TP1 - Préparation et indexation du corpus

Artus VUATRIN - Jianzhe ZHONG

April 18, 2016

1 Introduction

Dans ce premier TD de LO17 s'étendant sur quatre séances, nous allons créer un système d'indexation sur un ensemble de bulletins électoniques. Ce système pourra ensuite etre intérrogé afin de trouver tous les articles qui correspondront à des requetes posées en langage naturel. Dans la première partie, nous étudierons la préparation du corpus avant d'étudier son indexation.

2 Préparation du Corpus

Dans un premier temps, nous allons suivre les étapes qui nous ont mené à un fichier XML structuré contenant toutes les informations du corpus, regroupant tous les articles en un fichier. Il faut tout d'abord trouver dans la page HTML toutes les balises qui nous permettront d'identifier de manière unique chaque information capitale de chaque article.

Figure 1: Position des éléments à extraire dans la page HTML

Lors du lancement du script créant le fichier XML, un autre script est lancé sur chaque fichier du corpus avec le nom du fichier passé en paramètre, une fois que la balise racine < *corpus* > a été ouverte. Le premier élément à être extrait par ce sous-script est le nom de fichier, il est facilement récupérable grâce au paramètre envoyé au sous-script qui n'est autre que le nom du fichier. Ensuite, le numéro de bulletin doit être extrait. Pour cela, on doit identifier les balises et mots entourant ce numéro dans le fichier HTML. Le numéro de bulletin se trouvant toujours entre les éléments < *span class* = "*style*32" > *BE France* et &*nbsp*; < /*span* >, on peut l'extraire avec ces lignes de code perl:

```
if($ligne =~ /<span class="style32">BE France(.*)  </span>/ig)
   $bulletin = $1
```

Si on considère que $*ligne* est la ligne courante de l'article qui est en train d'être parcourue, on obtient dans $*bulletin* le numéro de bulletin, dans le cas où la ligne valide l'expression régulière. Nous nous sommes assurés que la classe *style*32 n'est utilisée que pour le numéro de bulletin avant de continuer. Pour rechercher la date, le titre et le texte, il nous a suffi de la même manière d'identifier les balises entourant l'information désirée de manière unique afin de s'assurer que toutes ces informations seront celles désirées. Pour ces éléments, on trouve les expressions régulières :

- < *span class* = "*style*42" > &*nbsp*; &*nbsp*; ([0−9]?[0−9]/[0−9]?[0−9]/[0−9][0−9][0−9][0−9]) < *span* > pour la date;
- < *p class* = "*style*96" >< *span class* = "*style*42" > (.∗) < *br* >< /*span* > pour la rubrique;
- < *span class* = "*style*17" > (.∗) < /*span* >< /*p* > pour le titre.

Pour le contact, il est nécessaire de faire un traitement supplémentaire, en effet, on désire avoir le lien sans les balises HTML < *a* >. Pour les enlever, on va d'abord extraire les informations de contact avec la regex suivante :

```
<p class="style44"><span class="style85">(.*)<\/a><\/span><\/p><\/td>
```

Puis, on retire les balises en utilisant la suppression par expression régulière de perl, ce qui nous donne

```
$contact = ($textRaw =~ s/<a href=".*">//g);
```

Le lien de contact avec les informations supplémentaires seront dans la variable $*contact*.

Pour le texte, nous avons du prendre en compte que les paragraphes sont, dans la plupart des articles, sur plusieurs lignes. Il faut donc trouver la marque indiquant le début du texte. Une fois le début du texte trouvé, un drapeau est levé afin d'indiquer qu'on se trouve dans du texte. Puis, on continue à boucler sur les lignes normalement avec le drapeau levé, ce qui indique que toutes les lignes seront enregistrées entre les balises textes. Une fois que la marque de fin a été trouvée, on retire toutes les balises inutiles du texte que l'on vient d'extraire et on enregistre le texte formaté, ensuite on rabaisse le drapeau indiquant que les lignes parcourues ne sont plus du texte.

A tout moment du parcours du texte, il est possible de trouver une image. Celles-ci peuvent être de deux formats différents.

