L2 - PROMOTION 2026

eFrei
PARIS PANTHÉON-ASSAS UNIVERSITÉ

RAPPORT DE PROJET

Générateur automatique de phrase

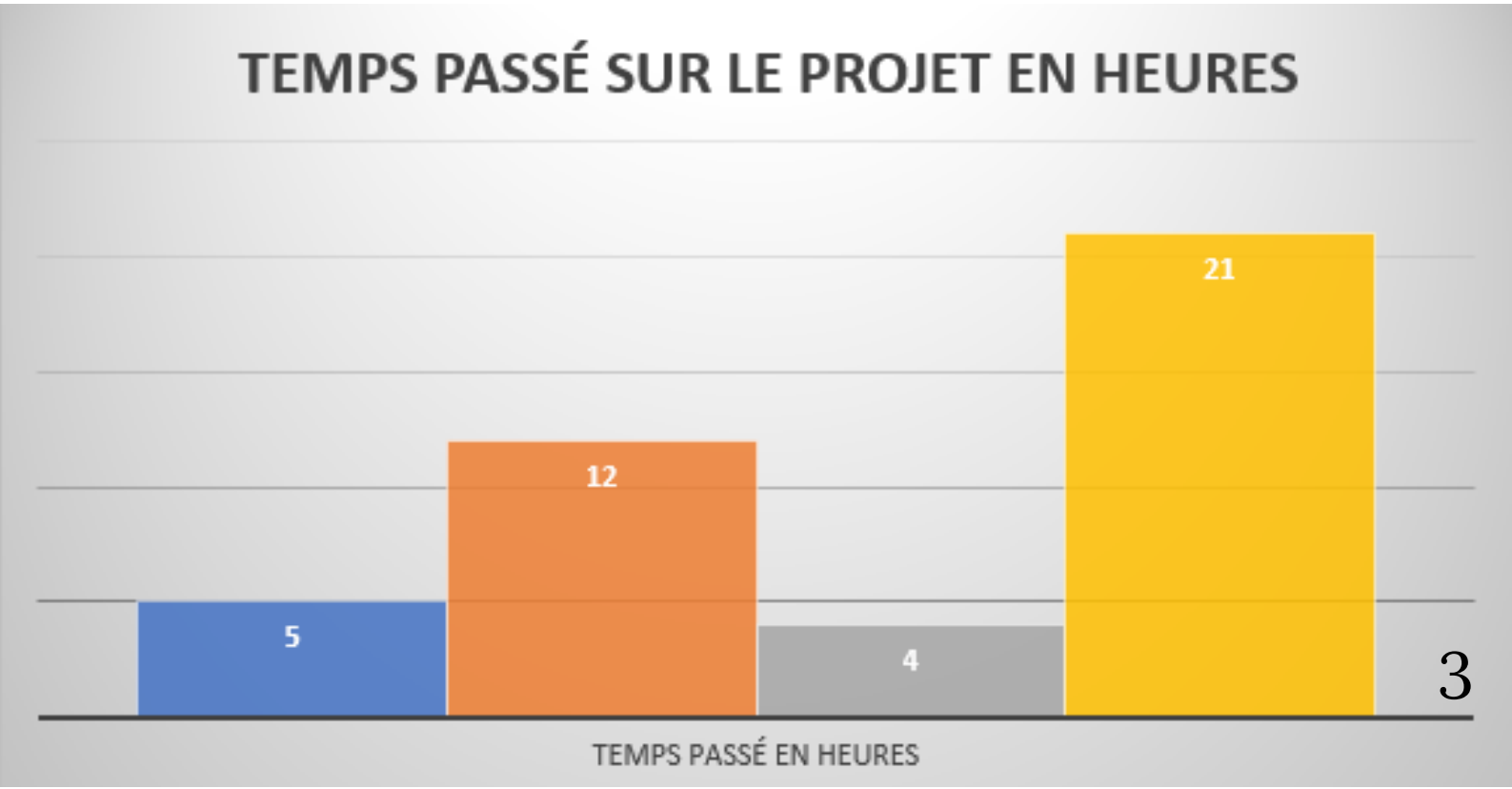
Réalisé par :

GAS ROMAIN
NADAUD RAYAN
LIHARD FLORENT
AKLIMOS JEAN

Table de matière :

