République Algérienne Démocratique et populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université des Sciences et de la Technologie Houari Boumediene

Faculté d'Electronique et Informatique

Département Informatique

Master spécialité : SII
Module : Traitement automatique du langage naturel

Intitulé du projet :

Implémentation d'un générateur automatique de grammaire Pour la langue française

Réalisé par :

- SLIMI Zakaria 161634019909

Année universitaire : 2020/2021

Contents

1. Introduction :

Le langage naturel est la langue « normale courante », et elle s'avère être le moyen de communication le plus répandu chez l'être humain. Ce qui suscite chez les chercheurs qui s'intéressent au comportement humain, un intérêt assez particulier pour cette propriété humaine. L'intelligence artificielle aspire à automatiser son traitement et de la manière la plus précise possible afin de faciliter l'interaction homme-machine. Un des buts ultimes de ce domaine est d'arriver à concrétiser le dialogue homme-machine. Cela depuis 1950, au temps de Turing déjà la possibilité d'un tel exploit était envisageable dans un futur proche. D'ailleurs, nous pouvons en témoigner à l'instant présent. Les avancées technologiques ont permis d'aboutir et de réaliser beaucoup de points qui auraient semblés très lointains pour d'autres époques. On s'intéresse aussi, dans ce domaine, à l'analyse du langage humain dans des textes précieux, historiques, ou autres et qui pourraient contenir des trésors bien invisible à l'observation humaine basique. Nous allons nous intéresser dans ce projet, d'analyser la grammaire de la langue Française.

1.1. Définition de TAL :

Un traitement automatique est une suite d'actions ou calculs à faire effectuer par la machine Le Traitement Automatique des Langues a pour objectif de traiter des données linguistiques (textes) exprimées dans une langue dite "naturelle" [Delafosse, 1999].

1.2. Le fonctionnement d'système de TALN :

- ✓ Analyse morpho-lexical :
- ✓ Analyse syntaxique :
- ✓ Analyse sémantique

2. Problématique :

La française et une langue parler par 300 millions de personne, Une des caractéristiques de la grammaire française vis-à-vis de nombreuses langues vivantes est la richesse de ses temps et modes

Pour cela, il est demandé de développer une application qui pourrait offrir certaines fonctionnalités de base afin de pouvoir aboutir à cette analyse symétrique. Elle permettrait, donc, de faire plusieurs choses sur la grammaire

3. Objectif :

- ✓ Introduire le nom d'un répertoire (contenant un corpus) ou d'un texte pour l'analyse syntaxique.
- ✓ Afficher le texte à analyser syntaxiquement ou donner à l'utilisateur de sélectionner un texte à afficher et à analyser ;
- ✓ Afficher le texte analysé syntaxiquement (ou n'importe quel texte du corpus analysé) chaque phrase sur une ligne (ou plusieurs lignes) suivie par son analyse syntaxique sur une autre ligne (ou plusieurs lignes)
- ✓ Permettre à l'utilisateur d'introduire un nouveau corpus d'apprentissage pour la régénération de la grammaire.

4. Description de la corpus:

On a six collections de corpus, **Free French Treebank master.zip**, une collection de textes Étiquetés. Les autres collections ne sont pas étiquetées. On utilise pour exploitent d'autre phrase.

5. Implémentation d'un algorithme d'induction de grammaire propositionnelle :

~~Il s'agit d'implémenter l'algorithme suggéré par Selab & Guessoum pour l'induction d'une~~ grammaire Propositionnelle. Cet algorithme se base sur les statistiques il calcule les fréquences des cooccurrences de n-grammes des Pos-tags. Chaque fois on prend le n-gram de POS-tags dont la fréquence est la plus élevée, une nouvelle règle grammaticale est créée pour ce n-gram. Le processus est répété itérativement jusqu'à ce que toute la grammaire soit produite.

mot	fréquence	prob	r*prob
de	129206	0.53	0.53
,	124776	051	1.02
.	83440	0.49	1.47
la	71592	0.46	1.84
le	50144	0.45	2.25
et	37594	0.39	2.34
les	34493	0.36	2.52
des	33967	0.33	2.64

Modélisation Avec Zifs Law

Pseudo algorithme

Input: POS-tagged English sentences (corpus)

Output: Induced English grammar

Begin

Data_File = set of POS-tagged sentences

Repeat:

```
    a = N_Gram_extraction(Data_File)
    F = Frequency_distribution(a)
    R = N_Gram[Max(F)]
    Substitution(Data_File, R, Nonterminal)
Until (no mor rules can be extracted)
End
```

5.1. Detaille sur l'algorithme

1. Initialisation :

Normalisation du texte pris en entrée en phrase puis le stocker dans une liste. Le format de cette liste est une liste de listes chaque liste présente une phrase et chaque phrase est transformée en tuples.

