Université des Sciences et de la
Technologie Houari Boumediene

Traitement Automatique du Langage Naturel

Rapport du mini projet
TALN

Rédaction:
MOULAI Hassina Safaa
MEDJKOUNE Sofiane
HADJERES Yasmine
HOUACINE Naila Aziza
M2 SII

Professeur
Mr. GUESSOUM Ahmed

December 17, 2018

Contents

Chapter 1

Introduction

[8 a 10 lignes]

Problématique

[travail demandé dans l'énoncé]

Description du système globale

1.1 Description des étapes

Comme le montre le schéma du fonctionnement de notre système ci-dessous, la création de notre dictionnaire historique c'est fait en plusieurs étapes:

- **Récolte de ressources :** Recherche des ressources fiable, en se basant sur un processus automatisé pour aspirer des sites à partir d'URLs et un processus de téléchargement de PDFs de manière automatique.

- **Nettoyage :** Extraction de contenu utile des résultats de la recherche obtenu lors de l'étape précédente, cela en utilisant *BeautifullSoup* pour les sites aspirés et *Textract* pour les PDFs téléchargés.

- **Organisation automatique du Corpus :** génération et organisation automatique de nos fichiers (contenant les textes nettoyés) par périodes et genres.
  - **Par période :** en exploitant les méta-données contenu dans les sites et PDF tel que le nom de l'auteur, qui nous permet d'obtenir sa date de naissance / mort via *l'API de wikipédia* et ainsi de le classé dans une période précise.
  - **Par genre :** en utilisant un modèle de la RI (le *modèle vectoriel*) ainsi que le champ lexical des genres sélectionnés.
- **Génération du corpus XML :** cela est effectué en passant chacun de nos textes par un module de découpage en phrase basé sur les expressions régulière, puis exploitant le résultat de celle-ci pour remplir nos fichiers .xml de nos phrase baisées (pour l'insertion de balise nous avons utilisé *xml.etree.ElementTree*).

- **Exploitation de la BDD offline de Al Maany :** après téléchargement de l'application *Al Maany* (en *.apk*) nous y trouvant la base de donnée des mots/ sens, que nous récupérant de manière structuré pour une utilisation dans les étapes suivantes.

- **Construction de dictionnaire historique XML :** A partir du corpus balisé, de fonction de stemming (basé sur *ISRIStemmer*), d'étiqueteur POS (*Stanford POSTagger*) ainsi que quelque fonctions de normalisation et tokenization (*pyarabic*) dédiés à la langue arabe et enfin des données extraites de la BDD de *Al Maany* offline, nous remplissons notre dictionnaires historique, celui-ci constitué de 28 dictionnaires XML (1 par lettre alphabétique).

- **Enrichissement du dictionnaire historique:** afin de continuellement enrichir notre dictionnaire historique, nous avons deux méthodes possibles:
  - **Mode Online :** Extraire des exemples d'utilisation du site Al Maany et les proposer au lexicographe. pour cela il s'agit de répéter les étapes 1 et 2 dédiées aux URLs.
  - **Mode Offline :** Permettre la mise à jour manuelle d'entrée dans le dictionnaire.