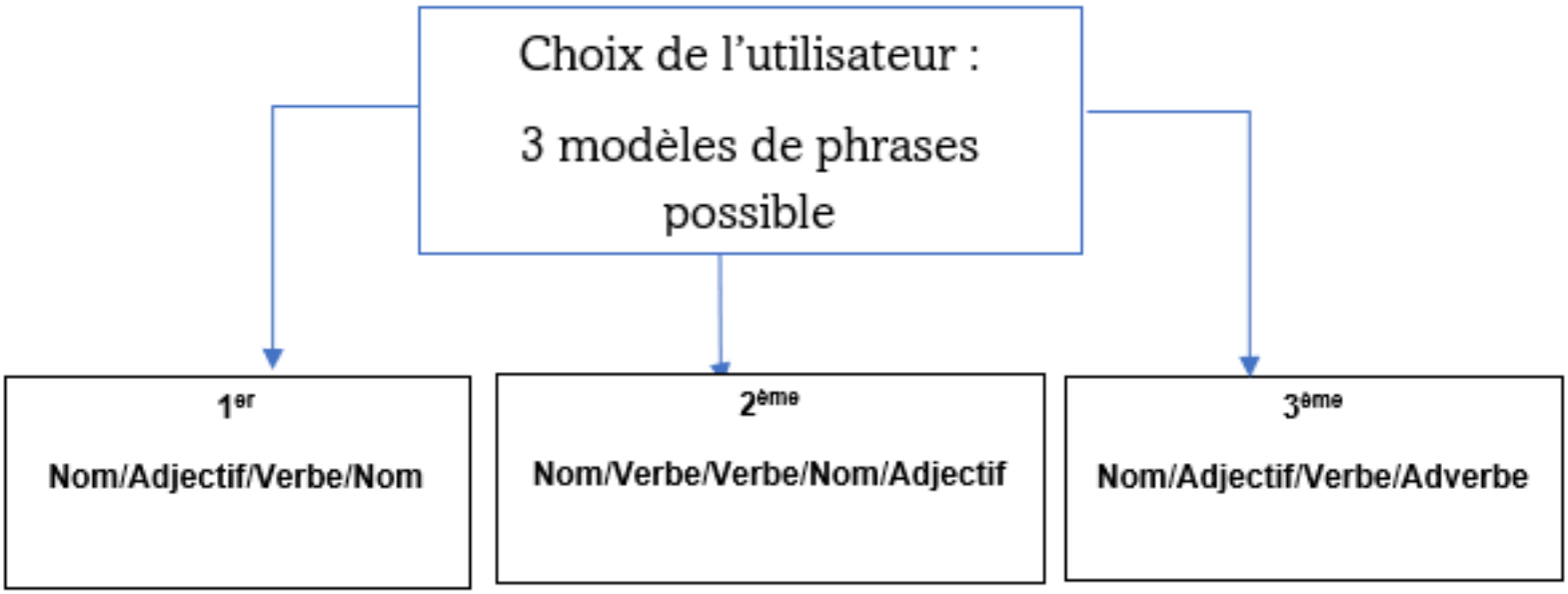
Introduction

L'objectif de ce projet est de créer un programme qui génère automatiquement des
Phrases grammaticalement et (dans la mesure du possible) orthographiquement correctes;
Mais qui n'ont pas forcément de sens.
Ce projet a pour objectif de renforcer nos compétences acquises durant ces deux années sur le langage [C].
Ce projet a aussi comme but d'améliorer notre façon de communiquer, de travailler en groupe, ainsi que notre organisation et notre répartition du travail.



