FERREIRA Dany
HARS Antoine

LO17 – Automne 2013

Projet : Rapport 1

Préparation et indexation du corpus

Sommaire

Introduction

Tout au long de l'UV LO17, nous devons réaliser un projet d'indexation et de recherche d'information à l'aide d'un ensemble de pages internet.

L'objectif du projet étant d'indexer l'archive et de stocker les index dans une base de données afin de pouvoir l'interroger selon différents critères (date, thème, sujet, contenu, image, région, ville, …).

Ce rapport présente les méthodes utilisées pour préparer le corpus et réaliser son indexation.

I. Préparation du corpus

On souhaite réaliser une archive du site de la chaîne d'information LCI, et plus particulièrement des pages relatives aux informations internationales (rubrique "Monde" de la page d'accueil de LCI).

Ci-dessous un exemple de page web de LCI privée des bandeaux latéraux et du bandeau frontal (parties non utiles pour notre traitement), avec les différentes rubriques de la page que l'on devra traiter.

La Une

A voir aussi

Focus

Gros titres

Rappels

Les cinq rubriques des pages sont donc les suivantes : “Une”, “À Voir Aussi”, “Le Focus”, “Les Gros Titres” et “Les Rappels”.
Notre corpus initial contient 328 fichiers html du site LCI datés entre le 25 février 2005 et le 2 mars 2006.
En partant de ces fichiers, il faut produire à l’issue de cette première partie, un seul et unique fichier contenant pour chaque page LCI-Monde les informations relatives à la partie information sélectionnée (les cinq rubriques).
Le fichier souhaité devant être structuré, il nous a paru légitime de choisir le format XML pour le construire. Ce fichier structuré permet de retrouver l’identifiant de la page (le nom et la position du fichier), la date ainsi que les différentes rubriques et leur identifiant.

1. Les différentes étapes réalisées

Pour chacune des étapes, nous avons développé les scripts perl selon la même méthode : tout d’abord tester des scripts simples sur un seul fichier, puis veiller à intégrer ces scripts à d’autres scripts plus complets. Enfin, après avoir appliqué les scripts sur quelques fichiers et validé le résultat obtenu, nous avons développé des scripts permettant de s’appliquer sur l’ensemble des fichiers d’un dossier donné.

a) Etape 1 : Repérage et normalisation de la partie informative des pages LCI-Monde

Nous avons détecté que la partie informative d’une page est contenue entre les expressions “IBL_ID=27303” ou “Blc=27303” et “IBL_ID=27916 - Temps” ou “Blc=27916, [0-9]”. Nous avons vérifié cela sur quelques fichiers.
Grâce à une commande “grep” nous avons vérifié la présence de deux de ces balises dans chaque fichier.
Ensuite, nous avons pour chaque fichier html, créé un nouveau fichier html avec le même nom, contenant uniquement la partie informative que nous souhaitions.

b) Etape 2 : Mise sur une seule ligne

Afin de faciliter la suite des traitements de la préparation du corpus, nous avons créé pour chacun des fichiers html précédemment obtenus un nouveau fichier html du même nom en ayant supprimé tous les caractères “line feed” \n et “carriage return \r”, ce qui a pour résultat de nous donner des fichiers ne contenant qu’une seule ligne.

c) *Etape 3 : Création d'une ligne par rubrique*

Afin de créer un fichier xml pour chacun des fichiers html obtenus à l'étape 2, nous avons commencé par structurer les fichiers. Nous avons repéré les balises html et arguments css (class) permettant de repérer de manière certaine les 5 rubriques. Nous avons vérifié que ces éléments d'identification de parties étaient les mêmes dans plusieurs fichiers que nous avons choisi aléatoirement. Ensuite à l'aide d'expressions régulières, nous avons organisé nos fichiers xml afin d'obtenir une rubrique par ligne tout en identifiant à quelle rubrique se référait chaque ligne (avec une balise ouvrante et une balise fermante pour chaque rubrique).

