

Projet Prep'ISIMA : Contrepèterie

Thomas Combettes & Corentin Sabier

29 mai 2021



Table des matières

1	Une contrepètrie c'est quoi ?	3
1.1	Définition stricte	3
1.2	Définition totale	3
2	Explication non-technique	5
2.1	De la recherche	5
2.2	De la création	5
3	Implémentation	7
3.1	De la recherche	7
3.1.1	Nos sources	7
3.1.2	Arbre	7
3.1.3	Fabrication de Contrepètrie	9
3.1.4	Menu	12
3.1.5	Affichage	12
3.2	Aide	13
3.2.1	Nos sources	13
3.2.2	Arbre	13
3.2.3	Outil d'aide	14
3.3	Menu	15
3.4	Affichage	15
4	Les filtres	17
4.1	Les filtres sur la recherche	17
4.1.1	Filtre Grossier	17
4.1.2	Filtre Grammatical	17
4.2	Les filtres sur l'aide	17
4.2.1	Filtre Grossier	17
4.2.2	Filtre Grammatical	18
4.2.3	Exemple d'utilisation :	18
5	Les problèmes	20
5.1	Les problèmes résolus	20
5.1.1	Le problème avec les phonèmes	20
5.2	Les problèmes non-résolus	20
5.2.1	Le problème de Classe Grammaticale	20
5.2.2	Le problème de découpage par mot	20
5.2.3	Contrepètries régulières	20
6	Message aux Suivants	21
6.1	Si on vous demande par hasard un site web	21
6.2	Si on vous demande d'améliorer l'algorithme	21
6.3	Contact	21
7	Conclusion	21

1 Une contrepètrie c'est quoi ?

1.1 Définition stricte

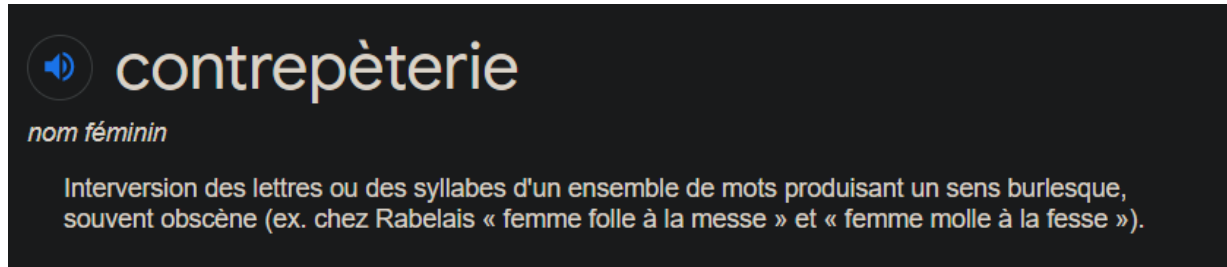


FIGURE 1 – Définition de Google

1.2 Définition totale

Cette définition nous explique que les contrepètries sont des inversions : mais il existe une infinité d'inversions. Par faire simple certains les ont classifiées par type dont voici la liste :

Contrepètries classiques

Ce sont celles qui portent sur des permutations de consonnes. Elles sont en général plus simples à résoudre que les autres et plus facilement sujettes au lapsus linguae, dont elles sont d'ailleurs souvent issues. Il n'en demeure pourtant pas moins que ce sont aussi les plus difficiles à inventer de toutes pièces.

- Consonnes simples au début des mots :
La cuvette est pleine de bouillon.
C'esptune jolie bête qu'un faucon !
- Consonnes simples à l'intérieur des mots :
Le séminariste rêve de se voir en curé avec une calotte.
A Beaumont-le-Vicomte.
- Consonnes simples au début ou à l'intérieur des mots :
Le cuisinier a un canard sur le feu.
Ne vous trempez pas, Lafitte, dans le Bosphore.
- Consonnes doubles ou multiples au début des mots :
Le fakir est arrivé à pied par la Chine.
Cette presse est spécialiste du foot.
- Consonnes doubles ou multiples à l'intérieur des mots :
Brancher les colonnes.
N'approchez pas les mains de la piste de l'égoïne !
- Consonnes doubles ou multiples au début ou à l'intérieur des mots :
La Chine se dresse devant les Nippons.
Elle avait un chapelet de citrouilles autour du cou.

Contrepètries décadentes Une contrepèterie est dite décadente lorsqu'elle porte sur la permutation de voyelles, groupes de voyelles, diphtongues, syllabes, fractions de mots, voire mots entiers. Du coup, le lapsus est moins naturel, et la permutation est plus souvent provoquée qu'involontaire. Mais elle n'en est pas moins drôle !