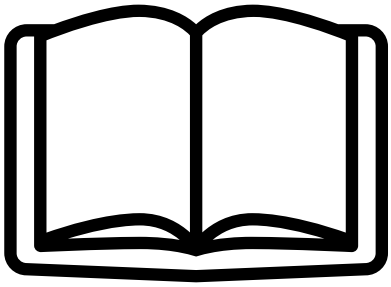
Présentation Fonctionnelle

Lors de l'exécution du programme l'utilisateur a le droit à 3 choix



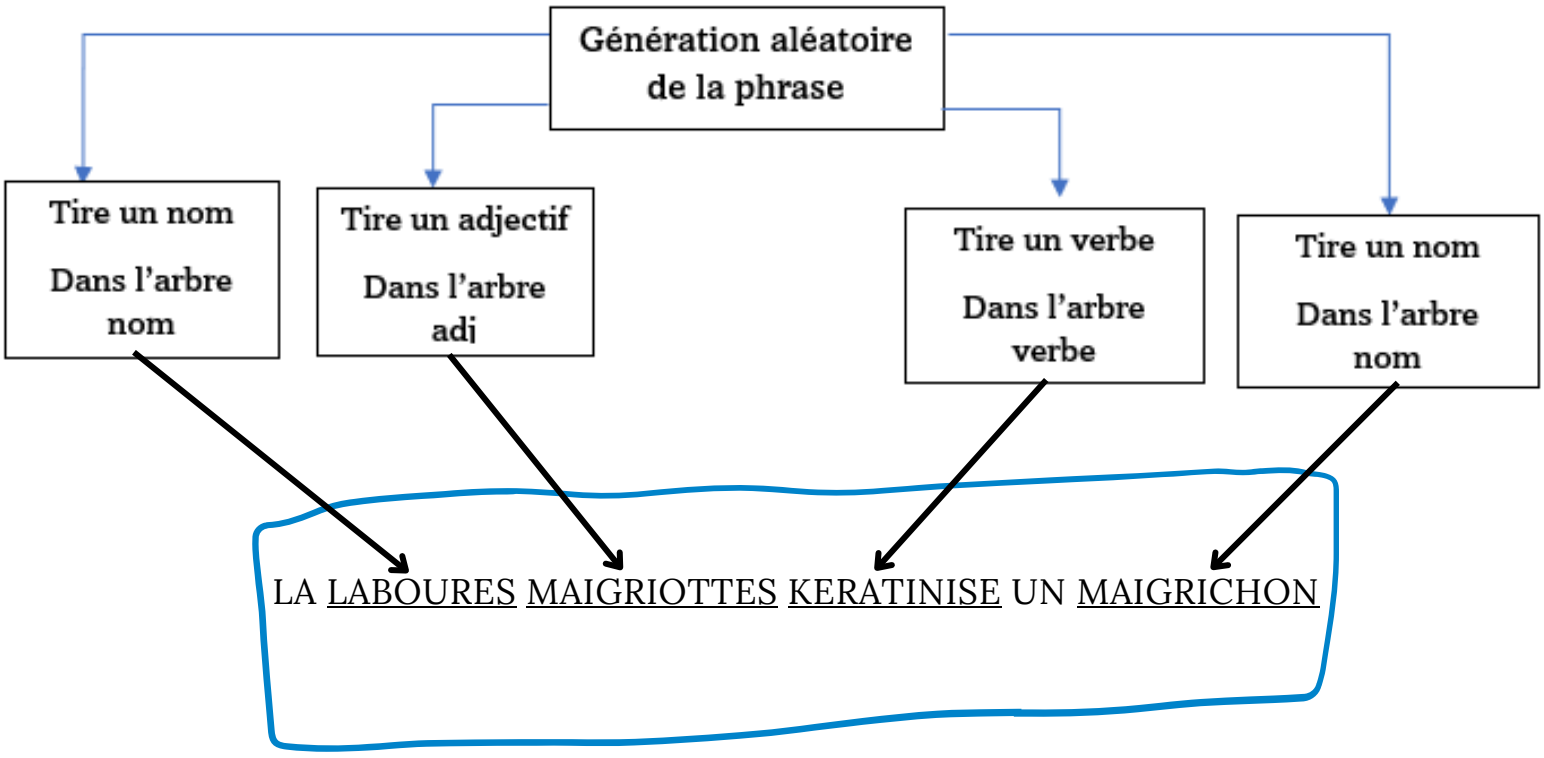
Pour l'exécution des trois choix possible le programme doit être capable de lire le document (dico) et de classer les mots à l'intérieur, selon leurs caractéristiques (nom / adjectif/ verbe/ adverbe). voici les 4 familles principales qui serviront de base pour la création des arbres.

Les mots de chaque famille sont ensuite classés selon des caractéristiques plus précises (exemple : genre/ temps/ singulier ou pluriel)



Selon le choix choisi par l'utilisateur le programme va crée une phrase de façon aléatoire mais en suivant un modèle différents. En fonction, de l'option choisit un nom sera tiré en plus, un verbe ou alors dans le cas du dernier modèle un adverbe.

par exemple dans le cas du modèle 1 on a



Présentation Technique

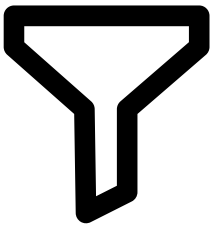
1) Lecture du dictionnaire

Dans un premier temps il nous a fallu faire des recherches pour apprendre a traiter des fichiers en C. En effet après quelques recherches nous avons compris comment manipuler des fichiers en C a l'aide de la librairie <stdio.h>