Par exemple, pour la rubrique "Une", l'expression régulière est de la forme s/class="S431"/\n<UNE>class="S431"/. La balise de classe S431 nous permet de repérer le début de la rubrique "Une".

d) *Etape 4 : Création du fichier structuré*

Pour chaque ligne des fichiers xml obtenus à l'étape 3, nous avons dû récupérer les informations qu'elles contenaient.
Par exemple, pour la rubrique "FOCUS", les informations suivantes étaient à récupérer : l'url de l'article, le titre de l'article, la date de l'article, l'url de l'image, le résumé de l'article, l'email de l'auteur et le nom de ce dernier.

Voici un exemple de résultat obtenu :

```
<FOCUS>
      <urlArticle>"/news/monde/0,,3204806-
      VU5WX0lEIDUy,00.html"</urlArticle>
      <titreArticle>Un demi-frère de Saddam Hussein
arrêté</titreArticle>
      <dateArticle>27/02/2005</dateArticle>
      <urlImage>http://s.tf1.fr/mmdia/i/13/2/627132.gif</urlImage>
      <resumeArticle>Sabaoui Ibrahim al-Hassan, ex leader baasiste,
est soupçonné de diriger la rébellion contre le gouvernement
irakien.</resumeArticle>
      <mailto>dstraus@tf1.fr</mailto>
      <auteur>D.S. (avec AFP)</auteur>
</FOCUS>
```

Les expressions régulières nous ont permis, après analyse des fichiers, de récupérer les détails souhaités (sous rubriques) grâce à des classes css utilisées dans les balises html, comme pour le passage de l'étape 2 à 3 avec les rubriques.
Nous obtenons finalement un fichier xml contenant l'ensemble de l'information des pages html initiales, sous le formalisme xml demandé.

Lors de cette étape 4, nous avons taché d'obtenir pour chaque fichier des pages LCI, le corpus

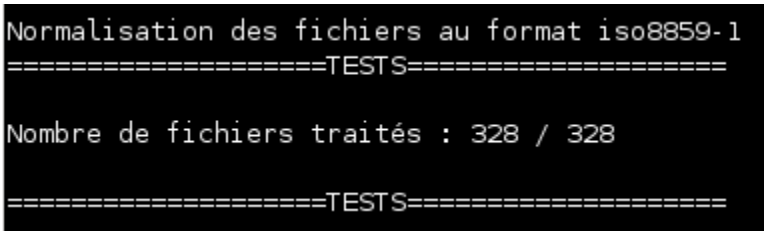
correctement réalisé lui correspondant puis de réitérer tout le traitement effectué en amont sur les fichiers afin d'obtenir un seul et même corpus.

2. Tests de vérification

Afin de vérifier la bonne exécution des scripts sur l'ensemble des fichiers lors des passages entre étapes, nous avons élaborés différents tests.

a) Etape 1 à étape 2

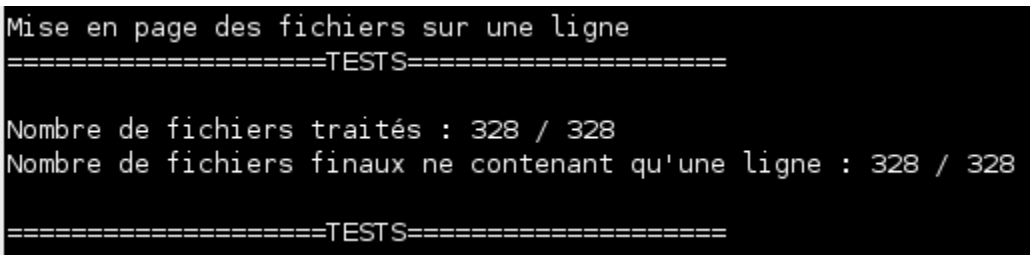
Afin de tester la bonne exécution de notre script pour le formatage en iso8859-1, nous comparons le nombre de fichiers obtenus dans notre répertoire de sortie LCI par rapport au nombre de fichiers contenus dans notre répertoire LCI en entrée :



Nous pouvons donc observer sur cette capture d'écran que nous avons exactement le même nombre de fichiers dans le répertoire avant formatage que dans le répertoire après formatage.

b) Etape 2 à étape 2

Concernant la mise en page des fichiers sur une seule ligne en retirant tous les retours chariot, nous avons choisi de compter le nombre de fichiers traités comme précédemment et de le comparer au nombre de fichiers ne contenant qu'une seule ligne après application du script. Afin de compter le nombre de lignes des fichiers traités, nous avons utilisé la commande "wc -l". Cependant nous avons observé que cette commande ne nous retournait pas le chiffre 1 mais 0. Après réflexion, nous avons compris que cette commande retournait le nombre d'occurrences du retour chariot que le traitement effectué sur les fichiers avait pour but de tous les retirer. Nous avons donc testé pour un nombre de lignes égal à 0 mais un nombre de mots strictement positif.



