

Décrire et manipuler un document numérique

Cours 3 : Techniques de recherche avancées

Loïc Grobol <lgrobol@parisnanterre.fr>

2022-01-31

Google NGram

Permet de faire des requêtes dans Google Books :

- n-grammes de mots
- n-grammes de catégories grammaticales (parties du discours)

Exemple : « vélocipède » vs « vélo » vs « bicyclette »

→ L'évolution des fréquences relatives des n-grammes dans le corpus

Questions d'usage

Dit-on « autant pour moi » ou « au temps pour moi » ?

« par contre » ou « en revanche »

« *the United States are* » ou « *the United States is* »

- Anglais Américain
- Anglais Britannique

Évolution des nominations

Ouvriers, travailleurs ou salariés

Et au pluriel

Féminisation des noms de métiers

et des fonctions

Le joker *

Le token * : remplace n'importe quel mot.

Exemple : `une colère *` donne les fréquences de trigrammes dont les deux premiers mots sont « une » et « colère »

Expressions régulières

Expressions régulières = ER = Regex

- Outil de recherche dans des textes
- Permet de trouver des séquences correspondant à un motif (patron). Ex :
 - Même préfixe / suffixe / contexte
 - Dates
 - Adresses
 - Urls

Définition de critères de recherche pour des chaînes de caractères

Outils :

- Éditeur de texte brut : Notepad++ (Windows), Kate/Gedit (Linux), CotEditor (Mac)...
- Traitement de texte : Google Drive, Word, Writer, Pages...
- Test en ligne : <https://regex101.com/>
- Langage de programmation : java, perl, python, PHP...

Dans un éditeur de texte

On va avoir besoin d'un bon éditeur de texte brut, par exemple :

- Notepad++ (Windows)
- Kate (Linux KDE)
- Gedit (Linux GNOME)
- CotEditor (macOS)
- Quikedit (Android)
- Regex Knife (iOS)

Ou dans <https://regex101.com> ou Google drive en désespoir de cause.

Pour tester :

1. Récupérez le fichier `Duchn-utf8.txt`
2. Ouvrez-le avec l'éditeur de votre choix (Clic-droit sur le fichier + ouvrir avec)
3. Ctrl+f pour accéder à la fonction de recherche/remplacement
 - Cochez « expression régulière »

Principe général

- Règle générale : chaque caractère se reconnaît (*match*) lui-même
 - `madeleine` reconnaît `madeleine` mais pas `Madeleine`
 - `France 2` reconnaît `France 2` mais pas `France2` ni `France 3`
- Cas particuliers : les **métacaractères**
 - Quantificateurs
 - Opérateurs
 - Caractères spéciaux
 - Classes de caractères

Les quantificateurs

Ils portent sur l'élément qui précède et indiquent une répétition

Quantificateur	?	*	+
Signification	« 0 ou 1 fois »	« 0 ou plusieurs fois »	« 1 ou plusieurs fois »

- `jolie?` matche `joli` et `jolie`
- `coup?s` matche `cous` et `coups`, mais pas `coupa`
- `jolie*` matche `joli`, `jolie`, `joliee`, `jolieee...`
- `jolie+` matche `jolie`, `joliee`, `jolieee...`

Exercice : que reconnaissent les expressions suivantes ?

1. `pommes?`
2. `a?typique`
3. `lo*1`
4. `loo*1`
5. `lo+1`

(Tester vos réponses dans un éditeur)

Accolades

Les accolades indiquent un intervalle de répétitions :

- `lo{1,3}ng` reconnaît `long`, `loong` et `looong`
- `lo{3,}ng` reconnaît `looong`, `loooong`, etc.
- `lo{3}ng` ne reconnaît que `looong`

Exercice : écrire une expression régulière qui reconnaît exclusivement `lool`, `loool` et `loooool`.

Les opérateurs

Ensembles de caractères

Les crochets indiquent un ensemble de caractères à reconnaître :

- `[ab]` reconnaît une lettre : `a` ou `b`
 - Avec un tiret : `[a-c]` reconnaît les lettres de `a` à `c`
 - `[a-z]` reconnaît les lettres de l'alphabet en minuscules
 - `[a-zA-Z]` reconnaît aussi les majuscules*
-

Exercice :

Que reconnaît :

1. `[ab][ab]`
2. `[MmRr]aison`

Écrire une expression régulière qui reconnaît exclusivement :

1. `moi`, `toi` et `soi` .
2. `ci`, `si`, `ce` et `se`.
3. Les nombres à deux chiffres.