Chaque tuple représente le mot et son POS-tag.

2. Extraction des n-grams :

Pour chaque phrase extraire tous les n-grammes allant de 2 jusqu'à n avec n la taille de la phrase. Les n-grams sont uniquement constitué des POS-tags sans tenir compte des mots. Construire une liste de tous les n-grammes extraits. Trier cette liste selon le type du n-gramme ; bigrams, trigrams, etc.…

Chaque type de n-grams est stocké dans une sous-liste.

3. Distribution de fréquences :

Calculer la fréquence de chaque n-grams extrait et trier les n-grams du plus fréquent au moins fréquent.

4. Extraction de règles :

Construction des règles en se basant sur la fréquence des n-grams. Pour chaque règle extraite son côté gauche est un symbole non-terminal, et son côté droit est le n-gramme qui a la distribution de fréquence maximale.

5.2. Application de la grammaire :

1.Nettoyage du paragraphe :

Pour pouvoir appliquer la grammaire sur un fichier de l'un des deux corpus proposés. On doit d'abord procéder par le nettoyage du fichier. D'abord normaliser le texte en phrase, pour chaque mot ne garder que son POS-tag.

2. Application de la grammaire :

Application de la grammaire sur le paragraphe nettoyé. Pour chaque phrase qui a pu être analysée syntaxiquement ; écrire la phrase suivie du résultat de son analyse syntaxique. Les phrases qui n'ont pas puent être analysées syntaxiquement sont écrite dans un fichier texte.

6. Implémentation :

6.1. Hardware :

PC Portable :

CPU : I3 4em Génération

Ram : 8 G

6.2. Software :

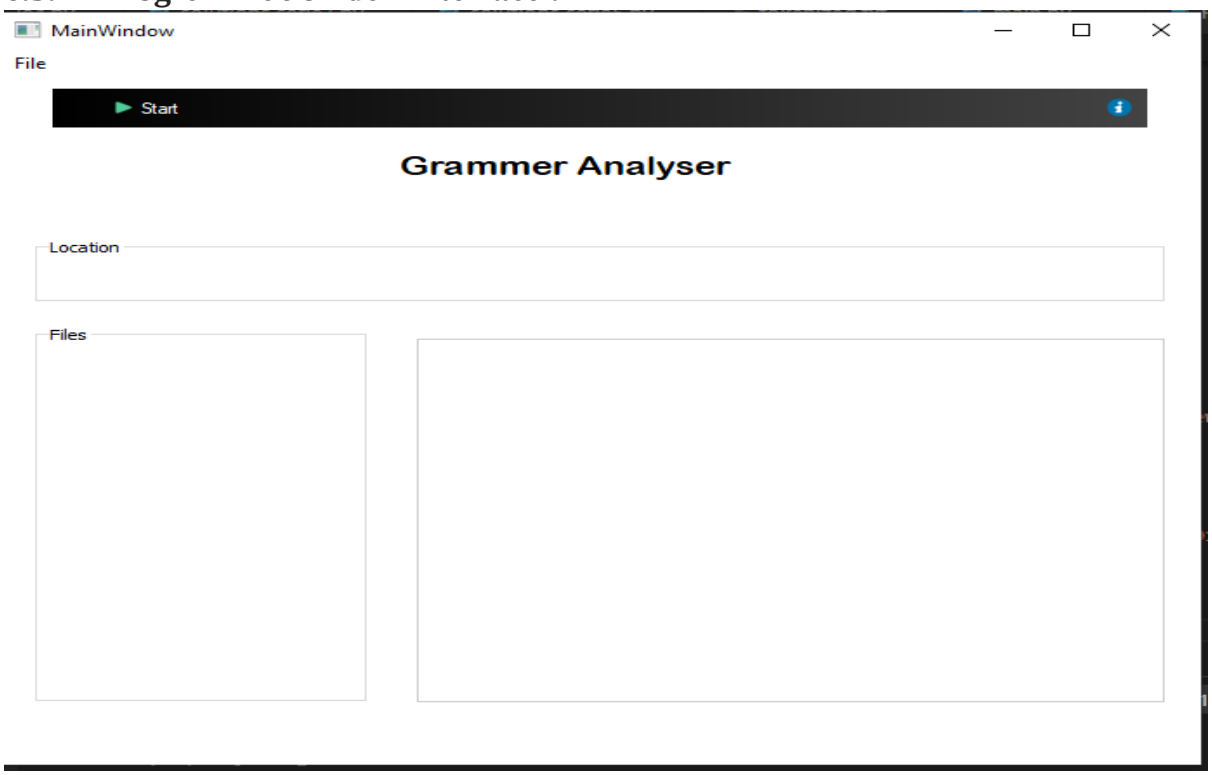
On utiliser :

