Linguistique de corpus

Outils et méthodes de traitement de corpus

Patrick Paroubek

LIMSI-CNRS
Dépt. CHM - Groupe LIR
Bât. 508 Université Paris XI, 91403 Orsay Cedex
pap@limsi.fr

mercredi 12 octobre 2016 / L3 - Cours 4

SGML, HTML, XML
Quelques acronymes

Voici par ordre d'ancienneté décroissant la signification de quelques acronymes :

- SGML = Standard Generalized Markup Language
- XML = Extensible Markup Language
- HTML = Hypertext Markup Language
- XHTML = Extensible Hypertext Markup Language (le plus récent, dont la dernière version du standard proposé par le W3C date du 26 janvier 2000

SGML, HTML, XML
Histoire

- L'ancêtre de SGML est GML (Generalized Markup Language), créé en 1960 par Charles Golfarb, Edward Mosher and Raymon Lorie alors chez IBM, pour annoter la structure des documents techniques.
- le progrès essentiel apporté par GML et ses descendants est de séparer la structure d'un document (division arbitraires hierarchiques, e.g. chapitre etc.) de son contenu (le texte du document) en rendant la structure explicite par l'intermédiaire d'un balisage spécifique,
- de permettre à l'utilisateur de définir son système de structuration, avec ses propres balises et sa propre granularité,
- de pouvoir soumettre facilement (sans avoir à l'inférer à partir du document) la structure du document à des traitements informatiques.

SGML, HTML, XML
HTML

- HTML a été créé en 1980 par le physicien Tim Berner-Lee du CERN (avec son premier prototype de navigateur hypertext ENQUIRE)
- en 1989 T. Berner-Lee et Roger Cailliau font séparément une proposition d'une norme d'hypertexte pour Internet.
- en 1990 leur proposition conjointe est acceptée par le CERN
- la première norme HTML, publiée en 1991, définissait 22 étiquettes d'annotation, dont 13 sont toujours présentes dans HTML-4
- à cette époque HTML était une application de SGML

SGML, HTML, XML
HTML...

- en 1993 T. Berner-Lee et Dan Conolly écrivent la première spécifiation HTML avec leur HTML Internet draft et sa DTD (Document Type Définition) SGML (la DTD est la grammaire définit les balises d'annotation de structure et qui contraint leur emploi).
- la formalisation de HTML en SGML a été le point de départ de son utilisation par le grand public avec le premier browser WEB gratuit MOSAIC de NCSA, avec en particulier la possibilté d'inclure des images dans un document avec une simple étiquette.
- en 1993, Dave Raggettt publie un Internet draft pour HTML qui inclut les tables et les formulaires interactifs.

SGML, HTML, XML
HTML...

Ensuite c'est l'enchaînement des nouvelles versions du standard

- 1995 HTML-2
- 1997 HTML-3.2
- 1997 decembre HTML-4.0
- 2000 le 15 mai, standard ISO/IEC

SGML, HTML, XML
Éléments et attributs

- Pour ce standards SGML, HTML, XML, les balises sont indiquées entre chevrons et délimitent des **éléments** structurels (portion de contenu)
- les éléments peuvent avoir des attributs (qui les caractérisent), aux valeurs plus ou moins codifiées
- ex. un élément titre encadré par ses balises ouvrantes et fermantes : **<TITLE>**Le tour du Monde en 80 jours**</TITLE>**
- ex. un attribut identifiant **ID** et sa valeur E34F23 pour une balise F encadrant un mot : <F **ID="E34F23"**>mot</F>

SGML, HTML, XML
Encodage des caractères

- SGML, HTML et XML utilisent aussi des entités, un symbole commençant par **&** et se terminant par un point-virgule **;** pour des caractères spéciaux ou toute portion de texte arbirtraire (par ex. un nom propre ou un acronyme) que l'entité remplace.
- ex. en sgml le charactères inférieur se code au moyen de l'entité **&lt ;** (*less than*), en html le & se note au moyen de l'entité **&amp ;**
- SGML utilise uniquement le codage ASCII (d'où la nécessité d'avoir des entités pour les charactères accentués),

SGML, HTML, XML

Encodage des caractères...