Nous pouvons voir sur la capture d'écran que nous avons le même nombre de fichiers entre le répertoire en entrée et le répertoire en sortie et que le nombre total de fichiers ne contenant qu'une seule ligne est égal au nombre de fichiers à traiter.

c) *Etape 3 à étape 4*

Afin de tester que les rubriques de chacun des fichiers étaient bien récupérées lors de l'étape 3, nous avons observé le nombre de fichiers traités comme les deux cas précédents, puis nous avons cherché à savoir si le nombre de rubriques récupérées était identique pour un même fichier en entrée et en sortie. Pour cela, nous avons dû veiller à garder les éléments d'identification des différentes rubriques pour comparer en amont du traitement et en aval de celui-ci :

```
Création d'une ligne par rubrique
====================TESTS====================

Nombre de fichiers traités : 328 / 328
Nombre de fichiers finaux contenant 5 rubriques : 241 / 241
Nombre de fichiers finaux contenant 4 rubriques : 87 / 87
Nombre de fichiers finaux contenant 3 rubriques : 0 / 0
Nombre de fichiers finaux contenant 2 rubriques : 0 / 0
Nombre de fichiers finaux contenant 1 rubrique : 0 / 0
Nombre de fichiers finaux contenant 0 rubrique : 0 / 0

====================TESTS====================
```

Nous obtenons donc, dans notre cas, 241 fichiers en sortie sur 241 en entrée qui contiennent 5 rubriques différentes et 87 fichiers en sortie sur 87 en entrée avec 4 rubriques (les rubriques manquantes peuvent différer entre les fichiers). La somme de ces deux nombres nous donne bien 328 fichiers ce qui est correct dans ce cas-ci. Nous avons veillé à observer le nombre de fichiers contenant un nombre de rubriques inférieur à 4 pour s'assurer qu'il n'y avait pas d'éventuelles erreurs de traitement pour cette étape.

d) *Etape 4 au corpus*

Pour le traitement de l'étape 4, nous avons veillé à comparer le nombre de fichiers en entrée et en sortie afin de s'assurer qu'aucun fichier n'avait été mis de coté. De plus, l'affichage du nombre de fichiers traités tout au long du processus de préparation du corpus nous permet de constater que nous ne laissons aucun fichier de côté.
Comme autre test, nous nous avons comparé entre chaque fichier en entrée et chaque fichier en sortie correspondant, le nombre d'éléments de chacune des rubriques, par exemple, dans le cas des gros titres, nous regardons si les fichiers en entrée contiennent bien le même nombre de gros titres que les fichiers en sortie correspondants. Si ce n'est pas le cas, nous obtenons un nombre non égal à la somme des fichiers traités.

```
Création d'un corpus par fichier
====================TESTS====================

Nombre de fichiers traités : 328 / 328
Nombre de fichiers dont la UNE a été correctement traitée : 328 / 328
Nombre de fichiers dont les VOIRAUSSI ont été correctement traités : 328 / 328
Nombre de fichiers dont les GROSTITRES ont été correctement traités : 328 / 328
Nombre de fichiers dont le FOCUS a été correctement traité : 328 / 328
Nombre de fichiers dont les RAPPELS ont été correctement traités : 328 / 328

====================TESTS====================
```

Dans le screenshot précédant, nous pouvons observer que nous avons un nombre de fichiers traités égal au nombre de fichiers à traiter et que chacune des rubriques a donc bien été traitée.

```
Création d'un corpus pour tous les fichiers
====================TESTS====================

Nombre de fichiers traités : 328 / 328

====================TESTS====================
```

Concernant la deuxième partie de l'étape 4, c'est à dire la création d'un corpus contenant les données de tous les fichiers, nous effectuons seulement une comparaison du nombre de fichiers en amont et en aval du traitement car le traitement effectué est exactement le même que pour la première partie de l'étape 4, ce qui diffère est seulement le fichier en sortie qui est le même (corpus_lci.xml) pour tous les fichiers ici.