- Les images ayant un lien contenant "streaming". Le lien est trouvé dans l'attribut href, et les crédits sont toujours situés à la ligne suivante. En revanche, on a jamais de legende pour ce type d'image.
- Les images provenant du site bulletins electroniques, ayant une URL de la forme : *www.bulletins − electroniques.com/Resources_fm/actualites.∗?.jpg*. On sait que pour ce type d'image, on a la légende sur la ligne suivante, et les crédits sur la ligne d'après.

Les images sont placées dans un tabeau associatif contenant les attributs : url, légende et crédits.

Une fois que toutes les lignes d'un fichier ont été parcourues, on a un ensemble de variables ou de tableaux contenant les informations de l'article. Toutes les balises sont écrites avec leur contenu. Les balises principales (rubrique, date, titre, image, ...) sont toujours écrites, tandis que leurs balises filles (si elles en ont) ne sont pas écrites dans le cas où leur contenu n'a pas été trouvé.

Une fois le corpus complet, un autre script se charge de vérifier si les données contenues dans le XML sont complètes. Pour cela, le script va parcourir tout le fichier XML en vérifiant s'il existe des balises ne contenant aucune données. A chaque fois qu'il rencontre les balises $< fichier >< /fichier >$, il enregistre le nom dans une variable temporaire, puis continue sa recherche. Dans le cas où il trouve une balise vide dans ce fichier, il écrira sur la sortie standard le numéro de fichier ayant l'erreur ainsi que la balise trouvée vide. Ce sera ensuite le role du developpeur de vérifier si cette omission est voulue ou s'il s'agit d'une erreur d'expression régulière. Un fichier *log* est adapté pour recevoir les erreurs renvoyées par ce script.

3 Indexation

3.1 Choix de l'unité documentaire

Pour effectuer l'indexation, on doit d'abord choisir l'unité documentaire. On choisit que le document sera équivalent à l'article. En effet, plusieurs articles peuvent se trouver dans un seul bulletin. En effectuant une recherche précise, choisir le bulletin comme unité documentaire pourrait sortir des articles qui n'ont pas de rapport avec la requete. En revanche, cela nous permettrait de ne pas louper d'informations importantes qui se trouvent dans des articles différents mais publiés dans un seul bulletin. En regardant les différents articles publiés dans un bulletin, on se rend rapidement compte que les articles n'ont pas toujours de rapport entre eux et choisir les bulletins comme unité documentaire renverrait trop souvent des articles inappropriés.

3.2 Stop-list

Le fichier segmente_TT.pl nous permet de séparer les mots un par un. Avant de récupérer les mots, il enleve la ponctuation dans le fichier en argument, ainsi que les majuscules. Il dispose convenablement tous les mots du corpus en en placant un par ligne.

La valeur $tf_{i,j}$ représente le nombre de mot i dans l'article j. Pour obtenir les $tf_{i,j}$, on cherche à créer un fichier avec trois colonnes : mot, valeur de $tf_{i,j}$ et le nom de l'article correspondant. On utilise pour cela la commande :

```
cat ./bulletins.xml | ./segmente_TT.pl -f | sort | uniq -c |
sed 's/^[[:space:]]*//g' > tfij.txt
```

En utilisant le fichier segmente_TT.pl, on obtient la liste de mot et le nom du fichier correspondant. Avec des commandes shell, on peut calculer le nombre d'occurence par fichier.

df_i est le nombre de documents dans laquelle le mot i apparaît. On calcule ensuite le coefficient idf_i avec la formule ci-dessous :

$$idf_i = log_{10} \frac{N}{df_i}$$

Le graphique 2 montre le nombre d'occurence des mots dans le corpus en fonction de leur nombre d'occurence, et le graphique 3 les montre en fonction de leur poids. Comme on peut le remarquer, ces graphiques sont fortement similaires, mais calculer idf_i nous permet de donner plus de poids aux mots les plus importants.