(Tester vos réponses dans un éditeur)

Avec un circonflexe au début : négation de l'ensemble

- `^[0-9]` reconnaît tout sauf des chiffres
 - `^[A-Z]` reconnaît tout sauf des majuscules sans diacritiques
-

On peut les combiner avec les quantificateurs :

- `[aeiouy]+` reconnaît une suite de voyelles : `a`, `aa`, `ae`, `iey`...
- `[^<]*` reconnaît une suite de caractères différents de `<`

Groupes

Les parenthèses permettent de construire des groupes

- `(ab)+` reconnaît une suite de `ab` : `ab`, `abab`, `ababab`...
- `([ab]c)+` reconnaît `ac`, `bc`, `acac`, `acbc`, `bcac`...
- `([aeiouy][^aeiouy])+` reconnaît une suite de couples voyelle-consonne.

Alternatives

Le *pipe* indique une alternative :

- `machin|truc` reconnaît `machin` et `truc`.
 - `je|tu|nous` reconnaît `je`, `tu` et `nous`.
 - `je|tu|[nv]ous` reconnaît `je`, `tu`, `nous` et `vous`.
-

On les utilise le plus souvent avec des parenthèses

- `(m|r)aison` est équivalent à `[mr]aison`.
- `(il|elle|iel)s?` reconnaît `il`, `elle`, `iel` et leurs versions plurielles.

Exercice : que reconnaissent les expressions suivantes ?

1. `vert(es)?`
2. `(anti|pro)nucléaire`
3. `Wiki[a-z]+`
4. `Wiki[a-z]*`

Caractères spéciaux

Caractère	Sens	Exemple
<code>.</code>	n'importe quel caractère	<code>p.re</code> : <code>pare</code> , <code>pire</code> , <code>père</code> ...
<code>^</code>	début de ligne	<code>^Je</code> (début de ligne)
<code>\$</code>	fin de ligne	<code>informatique\$</code> (fin de ligne)
<code>\b</code>	frontière de mot (début ou fin de mot)	<code>\bjour</code> : <code>journée</code> mais pas <code>ajouré</code>
<code>\n</code>	saut de ligne	<code>a\n</code> : <code>a</code> puis saut de ligne
<code>\t</code>	tabulation	<code>a\t</code> : <code>a</code> puis tabulation

Exercices : à quoi correspondent ces expressions ?

1. `Nom\tPrénom`
2. `\bcap\b`
3. `\bcap`
4. `premier paragraphe\ndeuxième paragraphe`

Désécialisation

On a vu un certain nombre de métacaractères (les caractères avec une signification spéciale).

Pour les reconnaître littéralement (les désécialiser), il faut les faire précéder d'un anti-slash `\` :

- `\^machin` : reconnaît `^machin`
- `3\\.0` reconnaît `3.0` mais pas `310`
- Pareil pour `$`, `[`, `]`, `(`, `)`, `*`, `+`, `?`, `{`, `}`

Et pour représenter un anti-slash ? Pareil

- `machin\\bidule` reconnaît `machin\bidule`

Exercices

Dans le fichier Père Duchesne :

1. Recherchez `^.évolution`
2. Recherchez `r?évolution`
3. Combien y a-t-il de mots...
 1. de 3 lettres finissant par `r` ?
 2. de 4 lettres au plus commençant par `r` ?
 3. commençant par `a` et finissant par `er` ?
 4. contenant la séquence `er` en fin de ligne ?

Classes de caractères

Classe	Signification	Équivalent
<code>\d</code>	un chiffre	<code>[0-9]</code>
<code>\D</code>	tout sauf un chiffre	<code>[^0-9]</code>
<code>\w</code>	les caractères alphanumériques et <code>_</code>	<code>[a-zA-Z0-9âäëèéôûü_]</code>
<code>\W</code>	tout sauf les caractères alphanumériques et <code>_</code>	<code>[^a-zA-Z0-9âäëèéôûü_]</code>
<code>\s</code>	une espace	espace + <code>\t</code> + <code>\n...</code>
<code>\S</code>	tout sauf une espace	...

Notes :

- En général `\X` reconnaît le contraire de `\x`
- Le sens de `\w` et `\s` peut varier suivant les moteurs de regex

Exercices

Faire les quatre premiers niveaux sur : <https://alf.nu/RegexGolf>

1. Warmup
2. Anchors
3. It never ends
4. Ranges

Pour s'entraîner

- Tester ses regex en ligne : <https://regex101.com>
- Exercices : <https://alf.nu/RegexGolf>
- Mots croisés de regex : <https://regexcrossword.com>

Compléments

Stratégie de reconnaissance

- Par principe, un quantifieur est **glouton** (*greedy*) : il reconnaît le maximum de caractères possibles.

- En ajoutant `?` après un quantifieur, on le rend *lazy* : il reconnaît le minimum de caractères.

Par exemple, si l'entrée est `AHHHHHHHHH` :

- `AH+` reconnaît la plus longue suite de caractères : `AHHHHHHHHH`
- `AH+?` reconnaît la plus petite suite de caractères : `AH7`

Références

On peut utiliser des références numérotées pour rappeler le contenu d'un groupe :

`(spam|egg), sausage and \1` reconnaît :

- `spam, sausage and spam`
- `egg, sausage and egg`
- Mais pas `egg, sausage and spam`