1.2 Schéma résumant le système

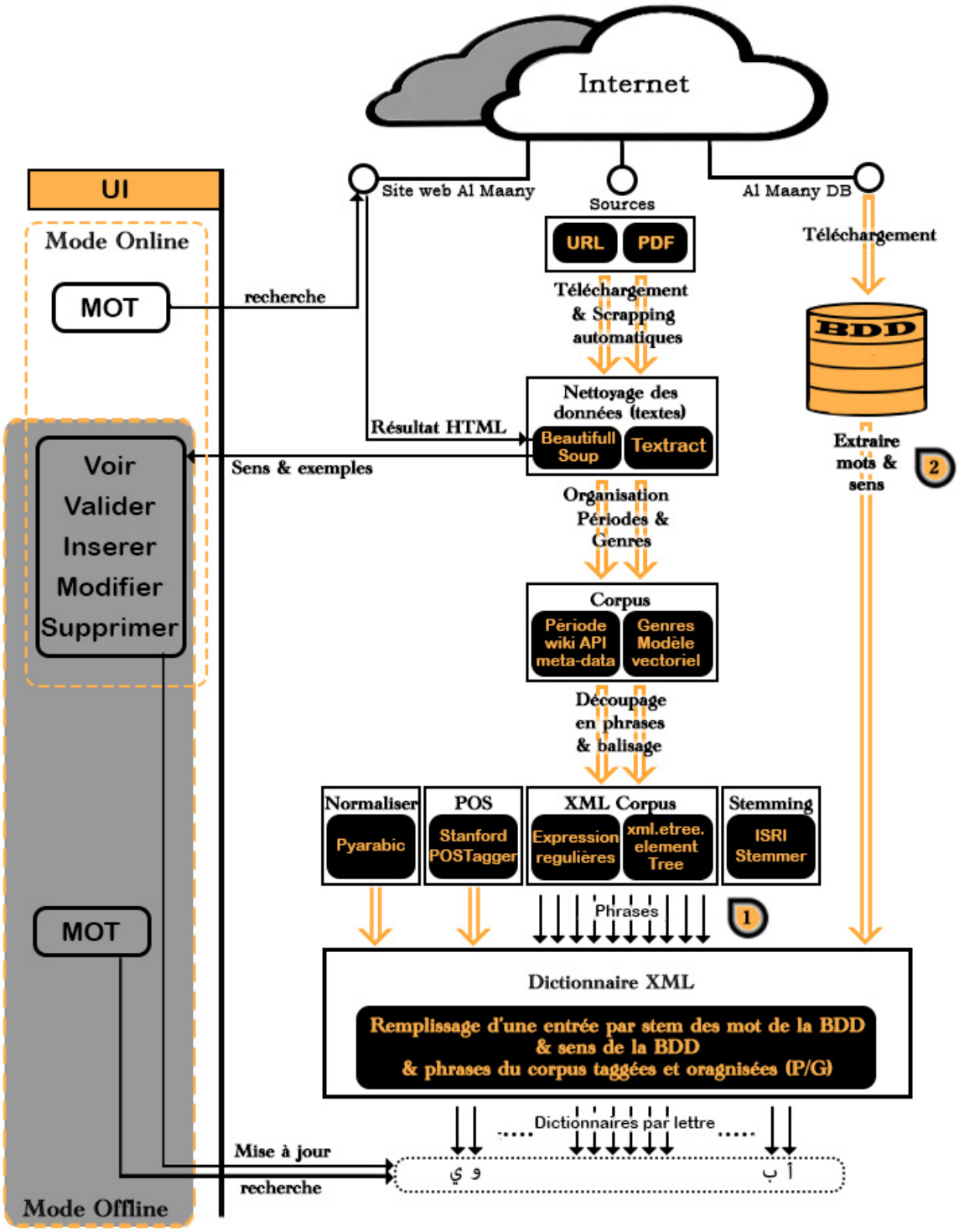
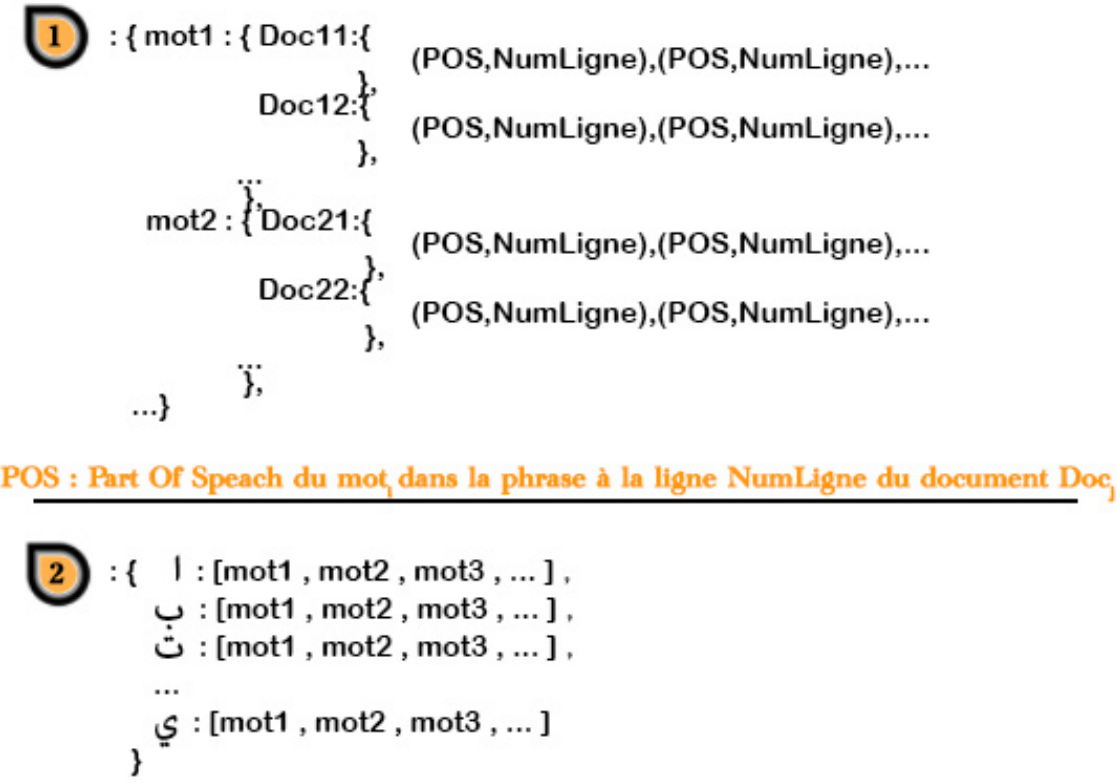


Figure 1.1: Schéma de fonctionnement de notre système.