II. Indexation du corpus

Dans cette partie, nous nous sommes attachés à créer les fichiers inverses en indexant le corpus que nous avons obtenu dans la première partie, en récupérant retirant les mots ayant peu d'intérêt pour l'indexation du corpus, en récupérant les lemmes de chacun des mots présents. Ces fichiers inverses nous permettent de décrire les documents à l'aide de tout ou une partie des éléments contenus dans le fichier xml obtenu auparavant.

1. Les différentes étapes réalisées

Pour chacune des étapes, nous avons développé les scripts perl selon la même méthode : tout d'abord tester des scripts simples sur un seul fichier, puis intégrer ces scripts à d'autres scripts plus complets. Enfin, après avoir appliqué les scripts sur quelques fichiers et validé le résultat obtenu, nous avons exécuté les scripts sur l'ensemble des fichiers.

a) Etape 1 : Création d'une stop-list

Afin d'établir une stop-list de mots à retirer du corpus, nous avons calculé le coefficient *tf * idf (vu en cours)* pour chacun des mots du corpus puis nous avons défini le seuil en nous basant sur le premier mot différent des mots usuels de la langue française (le mot "et" par exemple, une conjonction de coordination parmi d'autres) et de la conjugaison des verbes les plus courants ("été" par exemple).
Nous avons donc défini le seuil au *tf * idf* au mot "pays", seulement il se trouve que son *tf * idf* est exactement le même que celui du mot "sa" qui correspond à un mot pouvant faire parti de la stop-list. Nous avons donc veillé à récupérer le mot "sa" dans la stoplist et à garder "pays" dans le corpus.

Le calcul du coefficient *tf * idf* s'appuie sur la fréquence d'un mot dans un document et sur la fréquence de documents pour un mot donné. Nous avons donc choisi la page comme unité documentaire. Ainsi on s'intéresse à la fréquence des mots dans une page, même s'il apparaît dans différents titres ou résumés.
En effet, l'article comme unité documentaire, nous aurait amené à mettre en place des traitements complexes pour chaque fichier et lors des recherches nous souhaitons pouvoir accéder à une page web.

b) Etape 2 : Création des lemmes

Afin de créer le fichier .txt qui contient pour chaque mot, le lemme associé, nous avons, tout d'abord veillé à récupérer les mots du corpus privé des mots contenus dans la stop-list. Puis nous avons appliqué sur cette liste de mot le script *successeurs_2013.pl* donné qui est chargé de générer la liste des successeurs pour chaque lettre des mots d'une liste de mots.
Pour finir, nous appliquons, sur ce fichier de successeurs, le script *filtronc.pl -v* qui nous permet d'obtenir le lemme pour chaque mot de la liste.

a) Etape 3 : Création des fichiers inverses

Les fichiers inverses sont des fichiers permettant de décrire les documents à l'aide de tout ou une partie des éléments contenus dans ce fichier xml.
Les fichiers inverses contiennent une première colonne contenant un identifiant (mot, date, e-mail…) et dans les colonnes suivantes le nom du fichier html dans lequel il apparaît, puis la rubrique, etc …
Chaque fichier obtenu correspond à une balise ou un ensemble de balise (date de la page, rubriques, titre+résumé, date de l'article….)

Grâce au script *index.pl* nous avons pu créer les fichiers inverses pour les informations des balises du type dateArticle, auteur etc … sauf bien sûr pour la balise urlArticle où il n'y a pas d'intérêt.

Le script *newindexMot.pl* crée lui un fichier inverse pour les mots du titre et du résumé des articles car récupérant en entrée un flux du type "mot page rubrique urlArticle" obtenu avec le script *newsegmente.pl*.