- en HTML on peut soit encoder les charactères soit avec un encodage arbitraire (ex. ISO-8859-1, UTF8 etc.), soit au moyen des entites définies par HTML (e.g. **&eacute ;** pour é)
- de même en XML, on peut utiliser un encodage utiliser un encodage de charactères arbitraire ou le codage UNICODE, qui est choisi lorsqu'aucun encodage n'est spécifié, dans ce cas les caractères hors ASCII peuvent être représentés au moyen de références numériques (ex. **&#38 ;** en décimal ou **&#x26 ;** en hexadécimal)
- notez que le codage ASCII des caractères est commun au normes SGML, HTML et XML.

SGML, HTML, XML
Structure

- deux de balises ne peuvent avoir des contenus qui se croisent partiellement, ils sont soit totalement disjoints soit l'un contient l'autre,
- ex. <A>*contenu 1*</A>...<B>*contenu 2*</B> ou bien <A>*contenu 1*...<B>*contenu 2*</B>...</A> ;
- la structure d'un document est donc nécessairement un arbre (pas de croisement de balises).
- en SGML la DTD (Document Type Définition) associée à un document est la grammaire formelle qui définit les balises d'annotation de structure et qui contraint leur emploi dans des documents qui sont considérés "valides" pour la DTD.
- contrairement a SGML, XML autorise la validation d'extraits de documents

SGML, HTML, XML
DTD

Il existe des outils (parsers, i.e. analyseurs syntaxiques, par ex. xmllint sous Linux) pour valider automatiquement des documents SGML ou XML par rapport à une DTD. Il existe aussi des langages pour exprimer des contraintes sur les annotations XML, les deux plus utilisés sont RelaxNG et les *s*chemas XML du consortium W3C.

Vérification de la strucure d'un document XML par rapport à une DTD de référence (grammaire définissant la structure des documents) avec la commande :.

```
xmllint -noout -dtdvalid passage.dtd lemon.xml
```

Le fichier `passage.dtd` contient la DTD et le fichier `lemon.xml` contient le document xml à valider :

```
<?xml version="1.0" encoding="UTF-8"?>
<Document dtdVersion="1.0" file="lemon">
<Sentence id="E1">
<T id="E1T1" start="727" end="731">Voeux</T>
<T id="E1T2" start="733" end="734">de</T>
<T id="E1T3" start="736" end="742">Jacques</T>
...
<G id="E1G2" type="GP">
<W id="E1F2" tokens="E1T2"/>
<W id="E1F3" tokens="E1T3"/>
<W id="E1F4" tokens="E1T4"/>
</G>
...
<R type="MOD-N" id="E1R1">
<modifieur ref="E1G2"/>
<nom ref="E1G1"/>
</R>
```

Le fichier passage.dtd contient la DTD :

```
<?xml version='1.1' encoding="UTF-8"?>
<!ELEMENT Document (MSTAG*, Sentence*)>
<!ATTLIST Document dtdVersion CDATA #FIXED "1.0"
file CDATA #IMPLIED>

<!-- ================================-->
<!ELEMENT MSTAG (fs)>
<!ATTLIST MSTAG id ID #REQUIRED>
<!ELEMENT fs (f+)>
<!ELEMENT f (symbol | vAlt)+>
<!ATTLIST f name CDATA #REQUIRED>
<!ELEMENT symbol EMPTY>
<!ATTLIST symbol value CDATA #REQUIRED>
<!ELEMENT vAlt (symbol+)>
<!-- ================================-->
<!ELEMENT Sentence ( (T|G|W)+, R*, M* ,NE*)>
<!ATTLIST Sentence trust CDATA #IMPLIED
id ID #IMPLIED>
<!-- ================================-->
<!ELEMENT T (#PCDATA)>
<!ATTLIST T id ID #REQUIRED
                start           CDATA #REQUIRED
                end             CDATA #REQUIRED
>
```

Dans l'exemple précédent de définition d'une DTD, la définition des **éléments** (`ELEMENT`) utilise les opérateur de répétition non vide d'un élément `+` (e.g. `symbol+`), de répétition indéfinie (y compris vide) `*`, d'alternative `|`, les parenthèses `()` et l'opérateur de séquence : `,` (la virgule).
Notez que les éléments terminaux de la structure des éléments (feuilles de l'arbre) sont soit des `CDATA` (Caratère DATA) séquences de caractères pour lesquelles les balises XML ne sont pas interprétées comme telles, mais seulement comme des caractères), soit des `PCDATA` (Parsed Caractère DATA), i.e. des séquences de caractères où les balises XML sont prises en compte.