Chapter 2

Définition des périodes et genres

2.1 Présentation de la langue arabe

[jess]

2.2 Histoire de la langue arabe

[jess] [citer les source de jess et de naila pour justifier nos périodes]

2.3 Liste des périodes retenues

[jess] [nom de la période:interval de temps en miladi : intervale de temps en hdjri : quelques auteurs a titre d'exemple]

2.4 Liste des genres retenus

[naila]
Vu les évènements marquants des différentes période précédemment citées et en tenant compte des genres dominants nous avons sélectionné 3 genres pour classer nos textes et poèmes de chaque période, Ces genres sont :

Politique $\Leftrightarrow$
Religion $\Leftrightarrow$
Littérature $\Leftrightarrow$

Chapter 3

Extraction et nettoyage de texte arabe

3.1 Extraction et nettoyage à partir de lien

3.1.1 Description de la méthode

l'Extraction d'information à partir des sites webs est une pratique trés courante de nous jour selon de le besoin , on peut ,par exemple ,vouloir recuperer un article de journal et on aimerai bien cibler que le contenu de l'article .

On se confronte alors face à un problème les pages webs sont generalement bourrées de ce que appelera déchets , par déchets on veut dire tout ce qui est different de notre cible qui est le contenu de l'article comme:

du code javascript , du text des pubs , les balises html ...

Solution:

Pour palier à cette problèmatique on a exploité des librairies qui existent permettant de *nettoyer au maximum* la page web qu'on desire aspirer.(Beautiful Soup sous python vu en tp)

3.1.2 Fonctionnement

Differentes approches peuvent appliquer tel que:

- Combiner BeautifulSoup avec des expressions régulières
- Expressions régulières seulement
- BeautifulSoup seulement

Il est primordiale de noter que pour aspirer une page web et cibler un certain contenu il faut connaitre et etudier le code source de cette derniere avec une certaine precision .

Technique choisie

On a essayé d'éviter les expressions régulières car Beautiful soup offre assez d'outils qui permettent de faire un nettoyage presque parfait .

- Nettoyage en supprimant le contenu des balises Script et style

- Recuperation du texte arabe seulement en utilisant une expression régulière
- Analyse de la sortie des deux premieres étapes puis application de split et contains pour bornée le texte voulu

3.2 Extraction et nettoyage à partir de PDF/text

3.2.1 Description de la méthode

[À côté de l'extraction de textes à partir de liens, il a fallu élargir nos sources de données pour un corpus plus riche. Nous avons donc téléchargé des documents PDF et nous sommes passés à l'extraction du contenu de ces derniers. Pour notre application nous avons choisi de construire deux instances du corpus, une instance sous forme de fichiers texte et une seconde instance sous forme de fichiers XML. Nous justifions notre choix par les raisons suivantes :

- Permettre à l'utilisateur de naviguer plus rapidement dans le corpus.
- Gagner du temps lors de l'affichage du contenu du corpus en évitant les traitements sur les fichiers XML.
- Permettre d'utiliser le corpus en dehors des limites du projet, en effet il sera plus simple de manipuler des fichiers texte que des fichiers XML.

Problématique:
À travers nos tests, nous avons compris qu'il était impossible (vu la durée limitée que nous avons) d'extraire du contenu de documents PDF qui contiennent plus que du texte uniquement (tel que le **Qoran** par exemple).

3.2.2 Fonctionnement

Pour cette extraction, notre choix s'est vite portée sur le package **Textract**.

Avant de passer à l'utilisation de ce package, il faudra bien sûr télécharger les fichiers PDF dont on veut extraire le contenu. pour cela il suffit de lir ele contenu d'une URL et d'écrire le résultat dans un fichier en output. Plusieurs packages en Python peuvent être utilisés, dans notre cas nous avons utilisé **urllib**.

Pour ce qui de **Textract** son utilisation est des plus simples, elle consiste à envoyer en paramètre à la méthode **process** le chemin de notre document PDF, l'encodage du fichier qui sera en **UTF-8** dans notre cas et plus important la méthode utilisée pour lire le document, dans notre cas **pdfminer** comme suit :

```
import textract
import codecs

text = textract.process('exemple.pdf', method='pdfminer',encoding="utf-8")
```

Figure 3.1: l'appel de fonction

Chapter 4

Découpage de texte arabe en phrases

4.0.1 Description de la méthode

[jess laki al khat]
methode existante -¿ probleme rencontré -¿ methode choisie

4.0.2 Fonctionnement

[jess laki al khat]

Chapter 5

Organisation automatique du corpus

5.1 Organisation par période

5.1.1 période contenue dans la source

Pour certain sites aspirés (Diwan) les poèmes étaient organisées par auteur et par periodes , donc il était assez simple de les classer en récuperant l'information (periodes,auteur) à chaque fois.

