



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'INFORMATIQUE

RAPPORT DE PROJET TP ALGO (C)

Elaboré par :

- **BOUZOURINE Hichem**
- **BEN BORDI Taki eddine**

Section : ISILB

Groupe : 2

Table des matières :

Introduction

1. Problématique:
 - 1.1 Explication
 - 1.2 Solution
 2. Architecture du programme:
 - 2.1 Les fichiers
 - 2.1.1 Source
 - 2.1.2 Header
 - 2.2 les Schémas
 - 2.2.1 Schema de listeMots
 - 2.2.2 Schema de listeFreq
 - 2.2.3 Schema de dictContext
 - 2.3 Fonctions et leur utilisation avec **Complexité**
 - 2.4 Implémentation des différentes opérations
-

Introduction

Ce document décrit le travail réalisé dans le cadre du projet académique du parcours "*Licence Ingénierie des Systèmes d'Information et des Logiciels*".

Dans le but de concrétiser et améliorer nos connaissances pratiques et théoriques récoltées durant nos deux années universitaires par le développement d'un projet qui résout une problématique du monde réel.

C'est ainsi que nous avons procédé à la mise en œuvre du Programme « **Dictionnaire de fréquences des mots et ses applications** ». Ce travail s'articule principalement sur deux axes
Problématique et l'architecture du programme

Pour ce qui est de l'environnement de travail et de développement, le projet a été codé sur **LINUX Ubuntu 20.04**

L'interface est le **Terminal**.

La source des mots est le texte "Jane_Austen_Emma" qui a été envoyé par le professeur.

Chapitre 1: Problématique

Explication:

Dans les études de langues, nous sommes intéressés à étudier les contextes dans lesquels un mot est utilisé, mais certaines mesures doivent être prises à l'avance.

Alors, comment pouvons-nous traduire ce travail dans l'ordinateur pour avoir une bonne solution ?

C'est ce que nous allons voir ensuite.

Chapitre 1: Problématique

Solution:

L'utilisateur doit fournir un texte au format « .txt » donc nous devons le convertir en tout ce que l'ordinateur peut comprendre, puis nous devons créer une liste fréquente avec d'autres options internes, et tout cela pour avoir la possibilité de créer un dictionnaire flexible à lire.

On propose la solution suivante :

Sur la base de la structure de données (LL), nous allons créer 3 listes de chaînes listeMots, listeFreq et dictContext qui va manipuler le travail

Chapitre 2:Architecture du programme:

Fichiers Source :

liste_mots.c
liste_freq.c
dict_contexte.c
main.c

Fichiers headers :

liste_mots.h
liste_freq.h
dict_contexte.h

Selon la **programmation modulaire**, les fichiers **headers** contient que la **structure** de liste chaînée les **prototypes** des fonction appelées dans les fichiers **Source**

Schéma de listMots:

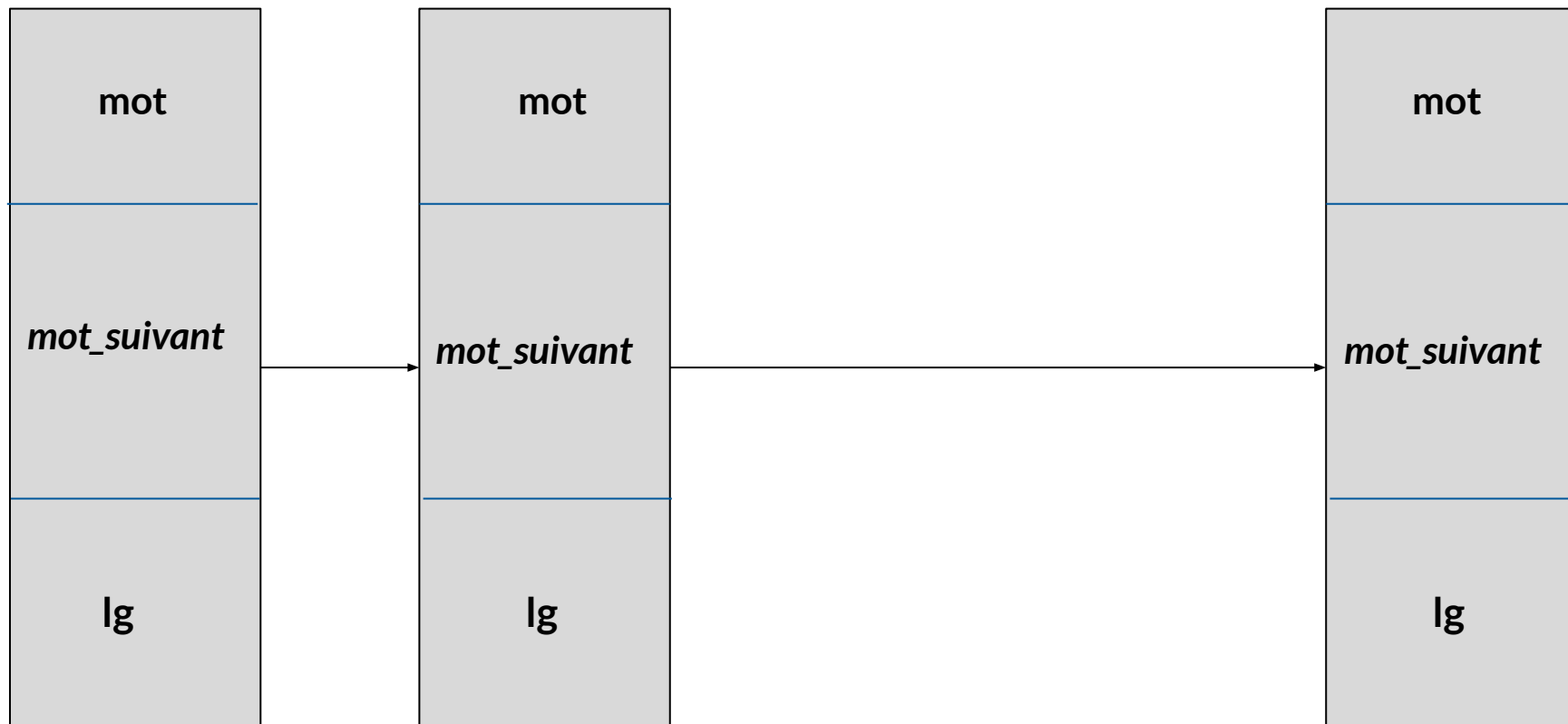


Schéma de listeFreq:

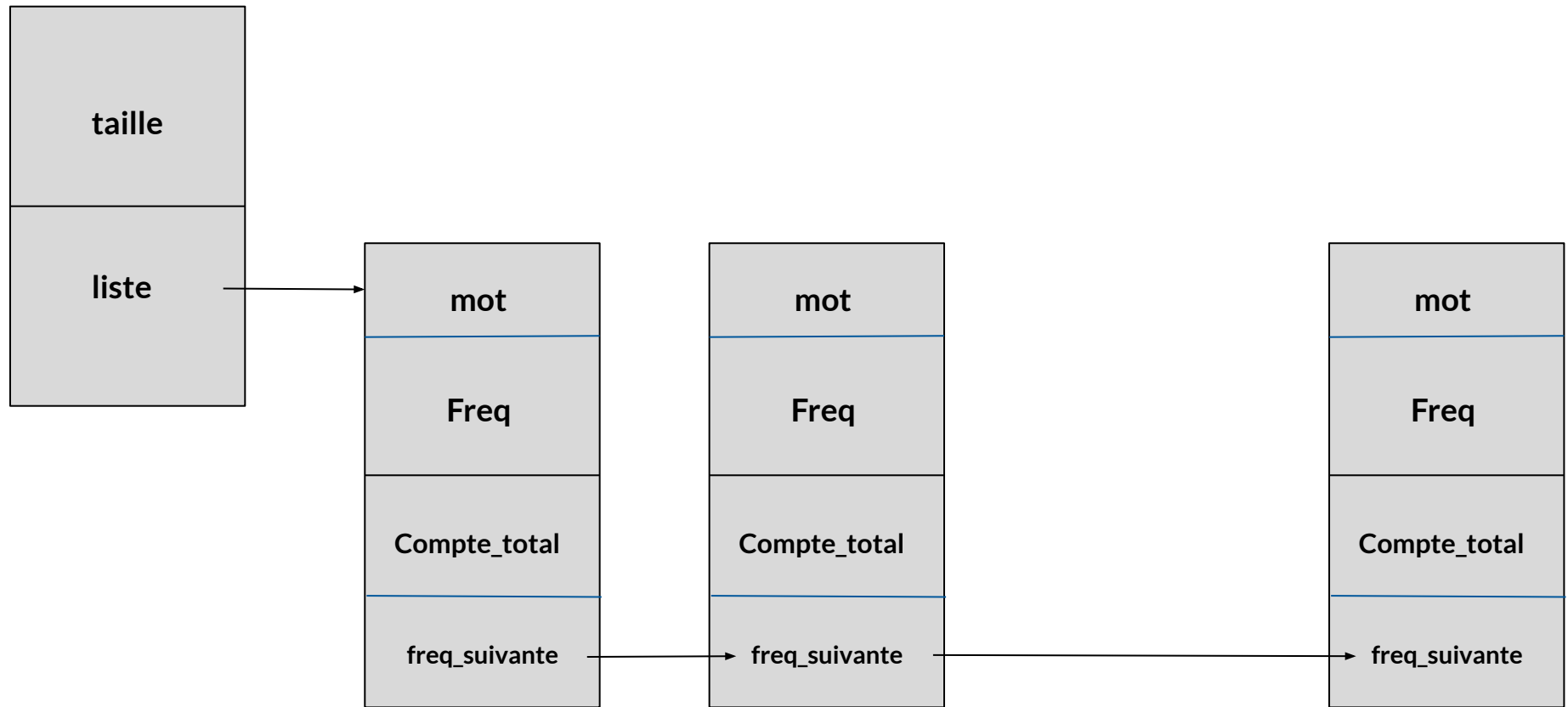
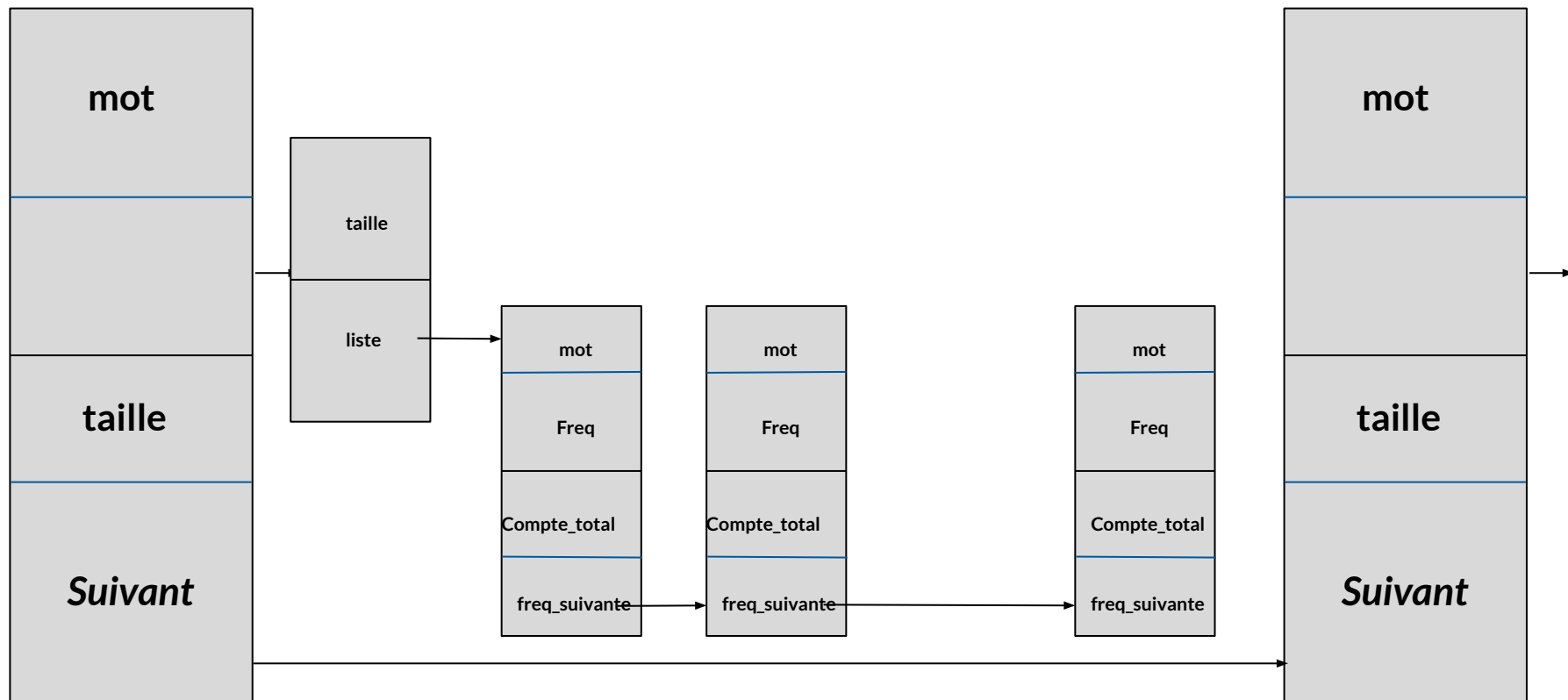