Python comme Langage

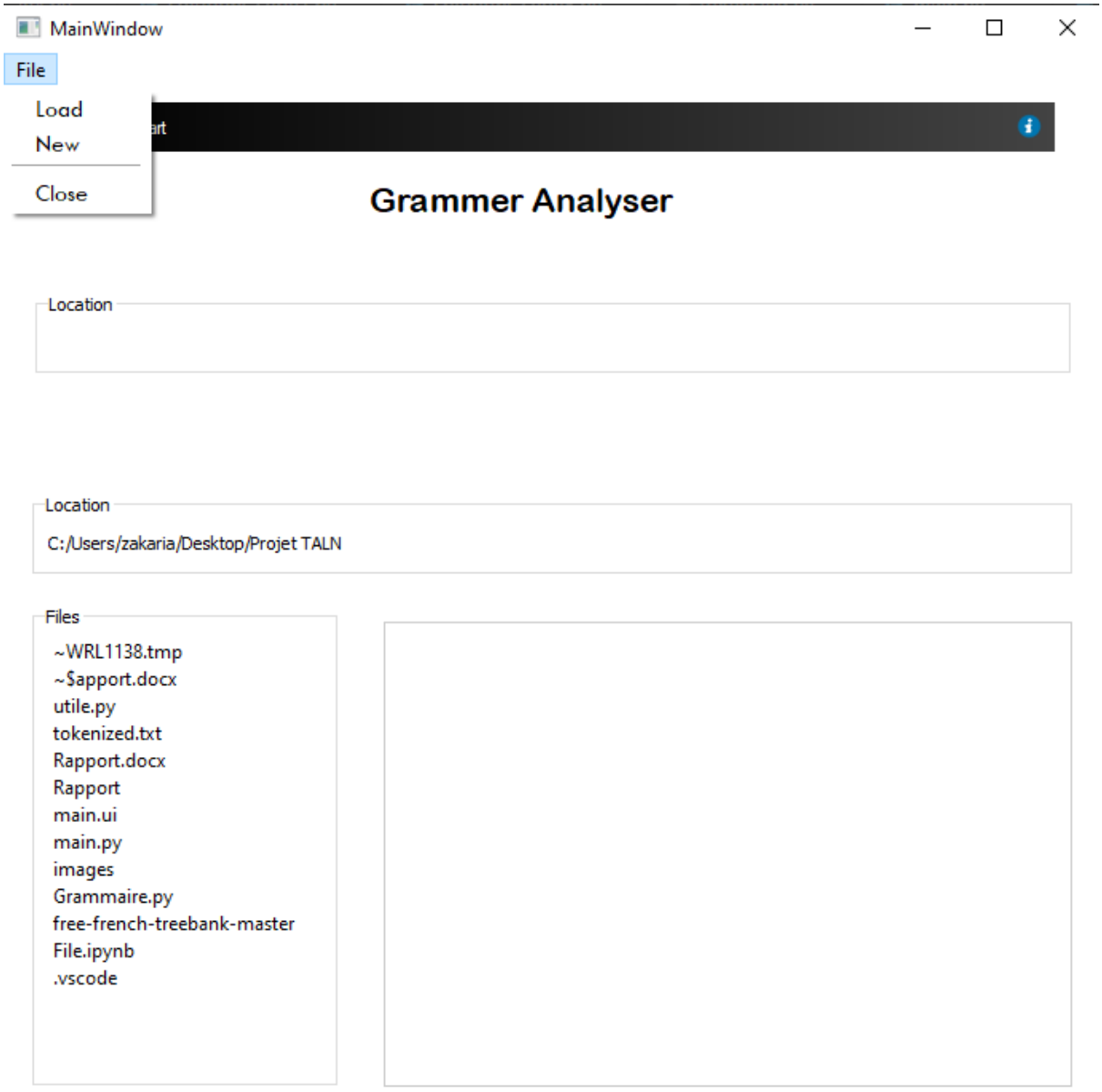
Vscode comme editor avec l'extension Jupyter