5.1.2 période à extraire à partir de méta-data des textes

Tous les sites n'offre pas le luxe de l'information sur la periode de leurs textes (poèmes ,textes narratifs ...).Il a fallu donc trouver une solution pour remédier à ce problème . On peut résumer ce qu'on a proposé dans les points suivants:

- Extraction du nom de l'auteur du texte ou poème d'un site aspirer (hikam , diwan)
- Recherche des méta-données , année de naissance , année de décés lancée sur WIKIPEDIA.
- Extraction de la periode selon la date de naissance et déces de l'auteur (exemple : en moyenne un auteur commence à écrire à partir de 30 ans si il est né en 690 alors on aura 720 et classifera la date résultante selon notre decoupage de periode)

5.2 Organisation par genre

Maintenant que chaque texte et poème de notre corpus est classé dans une période, nous devons trouver un moyen de les classer par genre, car cette information n'était pas donnée initialement par nos sources.

L'approche à la quelle nous avons pensé consiste à exploiter un modèle de **recherche d'information (RI)** pour calculer la similitude entre un texte et une requête,
tel que le texte correspond à un document entier de notre corpus et la requête à un lexique (champ lexical) d'un des 3 genres que nous avons choisi.

Dans l'utilisation classique des modèles de recherche d'information nous calculant la similitude d'un ensemble de document par rapport a une requête, afin de ne garder que les documents ayant la plus grande similitude.

Le but étant de classer un Document dans la catégorie ayant la similitude maximale avec son lexique, nous avons inversé le principe c'est-à-dire nous procédons à l'appariement de chaque document pris indépendamment des autres avec 3 requêtes, afin de garder la requête qui nous permet de maximiser la similitude.

5.2.1 Modèle de Recherche d'information

Parmi les modèles de recherche d'information les plus connus nous pouvons énumérer : le modèle booléen , le modèle vectoriel , le modèle probabiliste , ...

Notre choix c'est porté sur **le modèle vectoriel** pour les raisons suivantes :

- modèle booléen : trop rigide, ne représente pas le degré d'appartenance.
- modèle vectoriel : plus souple que le modèle booléen, donne de bons résultats.
- modèle probabiliste : représente bien la vrai nature du problème (basé sur les probabilités), donne de bons résultats, mais nécessite une prés-sélection de documents jugés pertinents à partir d'un échantillon et cela par un autre modèle ou un humain.

La formule de Jaccard est utilisé pour ce modèle:

$$\text{Sim}(Doc_j, requete_q) = \frac{\Sigma_{i=1}^{N_q} w_{ij} * w_{iq}}{\Sigma_{i=1}^{N_q} w_{ij}^2 + \Sigma_{i=1}^{N_q} w_{iq}^2 - 2 * \Sigma_{i=1}^{N_q} w_{ij} * w_{iq}}$$

Avec:
N_q : le nombre de terme de la requête q.
Doc_j : le j^{eme} document a classer.
$requete_q$: la q^{eme} requête a tester, $q \in 1, 2, 3$
w_{ij} : le poids du i^{eme} terme de la requête dans le j^{eme} document.
w_{iq} : le poids du i^{eme} terme de la requête dans le q^{eme} requête.

5.2.2 Récolte du lexique de chaque genre

Pour nos trois genres nous avons eu besoins de trois requête, chacune étant une liste de mots du lexique de ce genre.

Pour cela nous avons récupéré le lexique de chaque genre (Politique, Religion, Littérature) en français[https://www.rimessolides.com/motscles.aspx?m=litt%c3%a9rature], que nous avons par la suite traduit en arabe en exploitant l'API de Google traduction

googletrans[https://pypi.org/project/googletrans/], ainsi chaque requête (lexique) est enregistré dans un fichier texte.

5.2.3 Application du modèle vectoriel

On calcule la similitude de chaque document de notre corpus avec les trois requêtes (lexiques arabes) selon la formule du modèle vectoriel donnée plus haut, nous classons chaque document selon le genre qui maximise la similitude.

5.3 Construction du Corpus DOC et du corpus XML

Pour notre application nous avons choisi de construire deux instances du corpus, une instance sous forme de fichiers texte et une seconde instance sous forme de fichiers XML.

Nous justifions notre choix par les raisons suivantes :

- Permettre à l'utilisateur de naviguer plus rapidement dans le corpus.
- Gagner du temps lors de l'affichage du contenu du corpus en évitant les traitements sur les fichiers XML.
- Permettre d'utiliser le corpus en dehors des limites du projet, en effet il sera plus simple de manipuler des fichiers texte que des fichiers XML.

