier ou au pluriel).

- Extension du pivot

```
"fortune" expand to 3
```

« expand to 3 » permet d'inclure dans le pivot 3 formes qui précèdent et 3 occurrences qui suivent « fortune ». On peut ainsi afficher un index des contextes de ce mot. « expand to s » étend le pivot à la phrase entière.

- Recherche négative :

```
[word!="(qu|gr)an[tsz]"%c]
```

« ! » après le mot « word » signifie qu'on exclut tous les mots qui commencent par « qu » ou « gr » suivi de « an », suivi de « t », « s », ou « z », la recherche ignorant la casse.

2.2.2 Ponctuations

Comme nous l'avons déjà dit les ponctuations sont traitées de la même manière que les mots. Mais puisque certains signes de ponctuation sont aussi des opérateurs de recherche, il faut distinguer deux cas de figure :

- Lorsque le caractère concerné ne fait pas partie des opérateurs (la virgule, le point-virgule, les deux-points, le tiret, l'apostrophe, le point d'exclamation, les chevrons,...)

Le caractère de ponctuation est traité comme les lettres des mots.

```
[word="!"]
```

Recherche des points d'exclamation.

```
[word=".*'"]
```

Recherche des formes élidées.

- Lorsque le caractère concerné fait partie des opérateurs (le point, le point d'interrogation, le guillemet droit, les crochets, les parenthèses, l'astérisque, la barre oblique descendante...)

Il faut placer une barre oblique inversée avant, pour indiquer que le caractère n'a pas son sens d'opérateur, mais est lui-même un élément de la recherche.

```
[word="\?"]
```

Recherche des points d'interrogation.

```
[word="\""]
```

Recherche des guillemets.

2.3 Second niveau : recherche sur une succession de 2 ou n ensembles de formes

- Recherche de deux cooccurrents :

```
"bonne" "fortune"
```

Recherche le mot « bonne » suivi de « fortune ». L'espace entre les deux formes est indispensable.

- Recherche de la cooccurrence « bonne fortune », au singulier et au pluriel, en ignorant la casse :

```
"bonnes?"%c "fortunes?"%c
```

Recherche le mot « bonne », au singulier et au pluriel, en ignorant la casse, suivi du mot « fortune », au singulier et au pluriel, en ignorant la casse.

- Recherche de l'expression « Courir (la) fortune » :

```
"cou?r.+"%c []? "fortune"
```

Recherche un mot commençant par « co », peut-être suivi de « u », suivi de « r », suivi de n'importe quelle terminaison, en ignorant la casse. Ce mot sera peut-être suivi d'un autre mot de n'importe quelle forme« []? ». Ce mot sera suivi de « fortune ».

- Recherche de la cooccurrence « fortune + tourner », dans un contexte de 10 mots :

```
"fortune"%c []* "tou?rn.*" within 10
```

Recherche du mot « fortune » (en ignorant la casse), éventuellement suivi d'un autre mot, suivi d'un mot commençant par « to », peut-être suivi de « u », suivi de « rn », suivi de n'importe quelle terminaison, dans un contexte de 10 mots (« within »).

- Rechercher les participes présents, en excluant une série de formes qui se finissent en -ant mais qui ne sont pas des participes présents (maintenant, etc.) :

```
[word=".+an[tsz]" & word!="(qu|gr|dev|mainten|av|a?u?t|s|enf|neporqu)an[tsz]"%c]
```

Recherche d'un mot finissant par « an », suivi de « t », ou « s », ou « z », en excluant les mots qui commencent par « qu », ou « gr », ou « dev » ou « mainten » ou « av » ou « aut », « at »n « ut », voire « t », ou « s » ou « enf » ou « neporqu », suivi de « an », suivi de « t », « s », ou « z », la recherche ignorant la casse.

Le signe « & » exprime une combinaison de contraintes, que doit suivre l'occurrence.

2.4 Recherche sur les étiquettes : recherche sur les propriétés de mots

Jusqu'alors, les recherches effectuées portaient sur la forme graphique des mots, qui est une propriété par défaut : `[word="fortune"]` signifie qu'on recherche la valeur « fortune » de la propriété graphique (word). Cette requête équivaut à "fortune", car par défaut les requêtes portent sur la forme graphique de l'occurrence. Mais les requêtes peuvent aussi porter sur d'autres propriétés des mots (et les combiner). Les propriétés disponibles dans les corpus français de la BFM sont les suivantes :