Toujours dans l'exemple précédent de définition d'une DTD, la définition des **attributs** (`ATTLIST`) d'un élément va définir leurs identifiants, leurs types (chaîne de caractères non-analysée `CDATA`, analysée pour son balisage `PCDATA`, identifiant unique `ID`) et leurs natures (par ex. une valeur constante sera `#FIXED`, un attribut optionnel sera `#IMPLIED` et un attribut obligatoire sera `#REQUIRED`).

En cas d'erreur dans la structure XML du document un message apparait :

```
xmllint --noout --dtdvalid passage.dtd lemon_BAD.xml
lemon_BAD.xml:8: parser error : Opening and ending tag
mismatch: TOTO line 8 and T
<TOTO id="E1T5" start="750" end="750">:</T>
```

Bonne pratique : afin de préserver la séquence de caratères du document original sans aucun transcodage des caratères comme par exemple `&`, `<` ou `>`, il est recommandé, dans un document annoté avec des balises xml (et pas dans la DTD !), d'utiliser un balisage explicite des portions originales de document avec les balises **CDATA** :
`<![CDATA[` ...le contenu original... `]]>`
Dans ce cas la seule séquence de caractères interdite est `]]>`, mais elle peut être représentée
avec : `<![CDATA[]]]><![CDATA[]]]><![CDATA[>]]>`

Exemple d'utilisation des balises CDATA dans un document balisé en XML

```
<Sentence id="E1">
<T id="E1T1" start="727" end="731"><![CDATA[Voeux]]></T>
<T id="E1T2" start="733" end="734"><![CDATA[de]]></T>
<T id="E1T3" start="736" end="742"><![CDATA[Jacques]]></T>
```

Création d'une page HTML minimaliste avec echo et l'opérateur de redirection de flux uniquement.
Le contenu de la page :

```
<HMLT>
<BODY>
Hello
</BODY>
</HMLT>
```

Les commandes :

```
echo "<HTML>" > pageweb.html
echo "<BODY>" >> pageweb.html
echo "Hello" >> pageweb.html
echo "</BODY>" >> pageweb.html
echo "</HTML>" >> pageweb.html
```

Un résultat similaire peut être obtenu avec une seule commande :

```
echo -n -e "<HTML>\n<BODY>\nHello\n</BODY>\n</HTML>\n
> pageweb.html
```

ou bien avec votre éditeur de texte favori (mais sauvegarder la page en mode texte brut, sans format).

Pratique :

1. Affichez une page web et visualisez le code source avec la fonction "view source" de votre navigateur pour s'assurer que la page est codée en HTML.
2. Téléchargez la page.
3. Filtrer la page pour sortir toutes les balises présentes et en faire le décompte au moyen des outils du shell bash.

Solution :

1. Nous allons afficher chaque balise sur une seule ligne en ajoutant avec `sed` un caractère spécial, par ex. * (vérifiez qu'il n'est pas déjà présent dans votre page web avec la commande `grep`) puis en transcodant avec `tr` ce caractère en saut de ligne (code ASCII en octal `\012`)

```
cat ma_page_web.html | sed -e 's/</*</g' | sed -e 's/>/>*/g' | tr '*' '\012'
```

2. Ensuite nous sélectionnons uniquement les lignes qui contiennent des balises avec `grep`
, et nous les trions par ordre alphabétique

```
| egrep "<" | sort
```

3. Puis nous extrayons de ces lignes le premier mot avec `awk`

```
| awk '{ print $1;}' |
```

4. nous trions ces lignes par ordre alphabétique (`sort`), nous écrasons les lignes en doublons (`uniq`) et les comptons (`uniq -c`)

```
| sort | uniq -c
```