5.3.1 Corpus au format doc

L'automatisation de l'extraction de la période ne permettra de construire des fichiers sources qui vont contenir des paires de [période x : ensemble de liens], en voici un petit aperçu pour les liens du site **alDiwan** :

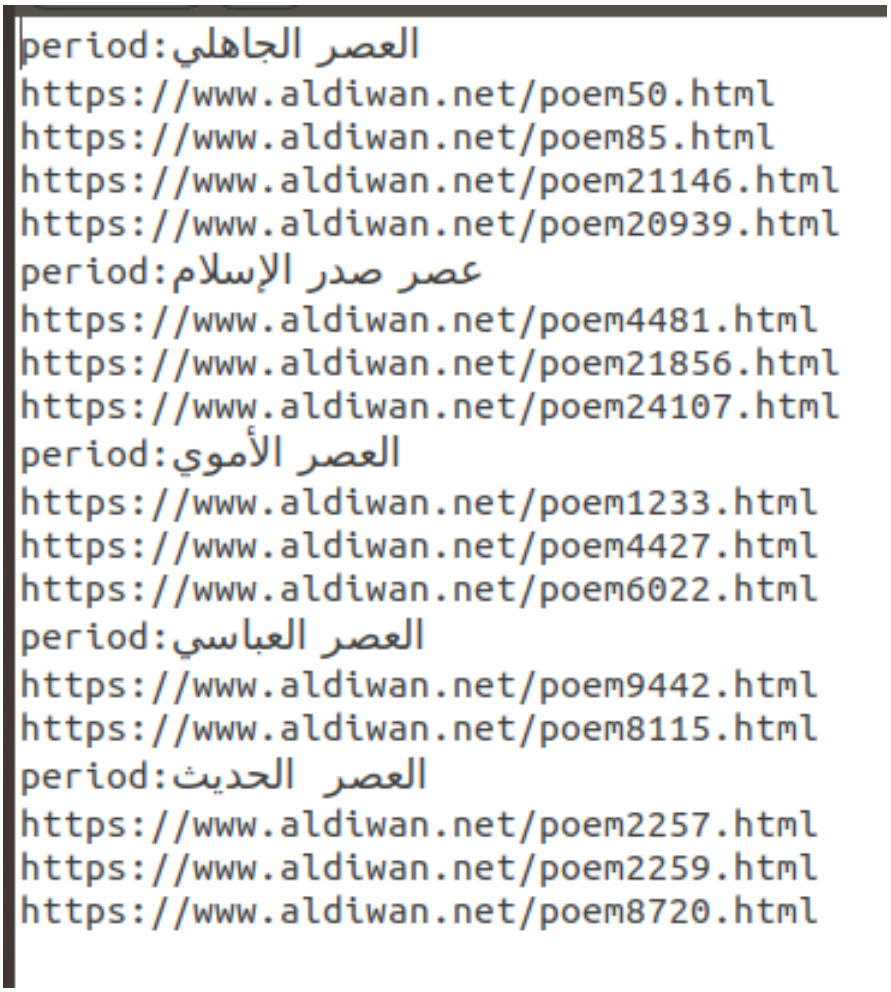


Figure 5.1: Structure du fichier contenant les liens organisés par periodes

Nous avons généré un fichier du genre pour chaque site sur lequel nous avons extrait du contenu. La génération de ces fichiers sources et bien sûr automatique mais nous préférons les inclure de base pour l'utilisateur car leur génération prend également un certain temps.

À partir de là, il nous suffit de parcourir ces fichiers sources, récupérer son contenu dans un dictionnaire de la structure :

clé : valeur ===> 'periode':[$lien_1, lien_2, \ldots, lien_N$**]** pour ensuite appelé les méthodes d'extraction de contenu.

Comme nous avons par défaut trois genre de contenus dans notre corpus, la structure de ce dernier est de la forme : **période/genre/document.txt** comme suit :