- `word` (forme graphique, propriété par défaut)
- `cattex-pos` (étiquette morphologique au format Cattex 09)
- `cattex-pos_syn` (étiquette morphosyntaxique au format Cattex 09, voir les Principes d'annotation pour connaître la différence entre l'étiquetage morphologique et morphosyntaxique)
- `etiquetage` (état de vérification de l'étiquetage)
- `id` (identifiant de l'occurrence)
- `lang` (langue de l'occurrence)
- `lemma` (lemme, voir la section 7.5 pour en savoir plus sur la lemmatisation de la BFM)
- `lemma_src` (source du lemme : DMF, TL, AND, BFM ou autre)
- `lemmatisation` (état de vérification de la lemmatisation)
- `q` (niveau d'imbrication dans discours direct : « 0 » si hors discours direct, « 1 » si l'occurrence se trouve dans un discours direct de niveau 1, … « 3 » si l'occurrence se trouve dans un discours direct imbriqué dans deux autres discours directs)
- `ref` (titre du texte et page où se trouve l'occurrence)
- `sp` (« t » si l'occurrence se trouve dans une prise de parole dans un texte dramatique ou dans un dialogue, sinon « f »)
- `supplied` (« t » si l'occurrence se trouve dans une conjecture de l'éditeur scientifique,

sinon « f »)

- `ud-pos` (étiquette de partie de discours du jeu Universal dependencies)
- `ud-feats` (étiquette qui regroupe les traits morphologiques du jeu `Universal dependencies`)

Noter que certaines propriétés sont ouvertes, c'est-à-dire que l'ensemble des valeurs qu'elles peuvent prendre n'est pas déterminé a priori (ici, c'est le cas de *word* et de *lemma*), alors que d'autres sont fermées, c'est-à-dire que les valeurs qu'elles peuvent prendre sont choisies dans une liste de catégories prédéfinies (ex. jeu d'étiquettes CATTEX ou UD, état de l'étiquetage morphosyntaxique ou de lemmatisation, etc.). Les propriétés *lemma* et *ud-feats* sont particulières, car elles peuvent contenir plusieurs informations alternatives (différents lemmes possibles pour les textes non vérifiés ou différents traits morphologiques). Les section 2.5 et 2.6 ci-dessous sont consacrées à leur interrogation.

Voici quelques exemples de requêtes portant sur les propriétés de mots simples :

```
[cattex-pos="V.*"]
```

Cherche toutes les formes de verbes. « V » est la première lettre de l'étiquette concernant les verbes (étiquette vérifiée ou non).

```
[cattex-pos="VERcjg"]
```

Cherche toutes les formes de verbes conjugués (étiquette vérifiée ou non).

```
"graal"%c [cattex-pos="PRO.*" & etiquetage="vérifié"]
```[1]

Cherche toutes les occurrences du mot « graal », en ignorant la casse, suivi d'un pronom (étiquette vérifiée).

```
"chevalier"%c [ ]? [cattex-pos="V.*"]
```

Cherche le mot « chevalier », en ignorant la casse, peut-être suivi d'un mot, suivi d'un verbe (étiquette vérifiée ou non).

```
[word="ne"%c & cattex-pos="CONcoo"]
```

Cherche toutes les occurrences de « ne » (avec ou sans majuscule) qui sont des conjonctions de coordination (étiquette vérifiée ou non).

```
[cattex-pos="PROper" & etiquetage="vérifié" & q="[123]"]
```

Cherche toutes les occurrences des pronoms personnels (étiquette vérifiée) dans le discours direct à tout niveau d'enchâssement (cf. l'explication plus haut).

```
[cattex-pos!="PON.*"]
```

1 Une autre manière de restreindre la requête aux seules étiquettes vérifiées est de travailler sur un sous-corpus de textes et d'extraits vérifiés. Dans la BFM2019, ce sous-corpus est disponible à tous les utilisateurs.

Cherche toutes les occurrences du corpus à l'exception des ponctuations (autrement dit, tous les mots).

```
[cattex-pos="(DETdem|DETdef)"] [word="parole"%c]
```

Cherche les occurrences d'un déterminant démonstratif ou défini (étiquette vérifiée ou non) suivi du mot « parole ».

```
[word=".+oir" & word!="p.*"%c][ ]*[cattex-pos="VER.*"]  within 10
```

Cherche un mot qui finit par « oir », mais qui ne commence pas par la lettre « p », en ignorant la casse, peut-être suivi d'un ou plusieurs mots, suivi d'un verbe (étiquette vérifiée ou non), dans un contexte limité à 10 mots.

La propriété *cattex-pos_syn* permet d'interroger l'étiquette morphosyntaxique d'une occurrence, lorsqu'elle est différente de l'étiquette morphologique *cattex-pos*. C'est notamment le cas des infinitifs substantivés :

```
[cattex-pos="VERinf" & cattex-pos_syn="NOMcom"]
```

La distinction des étiquettes morphologiques et morphosyntaxiques est uniquement disponible dans certains textes qui bénéficient de l'étiquetage vérifié, les résultats ainsi obtenus ne sont donc en aucun cas exhaustifs.