Qt Designer Pour Le Gui

6.3. Programmation de l'interface :



On peut ajouter le directory que contient un ou plusieurs fichiers texte



Les étape d algorithme :

dans se partie en extraire tout les POS-TAGS :

```python
#Utile Function
def splitbyspace(l):
    l= str(l)
    l = l.replace('\n','')
    l=l.split(" ")
    a=[]
    for i in l:
        tag = i.split('_')
        word = tag[1]
        tag = tag[0]
        a.append([tag,word])
    return a
def save_Tokens():
    a =[]
    global TokenList
    with open('tokenized.txt','w+',encoding='utf-8') as out:
        for line in f:
            a+=splitbyspace(line)
        out.write(str(a))
        out.close()
    TokenList = a
#Load File
f = open('free-french-treebank-master\\130612\\frwikinews\\txt-tok-pos\\frwikinews-20130110-pages-articles.
TokenList =[]
```

```
['mousse', 'NC'], ['.', 'PONCT'], ['Mais', 'CC'], ["aujourd'hui", 'ADV'], [',', 'PONCT'], ['Bill', 'NPP'], ['Parsons', 'NPP'], ['a', 'V'],
['déclaré', 'VPP'], [':', 'PONCT'], ['«', 'NC'], ['nous', 'CLS'], ['avons', 'V'], ['eu', 'VPP'], ['tort', 'NC'], ['»', 'ADJ'], ['.', 'PONC
'], ['Et', 'CC'], ["c'", 'CLS'], ['est', 'V'], ['un', 'DET'], ['coup', 'NC'], ['dur', 'ADJ'], ['pour', 'P'], ['la', 'DET'], ['NASA', 'NPP'
, ['.', 'PONCT'], ['Car', 'CC'], ['le', 'DET'], ['Congrès', 'NC'], ['des', 'P+D'], ['États-Unis', 'NPP'], ['a', 'V'], ['toujours', 'ADV'],
['été', 'VPP'], ['plus', 'ADV'], ['ou', 'U'], ['moins', 'ADV'], ['hostile', 'ADJ'], ['au', 'P+D'], ['programme', 'NC'], ['navette', 'NC'],
['spatiale', 'ADJ'], [',', 'PONCT'], ['qui', 'PROREL'], ['coûte', 'V'], ['extrêmement', 'ADV'], ['cher', 'ADJ'], ['à', 'P'], ['maintenir',
'VINF'], ['.', 'PONCT'], ['Entailles', 'NC'], ['sur', 'P'], ['les', 'DET'], ['tuiles', 'NC'], ['en', 'P'], ['céramique', 'NC'], ['Grâce',
U'], ['à', 'P'], ["l'", 'DET'], ['important', 'ADJ'], ['dispositif', 'NC'], ['de', 'P'], ['surveillance', 'NC'], ['mis', 'VPP'], ['en', 'P
], ['place', 'NC'], ['par', 'P'], ['la', 'DET'], ['NASA', 'NPP'], ['pour', 'P'], ['le', 'DET'], ['décollage', 'NC'], ['de', 'P'], ['Discov
ery', 'NPP'], [',', 'PONCT'], ["l'", 'DET'], ['agence', 'NC'], ['spatiale', 'ADJ'], ['a', 'V'], ['pu', 'VPP'], ['détecter', 'VINF'], ['deux
, 'DET'], ['entailles', 'NC'], ['sur', 'P'], ['les', 'DET'], ['tuiles', 'NC'], ['en', 'P'], ['céramique', 'NC'], ['constituant', 'VPR'], [
le', 'DET'], ['bouclier', 'U'], ['thermique', 'U'], ['de', 'P'], ['la', 'DET'], ['navette', 'NC'], ['.', 'PONCT'], ['Il', 'CLS'], ['semble
ait', 'V'], ['que', 'CS'], ['ces', 'DET'], ['entailles', 'NC'], ['ne', 'ADV'], ['remettent', 'V'], ['pas', 'ADV'], ['en', 'P'], ['cause',
NC'], ['la', 'DET'], ['sureté', 'NC'], ['de', 'P'], ['la', 'DET'], ['navette', 'NC'], ['et', 'CC'], ['à', 'P'], ['ce', 'DET'], ['sujet', '
C'], [',', 'PONCT'], ['le', 'DET'], ['directeur', 'NC'], ['adjoint', 'ADJ'], ['du', 'P+D'], ['programme', 'NC'], ['des', 'P+D'], ['navette
```

Extraire To les N-Grames est sauvegarder dans un fichier .cfg

```python
#extraireAll Gram
def get_Bgram():
    global f
    files = f.read()
    files = files.replace(',','.')
    files = files.split(' ')
    with open('gram.cfg',"w+") as gr :
        u = 0
        for i in files :
            rs = extraireRule(i)
            gr.write('S'+str(u)+"->"+rs)
            u+=1

def extraireRule(text):
    global dc
    text = text.split(' ')
    gram = ""
    for i in text:
        gram += dc[i]
    return gram
#Load File
```

7. Conclusion :

- Les expressions régulières étaient d'une très grande utilité.
- Nous avons manipulé les structures de données que propose python.
- Application en domaine réel le TALN et voir les résultats