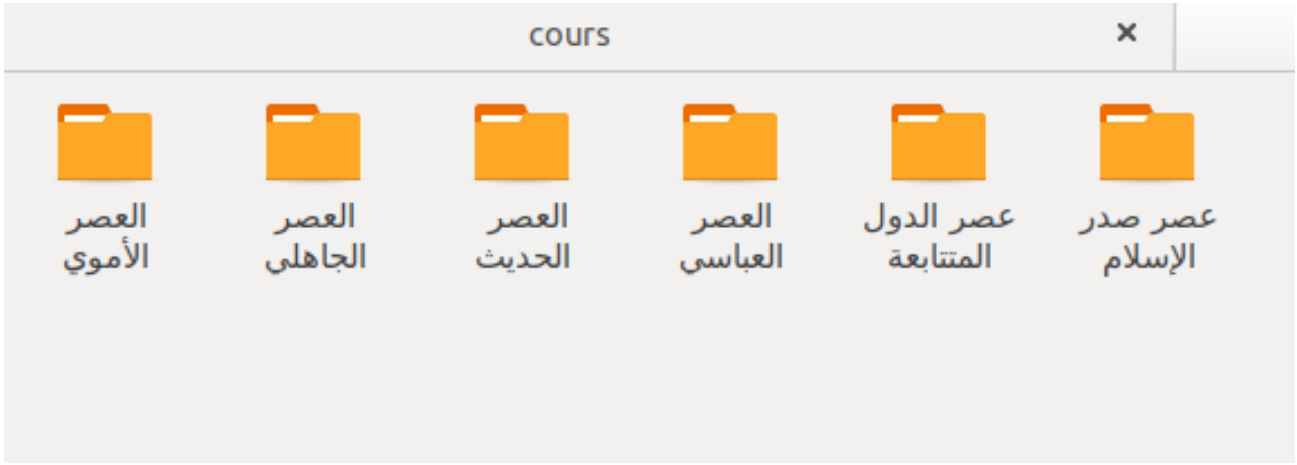


Figure 5.2: Organisation Periode choie dans notre cas

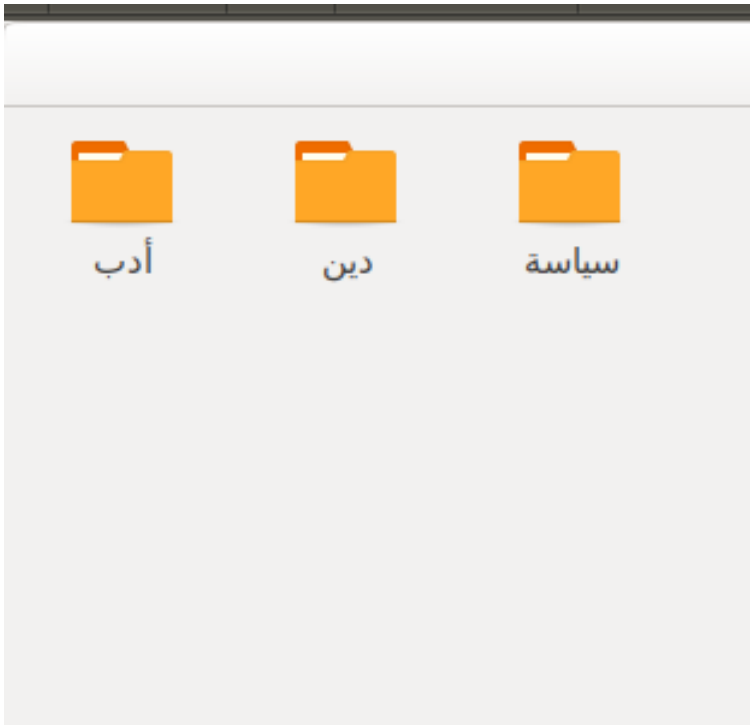


Figure 5.3: Organisation genre choisie dans notre cas

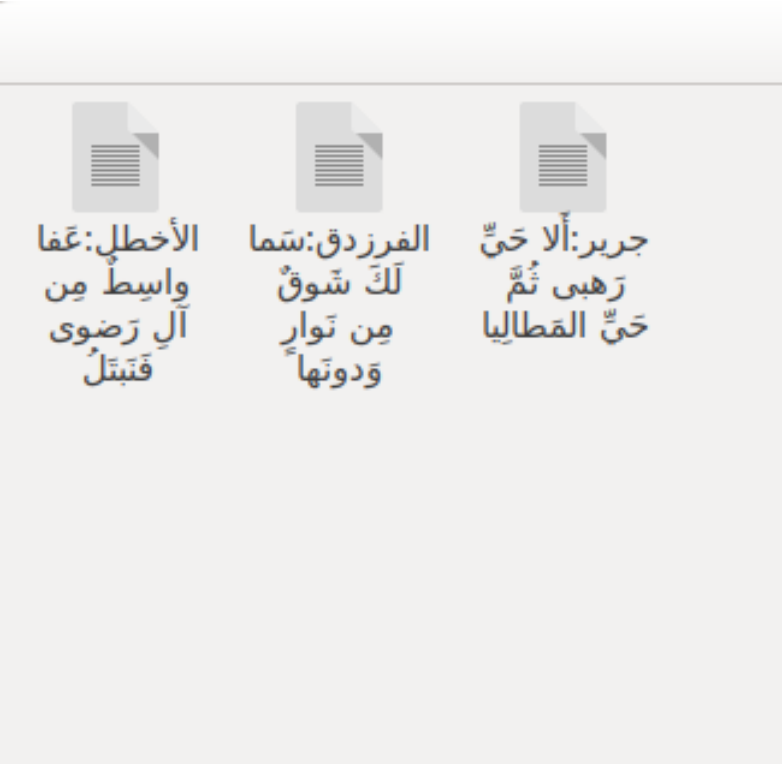


Figure 5.4: les fichiers txt qui sont relatives à un genre et à une periode précise

5.3.2 Corpus au format XML

Pour cette partie nous proposons à l'utilisateur la génération du corpus au format XML soit à partir d'une extraction de contenu sur le web ou bien, directement à partir du corpus doc générée et ce, pour lui permettre de générer un corpus XML à partir de n'importe quel source de documents texte présente localement, il est bien sûr important de mentionner que la génération à partir du corpus au format doc est beaucoup plus rapide que celle à partir du web c'est donc celle que nous recommandons à l'utilisateur.