Le programme lit le fichier ligne par ligne, cette méthode n'est pas la plus optimal niveau rapidité, mais c'est la seule méthode réellement efficace

```
void LireFichier(int option){

 FILE *fp = fopen("../dicodetest.txt", "r");   // Creer un pointeur pour ouvrir le fichier

    char buffer[MAX_LENGTH];
    char str0[MAX_LENGTH],str1[MAX_LENGTH],str2[MAX_LENGTH];
if (fp == NULL)             // Test si le fichier est ouvrable
    {
        printf("Le fichier texte.txt n'a pas pu être ouvert\n");
        return;
    }

    while (fgets(buffer, MAX_LENGTH, fp)){ // On recupere ligne par ligne

     ....}
```

2) Tri des elements



Lors de la lecture, ligne par ligne, le programme tri en fonction du type de l'élément (ex : Verbes / Noms / Adjectifs / Adverbes) et appelle la fonction pour les stocker.

```
if(strcmp(type,"Ver") == 0){                    // Trie  les verbes / adjectifs /

                fill_abr(Verbes,str0);

            }

        } else if(strcmp(type,"Adj") == 0){

                fill_abr(Adjectif,str0);
}


        }else if(strcmp(type,"Nom") == 0){
                fill_abr(Noms,str0);
}


        }else if(strcmp(type,"Adv") == 0){

                fill_abr(Adverbes,str0);

            }
```

3) Creation de l'arbre

```c
p_f_std_liste creerFlisteVide() { // Creer une liste chaine pour les formes flechies
    p_f_std_liste F_liste = (p_f_std_liste) malloc(sizeof(f_std_liste));
    F_liste->suivant = NULL;
    return F_liste;
}

p_std_liste creerListvide(){ //Creer une liste chaine pour les enfants
    p_std_liste Liste = (p_std_liste) malloc(sizeof (t_std_liste));
    Liste->head = NULL;
    return Liste;
}

arbre creerArbre() {  // Initialiser l'arbre
    arbre NouvelArbre;
    NouvelArbre.racine = creerNoeud(':');
    return NouvelArbre;
}



p_noeud_lettre creerNoeud(char lettre){  // Creer un noeud pour l'arbre
    p_noeud_lettre Noeud = (p_noeud_lettre)malloc(sizeof (noeud_lettre));
    Noeud->lettre = lettre;
    Noeud->enfants = creerListvide();
    Noeud->FormeFlechie = creerFlisteVide();
    return Noeud;
}



p_cellule creerCellule(char lettre) { // creer une cellule pour la liste chaine des enfants

    p_cellule NouvelleCellule = (p_cellule) malloc(sizeof(cellule));
    NouvelleCellule->enfant = creerNoeud(lettre);
    NouvelleCellule->suivant = NULL;
    return NouvelleCellule;
}

p_noeud_lettre creerNoeud(char lettre){  // Creer un noeud pour l'arbre
    p_noeud_lettre Noeud = (p_noeud_lettre)malloc(sizeof (noeud_lettre));
    Noeud->lettre = lettre;
    Noeud->enfants = creerListvide();
    Noeud->FormeFlechie = creerFlisteVide();
    return Noeud;
}
```

Nous avons donc décidé de créer 4 arbres N-air pour stocker les éléments triés. Chaque arbre contient donc un noeud avec un enfant, une liste chainée qui contient les formes fléchies ainsi que la lettre car chaque lettre des mots sont stockées dans l'arbre

4) Génération de la phrase

Selon l'option choisie par l'utilisateur le programme va générer une phrase suivant le modèle demandé

```
case 1:{
            // Generation de la phrase numero 1
            // ' nom - adjectif - verbe -nom '
             char *V_split,*N1_split,*N2_split,*A_split;
             V_split = strtok(Verbes,"/");
             N1_split = strtok(Noms,"/");
             N2_split = strtok(Noms,"/");
             A_split = strtok(Adjectif,"/");
             int x;
             x= rand()%  40 + 1;

             while (x>0 ){
                V_split = strtok(NULL,"/");
                x--;
             }

            x= rand()% 30 + 1;
            while (x>0){
                N1_split = strtok(NULL,"/");
                x--;
            }

            x= rand()% 30 + 1;
            while (x>0 ){
                N2_split = strtok(NULL,"/");
                x--;
            }

            x= rand()% 5 + 1;
            while (x>0){
                A_split = strtok(NULL,"/");
                x--;
            }
```