Ci-dessous un xemple pour le mot économiste apparaissant à la fois dans les titres et des résumés d'articles appartenant à la rubrique "Focus" ou "Gros titre".

```
économiste      lci-monde-2005-11-08.xml   focus
     "/news/monde/0,,3261023-vu5wx0leiduy,00.html    lci-monde-2005-
11-09.xml  grostitre  /news/monde/0,,3261023-vu5wx0leiduy,00.html
     lci-monde-2005-11-10.xml   grostitre  /news/monde/0,,3261023-
vu5wx0leiduy,00.html  lci-monde-2005-11-15.xml   grostitre
     /news/monde/0,,3263155-vu5wx0leiduy,00.html     lci-monde-2005-
11-16.xml  grostitre  /news/monde/0,,3263155-vu5wx0leiduy,00.html
     lci-monde-2005-11-17.xml   grostitre  /news/monde/0,,3263155-
vu5wx0leiduy,00.html
```

2. Tests de vérification

a) Test pour la stop-list

Afin de tester si l'application de la stoplist sur le corpus nous renvoyait bien un corpus privé des mots de cette liste, nous avons cherché à observer que des mots comme "une" ou "et" (présents dans la stop-list) étaient absents du corpus en sortie, ce qui s'avérait bien être le cas.

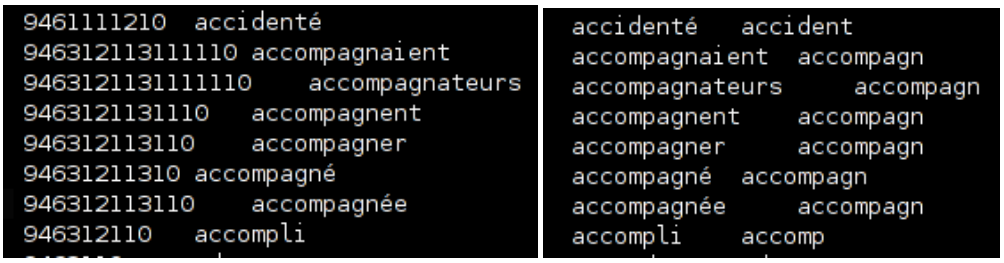
b) Test pour les lemmes

Le lemme d'un mot est trouvé en cherchant dans la liste des successeurs de ses lettres la plus longue sous-chaîne se terminant par le plus grand maximum précédé par un 1. Donc il est nécessaire que le minimum soit 1 et non pas un minimum local car cela nous donnerait un lemme trop général alors qu'il pourrait être plus précis. Nous perdrions de l'information dans l'indexation en procédant de la sorte. Plus le lemme peut être précis (donc grand), plus la qualité de l'information qu'il véhicule est appréciable.

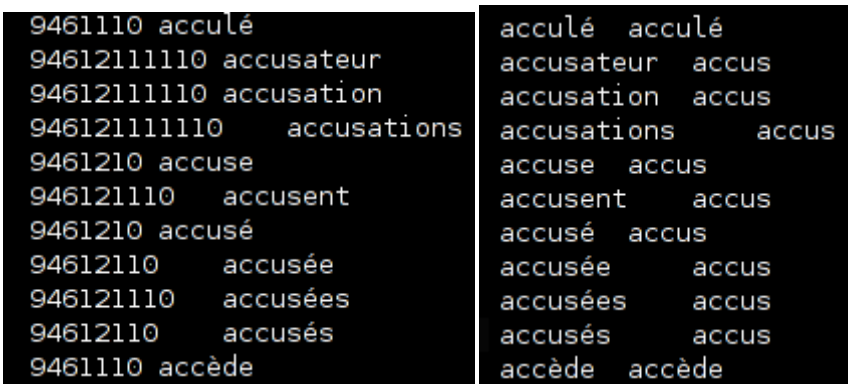
Il est nécessaire pour obtenir un bon lemme d'utiliser le plus grand maximum pour regrouper au sein d'un même lemme, le plus de mots avec une signification égale sinon nous perdons en qualité et nous avons 2 mêmes lemmes presque pareils alors que ce n'est pas utile et cela rendra les recherches ultérieures utilisant ces lemmes peu fructueuses

Il est aussi intéressant pour l'élaboration des lemmes d'essayer d'obtenir la plus longue sous-chaîne possible car nous obtenons dans ce cas un lemme pour un groupe de mots précis alors que dans le cas où nous prenons un lemme plus court, il va regrouper plus de mots mais ceux-ci n'exprimeront pas tous exactement la même idée. La qualité sera donc réduite.
Nous pouvons prendre par exemple le cas où nous prenons comme lemme le mot "acce" : ce lemme regrouperait donc tous les mots commençant par "acce" comme accepter et accentuer. Or ces 2 mots n'expriment pas du tout le même concept et donc la qualité du lemme est faible.