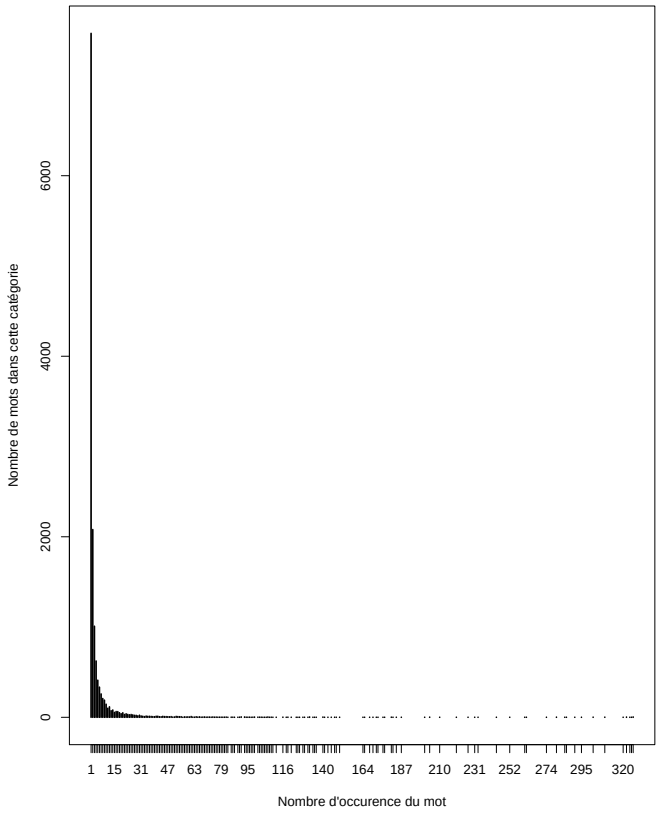

Figure 2: Nombre de mot dans le corpus ayant un nombre occurence donné

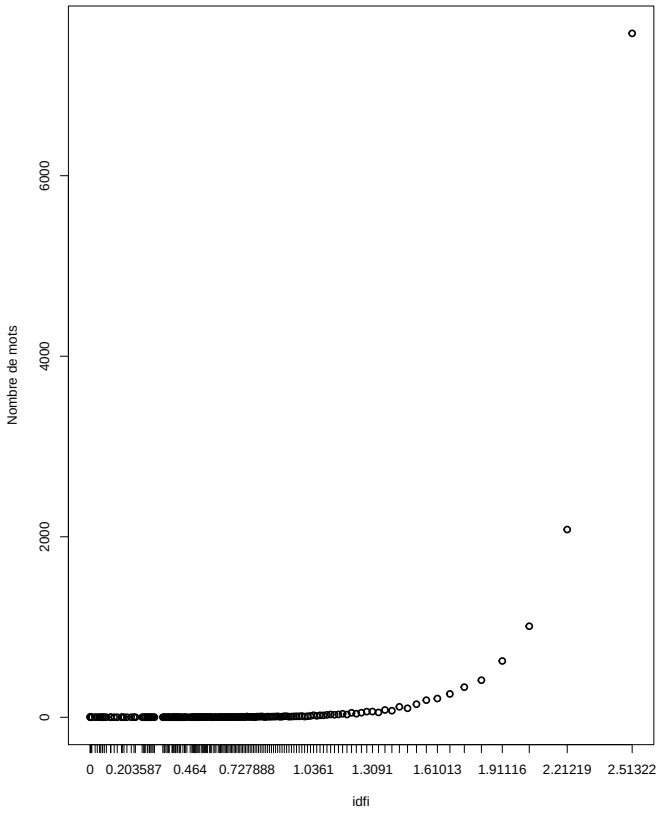

Figure 3: Nombre d'occurence de mots dans le corpus en fonction de leur poids

Pour le calcul du $tf_{i,j} \times idfi$, on parcours les fichiers précédents (contenant les idf_i et $tf_{i,j}$) et on dispose 3 colonnes : mot, fichier, $tf_{i,j} \times idfi$. Afin de récupérer les valeurs simplements, on utilise les tableaux associatifs avec les regex de perl qui nous permettent de classer facilement les valeurs que l'on trouve en parcourant les fichiers d'entrée.