Ici l'option 1 a été choisit, on aura donc 2 noms 1 verbe ainsi que 1 adjectif qui seront récupérés au hasard

Résultats

Lors de l'exécution du programme un menu utilisateur s'affiche

```
Bienvenu sur le Générateur de phrases :
Vous avez les choix entre 3 modèles de phrases différentes à générer
Option 1 : le modèle ' nom - adjectif - verbe - nom '
Option 2 : le modèle ' nom - qui - verbe - verbe - nom - adjectif '
Enfin Option 3 : le modèle 'nom - adjectif - qui - verbe - adverbe'
Que choisissez vous ? :
```

L'utilisateur peut alors choisir parmi les 3 options

Une entrée sécurisée a été mise en place

```
Que choisissez vous ? :4

Cette option n'existe pas !
Bienvenu sur le Générateur de phrases :
Vous avez les choix entre 3 modèles de phrases différentes à générer
Option 1 : le modèle ' nom - adjectif - verbe - nom '
Option 2 : le modèle ' nom - qui - verbe - verbe - nom - adjectif '
Enfin Option 3 : le modèle 'nom - adjectif - qui - verbe - adverbe'
Que choisissez vous ? :
```

Exemple numéro 1 : Si l'utilisateur choisit l'option 1

```
Bienvenu sur le Générateur de phrases :
Vous avez les choix entre 3 modèles de phrases différentes à générer
Option 1 : le modèle ' nom - adjectif - verbe - nom '
Option 2 : le modèle ' nom - qui - verbe - verbe - nom - adjectif '
Enfin Option 3 : le modèle 'nom - adjectif - qui - verbe - adverbe'
Que choisissez vous ? :1

Vous avez choisis le modèle ' nom - adjectif - verbe - nom '
Votre phrase généré : La exoterique palladien abstrait une normales
Process finished with exit code 0
```

Exemple numéro 2 : Si l'utilisateur choisit l'option 2

```
Bienvenu sur le Générateur de phrases :
Vous avez les choix entre 3 modèles de phrases différentes à générer
Option 1 : le modèle ' nom - adjectif - verbe - nom '
Option 2 : le modèle ' nom - qui - verbe - verbe - nom - adjectif '
Enfin Option 3 : le modèle 'nom - adjectif - qui - verbe - adverbe'
Que choisissez vous ? :3

Vous avez choisis le modèle 'nom - adjectif - qui - verbe - adverbe'
Votre phrase généré : Le non-euclidien observable qui logiques normaux
```

Exemple numéro 3 : Si l'utilisateur choisit l'option 3

```
Bienvenu sur le Générateur de phrases :
Vous avez les choix entre 3 modèles de phrases différentes à générer
Option 1 : le modèle ' nom - adjectif - verbe - nom '
Option 2 : le modèle ' nom - qui - verbe - verbe - nom - adjectif '
Enfin Option 3 : le modèle 'nom - adjectif - qui - verbe - adverbe'
Que choisissez vous ? :2

Vous avez choisis le modèle ' nom - qui - verbe - verbe - nom - adjectif '
Votre phrase généré : Le situes qui hypothecaires nord un subintrantes touchants
```

Conclusion :

En conclusion, nous allons faire un bilan des enseignements retenus de la réalisation du projet. Dans un premier temps pour les apprentissages sur le plan technique, Rayane a pu renforcer ses compétences en manipulation de chaine de caractère et en création d'arbre en C il a eu l'occasion d'en apprendre davantage sur la génération de phrases aléatoires et en stockage de mot dans des arbres grâce à ses coéquipiers et aux professeurs présents durant les cours de Travaux diriger.

Quant à Romain, il a consolidé ses compétences acquises en C, notamment avec les arbres et il a également appris à manipuler des fichiers et des chaines de caractères en C.

Ensuite concernant l'organisation du travail et la communication au sein de l'équipe, nous avons su nous adapter et ce grâce aux projets informatiques et en communication de l'année précédente qui nous ont formé sur les travails du groupe ce qui est une qualité nécessaire pour notre future vie professionnelle, aucun problème de communication n'a été rencontré.
Enfin nous avons eu un léger problème dans la gestion du temps et d'organisation avec les cours et les contrôles continus nous avons à tort mis en pause le projet pour favoriser les autres matières.