5. et finalement nous trions ces statistiques (`sort`) selon la valeur numérique (`sort -n`) décroissante (`sort -r`) du décompte (`-k 1,1`) qui se trouve être le premier mot de chaque ligne et constitue donc une clé de triage commençant au premier mot et finissant au premier mot.

```
| sort -n -r -k 1,1
```

La commande finale est donc :

```
cat ma_page_web.html | sed -e 's/</*</g' | sed -e 's/>/>*/g' | tr '*' '\012' |
    egrep "<" | sort  | awk '{ print $1;}' |  sort | uniq -c |
    sort -n -r -k 1,1
```

Pour info, un tutoriel en anglais sur les commandes shell est disponible a l'URL :
http ://www.ee.surrey.ac.uk/Teaching/Unix/index.html

et en français à l'URL :
http ://fr.wikibooks.org/wiki/Programmation_Bash/Notions_essentielles_du_shell_bash

Pour créer une page WEB contenant le message “Bonjour” :

```
echo '<HTML><BODY>Bonjour</BODY></HTML>' > ma_page_web.html
```

Pour l'afficher dans le navigateur, utiliser le protocole `'file:'` suivit du chemin d'accès depuis la racine vers l'endroit où se trouve le fichier, dans la zone de saisie des URLs, par ex.

`file:/home/tim/ma_page_web.html`

Les documents HTML
Pratique

- Télécharger une page web (par exemple un livre du site ABU) au format HTML.
- Visualiser le contenu avec l'item de menu, par exemple avec le navigateur FireFox ce menu est : `Tool/Web Developer/Page Source`
- Produire avec des commandes shell la liste des caractères de ce fichier triée par ordre décroissant de fréquence.
- Que remarque t'on ?
- Produire avec des commandes shell la liste des balises HTML présentes dans ce fichier triée par ordre décroissant de fréquence.
- Que remarque t'on ?

Les documents HTML
Pratique

Rappel, pour l'affichage d'un document HTML il faut prendre en compte :

- le format du fichier original et l'encodage de ses caractères,
- le format et l'encodage de ses caractères du fichier sauvegardé sur votre ordinateur par le navigateur WEB,
- l'outils de visualisation (application) du document HTML sauvegardé.

A chacune de ses étapes, les caractères spéciaux peuvent être corrompus (transcodage) ou mal-affichés (les fontes ne sont pas disponible ou l'application ne les reconnaît pas).

- *Produire avec des commandes shell la liste des balises HTML présentes dans ce fichier triée par ordre décroissant de fréquence.*
- Solution :

```
cat mon_fichier | sed -e 's/</\n</g' |  sed -e 's/>/>\n/g' | egrep '^<' | sort
  | uniq -c | sort -k 1,1 -r -n
```

- Les balises ouvrantes et fermantes apparaissent avec des décomptes égaux (heureusement !).

- *Produire avec des commandes shell la liste des caractères de ce fichier triée par ordre décroissant de fréquence.*
- Solution :

```
cat mon_fichier | sed -e 's/./&\n/g' | sort | uniq -c | sort -k 1,1 -r -n
```

- Très probablement le caractère le plus fréquent est le `e` ou bien les caractères < et >.

- *Transformer avec des commandes shell le résultat précédent (distribution des caractères) en une page web stockée dans le fichier* `/tmp/nwpage.html` *(attention à l'affichage des caractères* `&`*,* `<` *et* `>`*.*
- Solution, on suppose que le résultat précédent est stocké dans le fichier `/tmp/res.txt` :

```
echo "<HTML>" > /tmp/nwpage.html
echo "<head>" >> /tmp/nwpage.html
echo "<meta charset=\"utf-8\"/>" >> /tmp/nwpage.html
echo "</head>" >> /tmp/nwpage.html
echo "<BODY>" >> /tmp/nwpage.html
cat /tmp/res.txt | sed -e 's/&/\&amp;/g' > /tmp/f1
cat /tmp/f1 | sed -e 's/</\&lt;/g' > /tmp/f2
cat /tmp/f2 | sed -e 's/>/\&gt;/g' > /tmp/f3
cat /tmp/f3 | tr '\012' '*' | sed -e 's/*/<BR>\n/g' >> /tmp/nwpage.html
echo "</BODY>" >> /tmp/nwpage.html
echo "</HTML>" >> /tmp/nwpage.html
```