Le parcours du fichier des idf_i se fera comme tel :

```
while (my $row = <$fh>) {
      ($dfi, $mot) = split(/\s+/, $row);
      $dfi_words{$mot} = $dfi;
}
```

Puis, lors de la récupération des $tf_{i,j}$, on fait de même, ce qui rend le calcul des $tf_{i,j} \times idfi$ immédiat.

```
while (my $row = <$fh>) {
       ($frequence, $mot, $fichier) = split(/\s+/, $row);
       print  $dfi_words{$mot} * $frequence. " ".  $mot." ".$fichier. "\n";
}
```

Pour trouver le seuil de sélection des mots de la stop-list, on recherche le mot ayant la plus petite valeur idf_i pouvant être recherché via une requete. On sélectionne le mot "recherche" qui répond à ces critères. Son

poids est de 0.287908. On peut donc supprimer tous les mots ayant un idf_i inférieur, les mots supprimés sont les articles ou prépositions très utilisées en francais (à, le, la,...) Ensuite, on doit trouver un filtre à mettre sur les nombres. Le corpus contient un grand nombre de mots référencés qui sont en réalité des nombres. Nous faisons donc le choix de supprimer des mots indexés tous les nombres ayant strictement moins de 4 chiffres. On peut ainsi référencer les dates du dernier millénaire, sans garder tous les nombres n'ayant aucune valeur informative. On garde aussi les grands nombres qui peuvent être spécifiques à un domaine et donc informatifs. On utilise pour cela le script newcreeFiltre.pl nous permet de créer des ces filtres, c'est à dire crée d'autres scripts permettant d'éliminer un mot ou de le remplacer par un autre.

3.3 Lemmes et fichiers inverses

Pour construire la liste des mots du corpus en face de leurs lemmes, on va utiliser l'algorithme de troncation. Dans un premier temps, on doit calculer leurs successeurs. Pour cela, on utilise le fichier successeurs.pl. Il construit une liste de mots, et place en face de chaque mot son successeur. On obtient par exemple pour les mots commencant par le préfixe 'aff' :

sucesseur	mot
97711120	affaire
977111210	affaires
9771141110	affectant
97711410	affecte
977114110	affecté
9771141110	affectent
977114110	affecter
9771142110	affection
9771142110	affective
9774131110	affichant
97741310	affiche
977413120	affiché
9774131210	affichée

Puis, on utilise le fichier lemme.pl pour trouver leurs lemmes, ce traitement renvoie les lemmes suivants pour les mots sélectionnés :

mot	lemme
affaire	affaire
affaires	affaire
affectant	affect
affecte	affect
affecté	affect
affectent	affect
affecter	affect
affection	affect
affective	affect
affichant	affich
affiche	affich
affiché	affich
affichée	affich

On remarque que les mots de la même famille sont séparés, leurs lemmes sont différents ce qui prouve que l'algorthme fonctionne correctement.

En réutilisant le fichier newcreeFiltre.pl, on peut remplacer les mots dans le fichier XML par leur lemme.

Grace au fichier indexText.pl, on peut récupérer un ensemble d'informations (rubrique, fichier et numéro) d'un mot demandé. Par exemple, on obtient ce type de résultats pour ces trois mots pris au hasard :

mesurant	focus	69179.htm	291	en direct des laboratoires	69539.htm	292
capabilities	focus	71358.htm	292			
solitaire	focus	67383.htm	292			

Sinon, on peut donner une balise en argument au fichier index.pl, ce qui nous permet de classer tous les articles en fonction du contenu de la balise pour un article.