Schéma de dictContext :



Fonction pour détecter l'erreur :

```
void raler(const char * message) {  
    fprintf(stderr, "%s\n", message);  
    exit(EXIT_FAILURE);  
}
```

← Complexité: O(1)

- Cette fonction **arrête** le programme et affiche le **message** de l'erreur.
 - Cette fonction est **globale**, elle est utilisable n'importe où.
 - Cette fonction vous aidera en plusieurs fois à détecter l'erreur et à faciliter le **débogage (Debug)**.
-

Fonctions utilisées dans listeMots.C:

```
listeMots motsDe(char* t);  
int motEstDansListeMots(listeMots liste, const char *mot);  
void afficher_liste(listeMots liste);  
void lower_mot(char* mot);  
void detruit_liste_mots(listeMots liste);
```

Explication de certaines fonctions:

Function (motsDe) :

```
listeMots motsDe(char* t) { ...  
    ...  
    ...  
}
```

← Complexité: $O(n)$

- Cette fonction permet de créer la liste des mots d'une chaîne de caractère t.
 - Il faut mettre dans la tête que les caractères "spéciaux" devant compter comme des mots à part,
-

Explication de certaines fonctions:

Function (motEstDansListeMots) :

Complexité: $O(n)$

```
int motEstDansListeMots(listeMots liste, const char * mot) {  
    ...  
    ...  
}
```

-Cette fonction retourne vrai si le mot passé en entrée est dans la liste passée en entrée, faux sinon

cette fonction nous aidera dans le **Dictionnaire de contextes**

Explication de certaines fonctions:

Function (lower_mot) :

```
void lower_mot(char* mot) {  
    ...  
    ...  
}
```

Complexité: $O(n)$



-Cette fonction permet de rendre minuscule le mot passé en entrée.

Nous allons standardiser notre façon de travailler, tous les mots doivent être en **minuscules**.

Fonctions utilisées dans listeFreq.c:

```
listeFreq frequencesDe(char* t);  
_listeFreq trouverMot(_listeFreq liste, const char* mot);  
int motEstDansListeFreq(_listeFreq liste, const char* mot);  
void ajouterFreq(listeFreq* liste_freq, _listeFreq nouv_freq);  
void detruit_liste_freq(_listeFreq liste);  
listeFreq contexte(char* t, char* m);  
void afficher_liste_freq(listeFreq liste);
```

Explication de certaines fonctions:

Function (frequencesDe) :

```
listeFreq frequencesDe(char* t) {  
    ...  
    ...  
}
```

Complexité: $O(n)$



-Cette fonction créer la liste des fréquences pour la chaîne de caractère t donnée en entrée <le texte donnée>.

Remarque : il n'y a pas de répétition de mots dans la liste !

Explication de certaines fonctions:

Function (frequencesDe) :

Complexité: $O(n)$



```
listeFreq contexte(char* t, char* m) {  
    ...  
    ...  
}
```

-Cette fonction créer la liste des contextes du mot m, par ordre croissant

Remarque : le contexte est d'ordre **1**.

Explication de certaines fonctions:

Function (motEstDansListeFreq) :

Complexité: $O(n)$



```
int motEstDansListeFreq(listeMots liste, const char * mot) {  
    ...  
    ...  
}
```

- Cette fonction retourne vrai si le mot donnée en entrée est dans la liste des fréquences donnée en entrée

Fonctions utilisées dans dictContext.c:

```
dictContexte init_dictContexte(char* t);  
void afficher_dict(dictContexte dict);  
char* maxContexte(dictContexte d, char* m);  
dictContexte get_contexte(dictContexte d, char* m);  
int freqCont(dictContexte d, char* m, char* n);  
void genererTexte(dictContexte d, int n);
```