NB: La structure du corpus xml est la même (période/genre/document.xml)
.

La génération de ce corpus suit les étapes suivantes :

- Parcours du corpus doc pour récupérer les documents texte selon leur période et selon leur genre.
- Lecture des documents, et découpage en phrases comme vu au chapitre précédent.
- Utilisation de la méthode **insert_sent_into_corpus** qui permet d'initialiser un document s'il n'existe pas, d'y ajouter la période, le genre et un ensemble de phrases ce qui nous facilitera l'utilisation des exemples par la suite.

<Corpus doc_name="أقوال محمود درويش:Hikam.txt.xml" genre="أدب" period="العصر الحديث"><Sent num="1">أفضل من حاضرنا
Sent><Sent num="3">يا لهاويتنا كم هي واسعة

Figure 5.5: Structure du corpus en XML

Chapter 6

Stemming et POS

6.1 Stemming

6.1.1 Objective

Dans notre dictionnaire historique nous ne gardons que les stems des mots car le but n'étant pas de créer une base de données au sens classique, nous exploitant alors le stemming pour créer les entrées de notre dictionnaire mais aussi pour la recherche d'un mot dans ce dernier. Ainsi nous évitons le stockage de tous les mots de la langues arabes et réduisant considérablement le temps d'accès aux informations recherchées.

6.1.2 Méthodes utilisés

[jess]

6.1.3 Exemple

[jess]

6.2 POS

6.2.1 Objective

Dans le but de cibler les exemples les plus pertinents pour un mot recherché dans notre dictionnaire historique, nous avons introduit un module d'étiquetage (POS) pour les phrases (exemples) d'utilisation d'un mot donnée, tel que nous affichons en premier lieu les phrases (exemples) contenant le même stem que le mot recherché mais dont l'étiquette (POS) est identique à celle du mot recherché.

6.2.2 Méthodes utilisés

[jess]

6.2.3 Exemple

[jess]

Chapter 7

Exploitation de dictionnaire classique

7.1 Dictionnaire local (mode Offline)

Nous avons téléchargé la base de donné de l'application mobile **Al maany**, celle-ci contenant une table **WordsTable** de 119536 instances et 8 attributs qui sont:

id : identificateur numérique (entier) du mot,
word : le mot concernant l'instance,
explination : la source dont est extrait l'instance,
searchword : variantes du mot de l'instance,
root : stem du mot (valeur manquante),
meaning : différents sens du mot (séparés par le caractère "|"),
searchword_count : nombre de variante du mot données pour ce mot,
word_char_count : taille du mot de l'instance courante.

Nous avons exploité uniquement les attributs **Word** et **Meaning** dans la création/ remplissage de notre dictionnaire historique (XML).
Car les exemples d'utilisation sont automatiquement extrait de notre corpus.

Cette base de donnée ne contenant pas les exemples d'utilisation (phrases) pour enrichir notre dictionnaire, nous avons alors eu recourt à l'accès au dictionnaire en ligne pour en récupérer d'autres exemples d'utilisation.

7.2 Dictionnaire en ligne (mode Online)

Dictionnaire en mode online:

En plus de l'utilisation du dictionnaire en mode offline, nous proposons la recherche de sens des mots et d'exemples d'utilisations en mode online et ce à partir du dictionnaire en ligne **Al Maany**.

Cette recherche se fait de la manière suivante :

- Extraction du contenu de la page web du résultat d'une recherche, cela se fait en concaténant le mot recherché à l'URL **"https://www.almaany.com/ar/dict/ar-ar/"**

- Pour la liste des sens d'un mot donné, nous avons remarqué que les sens se trouvaient entre des balises de la forme **¡li¿ ¡b¿ the word to define ¡/b¿ a set of definitions mostly found between ¡b¿ and ¡/b¿ and ends with ¡/li¿**, nous utilisons donc une expression régulière en conséquent pour récupérer tous les sens possible d'un mode.
- Pour ce qui est des exemples d'utilisation d'un mot, nous remarquons le pattern suivant : **¡div class=c̈ol-md-12 text-right bd-plain¿usew example¡/div¿**
- La sortie de ces deux méthodes sont sous forme d'une liste contenant les sens d'un mot, ou encore des exemples d'utilisation.

Chapter 8

Construction du dictionnaire historique XML

Nous avons opté pour une solution à plusieurs fichiers XML afin de réduire la taille des fichiers et de ce fait réduire le temps d'accès.