Par exemple, on classe les articles selon la date, et on observe que seulement les articles ne sont parus que à 25 dates différentes entre juin 2011 et juin 2014. Les résultats obtenus sont montrés sur la figure suivante.

```
10      09      2013    73875.htm*      284     73876.htm*      284     73877.htm*      284     73878.htm*      284     73879.htm*      284     7
3880.htm*       284     73881.htm*      284     73882.htm*      284     73883.htm*      284     73884.htm*      284
24      01      2012    68881.htm*      266     68882.htm*      266     68883.htm*      266     68884.htm*      266     68885.htm*      266     6
8886.htm*       266     68887.htm*      266     68888.htm*      266     68889.htm*      266
29      09      2011    67794.htm*      261     67795.htm*      261     67796.htm*      261     67797.htm*      261     67798.htm*      261     6
7799.htm*       261     67800.htm*      261     67801.htm*      261     67802.htm*      261     67803.htm*      261     67804.htm*      261
30      08      2011    67553.htm*      260     67554.htm*      260     67555.htm*      260     67556.htm*      260     67557.htm*      260     6
7558.htm*       260     67561.htm*      260
21      12      2011    68638.htm*      265     68639.htm*      265     68640.htm*      265     68641.htm*      265     68642.htm*      265     6
8643.htm*       265     68644.htm*      265     68645.htm*      265     68646.htm*      265
23      12      2013    74744.htm*      287     74745.htm*      287     74746.htm*      287     74747.htm*      287     74748.htm*      287     7
4749.htm*       287     74750.htm*      287     74751.htm*      287     74752.htm*      287
22      11      2011    68273.htm*      263     68274.htm*      263     68275.htm*      263     68276.htm*      263     68277.htm*      263     6
8278.htm*       263     68279.htm*      263     68280.htm*      263     68281.htm*      263     68283.htm*      263
29      06      2012    70428.htm*      271     70429.htm*      271     70420.htm*      271     70421.htm*      271     70422.htm*      271     7
0423.htm*       271     70424.htm*      271     70425.htm*      271     70426.htm*      271
31      05      2012    70161.htm*      270     70162.htm*      270     70163.htm*      270     70164.htm*      270     70165.htm*      270     7
0166.htm*       270     70167.htm*      270     70168.htm*      270     70169.htm*      270     70170.htm*      270
20      03      2014    75457.htm*      289     75458.htm*      289     75459.htm*      289     75460.htm*      289     75461.htm*      289     7
5462.htm*       289     75463.htm*      289     75464.htm*      289     75465.htm*      289     75466.htm*      289
29      11      2012    71612.htm*      275     71614.htm*      275     71615.htm*      275     71616.htm*      275     71617.htm*      275     7
1618.htm*       275     71619.htm*      275     71620.htm*      275     71621.htm*      275
21      06      2011    67068.htm*      258     67071.htm*      258
26      03      2012    69533.htm*      268     69534.htm*      268     69535.htm*      268     69536.htm*      268     69537.htm*      268     6
9538.htm*       268     69539.htm*      268     69540.htm*      268     69541.htm*      268     69542.htm*      268     69543.htm*      268
31      01      2013    72113.htm*      277     72114.htm*      277     72115.htm*      277     72116.htm*      277     72117.htm*      277     7
2118.htm*       277     72119.htm*      277     72120.htm*      277     72121.htm*      277     72122.htm*      277
21      10      2013    74171.htm*      285     74172.htm*      285     74173.htm*      285     74174.htm*      285     74175.htm*      285     7
4176.htm*       285     74167.htm*      285     74168.htm*      285     74169.htm*      285     74170.htm*      285
21      12      2012    71835.htm*      276     71836.htm*      276     71837.htm*      276     71838.htm*      276     71839.htm*      276     7
```

Figure 4: Fichier inverse obtenu à partir de la date (*index.pl date corpus.xml*)

4 Conclusion

Avec ces TD, on a préparé le terrain pour les requetes de l'utilisateur. On obtient à la fin de ces séances un corpus modifié qui contient les lemmes des mots de chaque article, seulement pour les mots ayant une valeur informative. Ainsi en trouvant le lemme d'un mot recherché par l'utilisateur, on peut sortir les articles contentant le même lemme via le script indexText.pl sur le corpus contenant les lemmes. Ce qui nous renvoie des articles en rapport avec le mot recherché. On a donc déjà un système de recherche simple pour un seul mot. Pour obtenir les mêmes résultats avec plusieurs mots, il suffirait de trouver l'intersection des fichiers contenant l'un des lemmes.