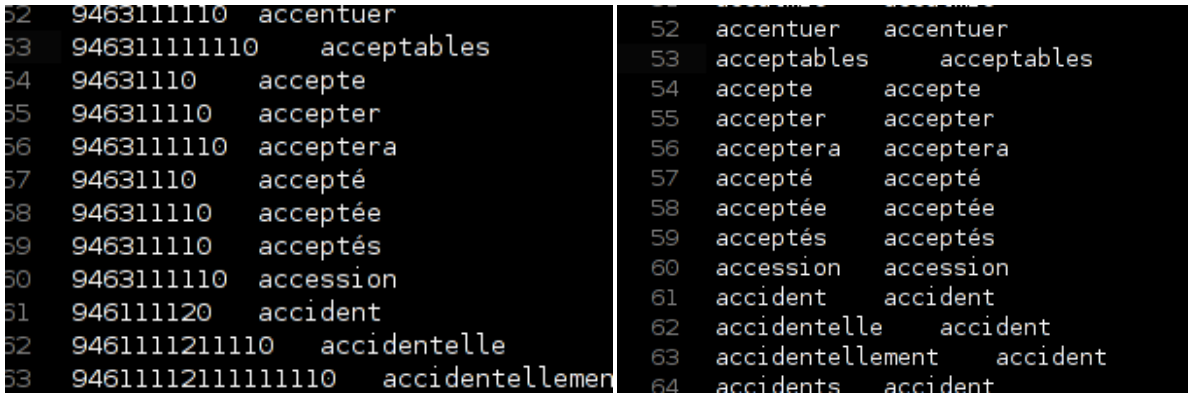
Nous avons observé la qualité des lemmes que nous obtenons pour 3 familles de mots :



Tout d'abord pour la famille de mots du lemme "accompagn", nous remarquons que nous obtenons pour ce lemme le tronc "9463121131" commun avec les mots "accompagnaient", "accompagnateurs", "accompagnent", "accompagner", "accompagné", "accompagnée". On peut observer que ces mots font partie de la même famille donc ce lemme est de plutôt de bonne qualité.



Le lemme "accus" regroupe quant à lui les mots "accusateur", "accusation", "accusations", "accuse", "accusent", "accusé", "accusée", "accusées", "accusés" qui sont des mots qui expriment le même concept. Le mot "acculé", quant à lui, a pour lemme le mot "acculé" alors qu'il a 4 lettres en commun avec le premier lemme.



Enfin dans le cas des mots de la famille “accept”, on peut voir que notre algorithme pour déterminer les lemmes montre quelques limites, car nous avons exactement le même tronc entre le mot “accentuer” et “accepter”. Si nous prenons le lemme “acce”, nous obtenons un lemme trop petit par rapport à la taille des mots s’y référant donc l’algorithme prend pour lemme chacun des mots qui rencontrent ce problème. Dans notre cas, il s’agit des mots allant de “accentuer” à “acceptés”.

c) Test sur les fichiers inverses

Pour vérifier la qualité des fichiers inverses, nous avons regardé si quelques mots comme “économistes”, nous les retrouvions bien dans les rubriques et les fichiers indiqués dans les fichiers inverses, ce qui s’avère être le cas pour les quelques mots testés.

Conclusion

Au cours de ces TD’s, nous avons réalisé la préparation d’un corpus à partir de documents web (fichiers html), ainsi que l’indexation de ce corpus au moyen du langage PERL. Ceci afin de pouvoir interroger notre base de données plus tard afin de répondre aux requêtes des utilisateurs.

Grâce au processus mis en place, nous sommes passés de 328 fichiers html à un unique fichier xml donc structurant l’information. Cela rend ainsi son accès plus facile.

Une étape délicate aura été la lemmatisation car l’algorithme de troncation ne donne pas toujours le résultat que l’on attendrait.

Nous obtenons finalement un document structuré xml ne contenant que des lemmes.