Explication de certaines fonctions:

Function (init_dictContexte) :

Complexité: $O(n)$



```
dictContexte init_dictContexte(char* t) {  
    ...  
    ...  
}
```

- Cette fonction créer le dictionnaire des contexte de la chaîne de caractère t passée en entrée <texte donnée>

Explication de certaines fonctions:

Function (get_contexte) :

Complexité: $O(n)$



```
dictContexte get_contexte(dictContexte d, char* m) {  
    ...  
    ...  
}
```

- Cette fonction renvoie la liste des contextes du mot passé en entrée, dans le dictionnaire d


Remarque : on part du principe que le mot à bel et bien on contexte dans d

Explication de certaines fonctions:

Function (maxContexte) :

Complexité: $O(1)$

```
char* maxContexte(dictContexte d, char* m) {  
    ...  
    ...  
}
```



Cette fonction renvoie le mot le plus présent dans le dictionnaire des contexte du mot m

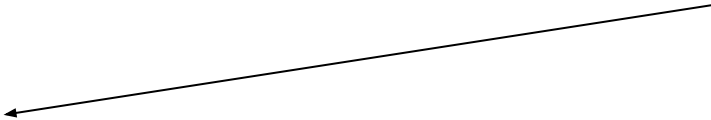
Remarque : on part du principe que le mot à bel et bien on contexte dans d

Explication de certaines fonctions:

Function (freqCont) :

Complexité: $O(n)$

```
int freqCont(dictContexte d, char* m, char* n) {  
    ...  
    ...  
}
```



Cette fonction renvoie le nombre de fois qu'apparaît de mot
n dans le dictionnaire des contextes de m.

Explication de certaines fonctions:

Function (genereTexte) :

Complexité: $O(\text{taille}(d)) = O(n')$

```
void genererTexte(dictContexte d, int n) {  
    ...  
    ...  
}
```

Cette fonction affiche la séquence de **n** mots ou le mot à la position **i** est le contexte le plus fréquent du mot à la position **i-1**.

Remarque : on suppose que **n** < **taille** du dictionnaire

main.c (menu de programme) :

```
char* chargerFichier(const char* path);
```

```
int main(void);
```

Explication de certaines fonctions:

Function (chargerFichier) :

Complexité: $O(n)$

```
char* chargerFichier(const char* path) {  
    ...  
    ...  
}
```

- Cette fonction reçoit l'emplacement du fichier et elle va charger le contenu du fichier dans la variable **texte**.

- Si le path est **invalide** ou le fichier est **vide** elle va appeler la fonction globale **raler()**;

Explication de certaines fonctions:

Function (main) :

```
int main(void) {  
    ...  
    ...  
}
```

-Cette fonction affiche à l'utilisateur les différentes opérations possibles telles que : liste des mots, liste de fréquence de chaque mot, liste des contextes, le maxContexte d'un mot, la fréquence d'apparition d'un mot dans le contexte d'un autre, génération du texte, changement de chemin ou Quitter le programme.

Le choix est assuré par **Switch(choix)**

Implémentation des différentes opérations :

Chargement du fichier et choix de l'opération :

```
~/dict_freq_mots$ make run
```

Quel est le chemin vers votre fichier texte ?

t2.txt

Quelle opération voulez-vous effectuer ?

- 1) Afficher la liste des mots
- 2) Afficher la liste de fréquences (ordre décroissant)
- 3) Afficher la liste des contextes d'ordre 1 d'un mot de votre choix
- 4) Afficher le mot le plus fréquent d'un contexte du mot de votre choix
- 5) Afficher le nombre d'apparition d'un mot dans le contexte d'un autre
- 6) Afficher le texte généré
- 7) Changer de fichier texte
- 8) Quitter

Entrez votre choix: █

1)Affichage de la liste des mots.

```
4759: -
4760: enough
4761: to
4762: marry
4763: a
4764: woman
4765: as
4766: portionless
4767: even
4768: as
4769: miss
4770: taylor
4771: ,
4772: and
4773: to
```

Implémentation des différentes opérations :

2) Affichage de la liste de fréquences (ordre décroissant):