2.5 Étiquettes morphosyntaxiques Universal dependencies (UD)

Cf. la section 7.5.2 du Tutoriel BFM

Dans les corpus BFM2019, PALAFRAFRO-V2-2 et PALAFRAPAR, ainsi que dans le corpus latin PALAFRALAT-V2-0, il est possible d'utiliser les étiquettes du jeu universel défini par le projet *Universal dependencies*. Pour les textes français ces étiquettes ont été obtenues par conversion automatique à partir des étiquettes morphologiques Cattex 09. Des propriétés différentes sont utilisées pour indiquer la partie du discours et les différents traits morphologiques :

- **`ud-pos` (partie du discours)**
- `ud-feats` (ensemble de traits morphologiques)

Le format de la propriété ud-feats a été optimisé pour faciliter les requêtes sur un trait particulier d'un ensemble complexe. Pour ce faire, il convient d'utiliser l'opérateur *contains* dans la requête :

```
[ud-feats contains "PronType=Art"]
```

Cherche un mot dont la propriété ud-feats contient une valeur PronType=Art (toute sorte d'article).

Voici un exemple de requête permettant de combiner deux propriétés distinctes et deux valeurs au sein d'une de ces propriétés :

```
[ud-pos="VERB" & ud-feats contains "VerbForm=Part\|Tense=Pres"]
```

Cherche un mot dont la propriété ud-pos est égale à « VERB » (verbe) et la propriété ud-feats contient une combinaison de deux étiquettes : VerbForm=Part (forme verbale participe) et Tense=Past (temps passé).

Les différentes valeurs sont séparées par des barres verticales, et les barres verticales entourent également l'ensemble des valeurs. On peut donc obtenir un résultat équivalent avec la requête suivante :

```
[ud-pos="VERB" & ud-feats="\|VerbForm=Part\|Tense=Pres\|"]
```

Les barres obliques inversées permettent d'indiquer qu'on cherche le caractère « | » et qu'il ne s'agit pas d'un opérateur d'expression régulière (cf. le tableau dans la section 5).

La documentation complète du jeu UD est consultable sur le site http://universaldependencies.org.

Le tableau de conversion des étiquettes Cattex09 → UD est donné dans l'Annexe 3 du *Tutoriel de la BFM* et dans la rubrique « Documentation » du site http://bfm.ens-lyon.fr.

2.6 Lemmes

Cf. la section 7.5.3 du Tutoriel BFM.

La lemmatisation est actuellement disponible uniquement dans les corpus BFM2019, PALAFRAFRO-V2-2 et PALAFRAPAR. Elle est vérifiée dans un nombre restreint de textes et automatique dans le reste du corpus. Le sous-corpus 'lemmatisé' de la BFM2019 donne accès uniquement aux textes dont la lemmatisation et l'étiquetage ont été vérifiés.

En cas de lemmatisation automatique, plusieurs hypothèses peuvent être proposées. Le format des lemmes a été optimisé pour faciliter la recherche parmi les hypothèses multiples. Ainsi, pour rechercher toutes les formes du verbe *être* vous pouvez utiliser la requête suivante :

```
[lemma contains "être"]
```

La majorité des lemmes sont issus du *Dictionnaire du moyen français* (DMF), mais dans certains cas les lemmes proviennent du dictionnaire Tobler-Lommatzsch ou d'autres sources. Il est donc utile d'inclure les formes des lemmes de différentes sources :

```
[lemma contains "être|estre"]
```

Notez que les chiffres utilisés dans le DMF pour distinguer les homonymes ne sont pas inclus dans les lemmes. Une requête combinée sur le lemme et l'étiquette morphosyntaxique peut aider à lever certaines ambiguïtés :

```
[lemma contains "être|estre" & ud-pos="VERB"]
```

Permet d'exclure les occurrences des noms fust *et* esté *par exemple.*

Pour restreindre la recherche aux lemmes vérifiés, il faut utiliser la propriété « lemmatisation » :

```
[lemma contains "être|estre" & lemmatisation="vérifiée"]
```

Enfin, pour les lemmes vérifiés, il est possible de connaître la source du lemme grâce à la propriété « lemma_src » :

[lemma contains "être" & lemma_src="DMF"]

Cherche les occurrences du lemme *être* provenant du *Dictionnaire du moyen français*.

La liste des sources des lemmes est donnée en Annexe 4 du *Tutoriel BFM* et dans la rubrique « Documentation » du site http://bfm.ens-lyon.fr.

3 Les expressions régulières et les opérateurs

Les opérateurs se présentent comme des caractères spéciaux qui permettent de cibler des variantes graphiques et/ou morphologiques.