Ainsi nous aurons 28 fichiers XML, ce qui correspond à un par lettre alphabétique, cette organisation nous sembla la meilleur solution pour rejoindre l'esprit de l'organisation des dictionnaires classique tout en ayant un nombre raisonnable de fichier XML.

La structure de chacun de nos fichier XML du dictionnaire se présente comme suit:

```
<Dictionary>

  <Entry type="manual" valide="0" supp="0" stem="my_stem">

    <Meanings>
        <Meaning p="1" g="2">meaning de la periode 1 du genre 2</Meaning>
        <Meaning p="2" g="1">meaning de la periode 2 du genre 1</Meaning>
        <Meaning p="3" g="1">meaning de la periode 3 du genre 1</Meaning>
        <!-- suite des sens du mot par genre et période-->
    </Meanings>

    <Examples>
            <Exemple period="1">
                <Sentence genre="1" document-name="text1" position="5" pos="VBZ"></Sentence>
                <Sentence genre="1" document-name="text2" position="11" pos="ADJ"></Sentence>
                <Sentence genre="2" document-name="text3" position="1" pos="NN"></Sentence>
                <!--suite des phrases de la periode 1-->
            </Exemple>

            <Exemple period="2">
                <Sentence genre="3" document-name="text12" position="17" pos="VBZ"></Sentence>
                <Sentence genre="1" document-name="text2" position="25" pos="NN"></Sentence>
                <!--suite des phrases de la periode 2-->
            </Exemple>
            <!-- suite des exemples des autres periodes -->
    </Examples>
  </Entry>
  <!-- suite des entrées -->
</Dictionary>

```

Figure 8.1: squelette d'un fichier XML de notre dictionnaire historique.

Chaque balise **Entry** (entrée) de notre dictionnaire est représentée par :

Stem : un stem,
Valide : un degré de validation (de 0 à 3 qui correspond au nombre de lexicographe ayant confirmé l'entrée),
Supp : le nombre de lexicographes ayant supprimé une entrée, tel que ce n'est que lorsque les 3 lexicographes supprime une entrée qu'elle est réellement supprimé du dictionnaire,
Type : la méthode d'ajout qui peux être manuelle ou automatique.

Elle possède aussi :
Une balise **Meanings** qui contient l'ensemble des balises **Meaning** cette dernière étant caractérisé par :
p : la période ou le mot avez le sens contenu dans cette balise,
g : le genre dans le quel le mot avez le sens contenu dans cette balise,

Une balise **Examples** qui contient l'ensemble des balises **Example** cette dernière étant caractérisé par :
period : la période d'utilisation du mot courant dans les phrases référencées par les balises **Sentence** qu'il comporte.

Enfin chaque balise **Sentence** possède les attributs suivants:

genre : le genre du texte à partir du quelle la phrase est extraite,
document-name : le nom du document au quel la phrase appartient,
position : le numéro de la ligne de la phrase dans le document,
pos : (Part-Of-Speech) étiquete du mot au quel on fait référence dans cette phrase.

Chapter 9

Statistique et mesure de d'évaluation de la performance du système

9.1 Statistiques

9.2 Mesure de d'évaluation de la performance du système

Soient les trois opérations élémentaires : **Insertion, suppression** et **modification** d'une entrée dans notre dictionnaire historique.

Nous définissons des couts de chacune des opérations précédentes comme suit:
Insertion $\Rightarrow$ 1
Suppression $\Rightarrow$ 1
Modification $\Rightarrow$ 0.5

Nous avons pensé à évaluer notre système en utilisant le rappel et la précision de manière à les adapter au contexte de notre système, ainsi nous définissant les formules de ces deux mesures par:

$$Rappel' = \frac{\text{nombre d'entrée à la création}}{\text{nombre d'entrée après les insertions du lexicographe}}$$

$$Précision' = \frac{\text{nombre d'entrée après les suppressions et modifications}}{\text{nombre d'entrée existantes à la création}}$$

à travers ces deux mesures nous sommes capable d'évaluer notre système en utilisant par exemple, le **F-mesure** avec le poids β (avec le poids car nous jugeons que la précision doit avoir un poids supérieur à celui du rappel car nous avons conscience que notre dictionnaire à besoin d'être enrichi, ce qui fait partie du but de l'application).

$$F_\beta = \frac{(1+\beta^2)\cdot(\text{Précision'}\cdot\text{Rappel'})}{(\beta^2\cdot\text{Précision'}+\text{Rappel'})}$$

Avec:
β : le poids de la précision.
Nous avons pris $\beta = 2$

Bien évidemment plus cette mesures se rapproche du 100% plus nous confirmerons l'efficacité et la précision de notre système.

Chapter 10

Outils de développement

blabla pour backend python avec pycharm et compgnie
pour frontend pyQt
liste des jar et packages utilisé : pyarabic, googletrans, ...

Chapter 11

Conclusion

Manuel d'utilisation

[jesssss mdrr]