- 1) Afficher la liste des mots
- 2) Afficher la liste de fréquences (ordre décroissant)
- 3) Afficher la liste des contexte d'ordre 1 d'un mot de votre choix
- 4) Afficher le mot le plus fréquent d'un contexte du mot de votre choix
- 5) Afficher le nombre d'apparition d'un mot dans le contexte d'un autre
- 6) Afficher le texte généré
- 7) Changer de fichier texte
- 8) Quitter

Entrez votre choix: 2

```
,: 303
.: 197
and: 143
to: 126
of: 123
the: 115
a: 113
-: 95
her: 75
i: 57
in: 56
it: 52
;: 52
was: 51
you: 50
had: 50
": 44
for: 44
she: 42
be: 40
not: 39
mr: 39
have: 37
but: 35
that: 33
he: 31
```

3) Affichage de la liste des contexte d'ordre 1 d'un mot Pa exemple "Emma"

Quelle opération voulez-vous effectuer ?

- 1) Afficher la liste des mots
- 2) Afficher la liste de fréquences (ordre décroissant)
- 3) Afficher la liste des contexte d'ordre 1 d'un mot de votre choix
- 4) Afficher le mot le plus fréquent d'un contexte du mot de votre choix
- 5) Afficher le nombre d'apparition d'un mot dans le contexte d'un autre
- 6) Afficher le texte généré
- 7) Changer de fichier texte
- 8) Quitter

Entrez votre choix: 3

Choisissez un mot

Emma

Le mot

```
,: 3
woodhouse: 2
turned: 1
bears: 1
herself: 1
playfully: 1
;: 1
spared: 1
smiled: 1
could: 1
was: 1
first: 1
': 1
doing: 1
.: 1
by: 1
should: 1
```

Implémentation des différentes opérations :

4) Affichage le mot le plus fréquent d'un contexte du mot Par exemple "Emma"

Quelle opération voulez-vous effectuer ?

- 1) Afficher la liste des mots
- 2) Afficher la liste de fréquences (ordre décroissant)
- 3) Afficher la liste des contexte d'ordre 1 d'un mot de votre choix
- 4) Afficher le mot le plus fréquent d'un contexte du mot de votre choix
- 5) Afficher le nombre d'apparition d'un mot dans le contexte d'un autre
- 6) Afficher le texte généré
- 7) Changer de fichier texte
- 8) Quitter

Entrez votre choix: 4

Chossissez un mot

Emma

Le mot le plus fréquent du contexte de "emma" est ",",

5) Affichage le nombre d'apparition d'un mot dans le contexte d'un autre Par exemple "Emma", "woodhouse"

Quelle opération voulez-vous effectuer ?

- 1) Afficher la liste des mots
- 2) Afficher la liste de fréquences (ordre décroissant)
- 3) Afficher la liste des contexte d'ordre 1 d'un mot de votre choix
- 4) Afficher le mot le plus fréquent d'un contexte du mot de votre choix
- 5) Afficher le nombre d'apparition d'un mot dans le contexte d'un autre
- 6) Afficher le texte généré
- 7) Changer de fichier texte
- 8) Quitter

Entrez votre choix: 5

Chossissez un mot

Emma

Chossissez un second mot

woodhouse

"woodhouse" apparait 2 fois dans le contexte de "emma"

Implémentation des différentes opérations :

6) Affichage du texte généré:

Quelle opération voulez-vous effectuer ?

- 1) Afficher la liste des mots
- 2) Afficher la liste de fréquences (ordre décroissant)
- 3) Afficher la liste des contextes d'ordre 1 d'un mot de votre choix
- 4) Afficher le mot le plus fréquent d'un contexte du mot de votre choix
- 5) Afficher le nombre d'apparition d'un mot dans le contexte d'un autre
- 6) Afficher le texte généré
- 7) Changer de fichier texte
- 8) Quitter

Entrez votre choix: 6

Entrez le nombre de mot que vous souhaitez

10

Création du dictionnaire de contexte..

her father , and the match , and the match

Conclusion :

Le programme a été conçu dans le but de concevoir plusieurs opérations à partir de texte .

La réalisation de ce projet nous a permis en terme pédagogique d'avoir une approche concrète de la façon dont est conçu un programme qui aidera l'étude des langages en passant par les différentes étapes de l'analyse des besoins jusqu'à la réalisation du code nous permettant ainsi, de pratiquer toutes les notions de **programmation modulaire** acquise tout au long du semestre ainsi qu'une initiation aux **LINUX** afin de prendre l'habitude de travailler comme des professionnels .

Merci.