Une expression CQL fonctionne à l'image d'une expression régulière classique (voir l'article suivant : Expressions Régulières@Wikipédia). Comme nous l'avons vu précédemment, il s'agit d'une combinaison de caractères associée à divers opérateurs. Ces derniers sont présentés dans le tableau ci-dessous :

Opérateur	Description	Exemple
[ ][2]	Liste de caractères alternatifs les uns par rapport aux autres (« OU ») : Un seul des caractères contenus entre les crochets est pris en compte dans les résultats.	"[ict]" trouvera les caractères « i » ou « c » ou « t ».
[^]	Négation (exclusion) : Tous les autres caractères, sauf ceux qui figurent entre crochets, sont pris en compte.	"[^ict]" trouvera tous les caractères sauf « i », « c » et « t ».
.	Joker de caractère : un point signifie un caractère quelconque, n'importe lequel.	"i." trouvera tous les mots de deux lettres commençant par « i ». "i.." trouvera tous les mots de trois lettres commençant par « i ».
*	Recherche entre 0, une ou n fois ce qui précède l'étoile (un caractère ou groupe de caractères entre parenthèses). Cet opérateur s'utilise souvent en combinaison avec le point.	"i.*" trouvera tous les mots de une ou n lettres commençant par « i ».
?	Le caractère qui précède est facultatif.	"ic?t" trouvera tous les mots commençant par « i » peut-être suivi de « c » et suivi de « t » donc les formes ict, it...
+	Recherche le caractère qui précède le signe, une ou n fois.	"ab+.*" trouvera tous les mots commençant par « ab » ou « abb »... "abes.+" trouvera tous les mots commençant par « abes » avec au moins une lettre supplémentaire. "abes.*" trouvera tous les mots commençant par la racine « abes » dont « abes ».
|	Choix entre plusieurs formes. A l'intérieur des parenthèses, il indique un choix entre les expressions placées avant ou après l'opérateur, le tout étant	"(a|e)m.*" trouvera tous les mots commençant par « a » ou « e » suivi de « m » (équivaut à "[ae]m.*"), suivi de n'importe quoi. "(chevalier|baron).*"

2 Attention : les crochets en dehors des guillemets de type [word="Lancelot" & pos="NOMpro"] signifient que l'expression ne concerne qu'un seul mot. Les crochets seuls [ ] signifient une occurrence quelconque (cf. 5.2.2.2).

	combiné à d'autres caractères placés à l'extérieur des parenthèses.	trouvera toutes les formes commençant par « chevalier » ou « baron ».
( )	Isole une partie de la chaîne de caractères.	Voir l'exemple ci-dessus.
\	Caractère de neutralisation qui permet de rechercher dans le texte des caractères qui feraient normalement partie du langage CQL (tous les caractères décrits dans la colonne de gauche de ce tableau). Il doit précéder le caractère.	"\." trouvera tous les points contenus dans le texte ; "\?" permet de chercher les points d'interrogation ; c'est aussi ainsi qu'il faut procéder pour rechercher les parenthèses et les crochets.
%c	Indique une recherche non sensible à la casse (opposition majuscules/minuscules).	"lancelot"%c trouvera les occurrences de « Lancelot » et « lancelot ».

Pour illustrer la forme parfois complexe d'une expression CQL, prenons la requête suivante :

```
[word="i?ch?(e|i)(x|st|z)(e|i|ui|ez|es)?"%c]
```

Cette requête signifie que l'on cherche :

- un mot (paire de crochets)
- dont la graphie (« `word=` »)
- commence éventuellement (modifieur « `?` ») par « `i` »,
- suivi de « `c` »,
- suivi éventuellement de « `h` »,
- suivi de la voyelle « `e` » ou bien « `i` »,
- suivi de la consonne « `x` » ou bien de « `st` » ou bien de « `z` »
- et enfin ayant éventuellement (« `?` ») l'une des terminaisons suivantes : « `e` » ou bien « `i` » ou bien « `ui` » ou bien « `ez` » ou bien « `es` »
- sans faire de différence entre minuscules et majuscules (« `%c` »).

4 Références

- Liste des étiquettes morpho-syntaxiques Cattex 2009 : http://bfm.ens-lyon.fr/spip.php?article176
- Manuel de référence Cattex 2009 : http://bfm.ens-lyon.fr/spip.php?article323
- Tutoriel TXM pour la BFM : http://bfm.ens-lyon.fr/spip.php?article297

- Manuel de référence TXM : http://textometrie.ens-lyon.fr/html/doc/manual/0.7.9/fr/manual1.xhtml
- Tutoriel du langage CQL (en anglais) : http://cwb.sourceforge.net/files/CQP_Tutorial