- Contrepèteries portant sur deux voyelles
Attention aux pannes de micro !
J'ai trempé ma botte dans votre citerne.
- Contrepèteries portant sur voyelles et diphtongues
Jeune homme, ces crampes vous font-elles bouder ?
J'ai croisé une belle Africaine au Pont-Marie.
- Contrepèteries de syllabes ou monosyllabes
Le Pont-Neuf fait soixante pieds.
Le prêtre est-il fou ?
- Contrepèteries échangeant voyelles (ou diphtongues) et syllabes
Joseph a maculé Henri.
La supérieure n'admet pas l'abbé parasite.
- Contrepèteries portant sur des mots ou fractions de mots
Je vous propose une escalope sur une belle salade.
C'est trop de puces pour un village.

Contrepèteries irrégulières Pour simplifier, on range dans cette dernière catégorie, toutes les contrepèteries qui n'appartiennent pas aux deux précédentes. Ces contrepèteries résultent du déplacement d'une ou plusieurs lettres, ou d'une permutation de lettres à l'intérieur d'un mot, voire dans les cas extrêmes d'une véritable salade de phonèmes relevant de l'anagramme phonétique. A ce niveau, le lapsus n'est même plus vraisemblable.

- Déplacement de consonnes
La comtesse déteste les gants qui plissent.
La bibliothécaire fait exposer les Pline.
- Déplacement de voyelles
Murillo a peint une vierge entre deux ascètes.
Pas de bonne chasse sans solides pieds !
- Contrepèterie à l'intérieur d'un même mot
Gershwin.
Caleçon !
- Déplacement de syllabes, mots ou fractions de mots
Attention aux congères de mai !
Ah, Monsieur, votre piscine est belle !
- croisement de voyelles et de consonnes
Ce cas de Corée me turlupine.
Cette pauvre cale s'est retrouvée enfumée.
- Permutation circulaire
Ce jeune homme danse comme un ballot.
L'on voit parfois sur les quais des gueux qui ronflent.

Dans le cas d'une permutation circulaire les éléments à permuter prennent successivement la place du suivant (ou du précédent) dans le sens de lecture de la phrase, le dernier prenant alors la place du premier (ou inversement).

Si vous souhaitez les réponses affichées (pour ceux qui n'aiment pas chercher) :

`urlhttps://www.contrepets.net/classification.php`

2 Explication non-technique

2.1 De la recherche

La plupart des contrepèteries ci-dessus sont très dures à imaginer pour un humain (on peut imaginer que tous les textes en regorgent que nous n'arrivons pas à trouver vu le nombre insensé d'inversions possibles). Il nous vient alors une question : si nous le faisons faire à un ordinateur ? C'est une partie de notre projet qui se résume à la recherche de contrepètries dans les phrases. Elle consiste à retrouver dans des phrases les échanges qui permettent de former une autre phrase. Pour des raisons de complexités (dans les deux sens du mots) nous ne cherchons que les contrepètries dites classiques ou décadentes.

L'objectif de cette fonction est de trouver tout les échanges possibles entre les mots la phrase qu'on lui donnera et donc de retourner toutes les contrepètries possibles de la phrase rentrée.

C'est cette partie qui représente le plus gros challenge au niveau technique, car il faut élaborer des algorithmes pour tout les types de contrepètries citées ci-dessus, et filtrer les phrases que la fonction retournera. C'est filtrer sont particulièrement durs à élaborer car il s'agit de détecter si une phrase en *français* est syntaxiquement correcte.

2.2 De la création

D'autre par, nous avons mis au point un autre outil permettant de voir à partir d'un mot, tout les mots que l'on peut retrouver en échangeant une partie de celui-ci (que ce soit une partie de ses caractères ou de ses sons).

Addmettons que nous voulions trouver les mots obtenables à partir de **poule** en ne changeant qu'une seule lettre

(note : on pourrait chercher à le faire sur plusieurs lettre ou même les sons, mais l'exemple est plus parlant ici) :

- boule
- coule
- goule
- moule
- puce
- pouls

À cette liste est donc l'ensemble des mots que l'on peut obtenir en échangeant une lettre de **poule**. On obtient "**moule**" en échangeant le "**p**" par un "**m**".

Maintenant, que l'on sait vers quel mot on veut transformer notre mot d'origine (ou l'opposé), il serait utile de savoir avec quels autres mots dans une phrase il serait possible de faire des échanges avec les mêmes lettres (parties ou sons, encore une fois l'idée est la même selon ce que l'on veut échanger). Nous cherchons donc un quadruplet de mots de la manière suivante :

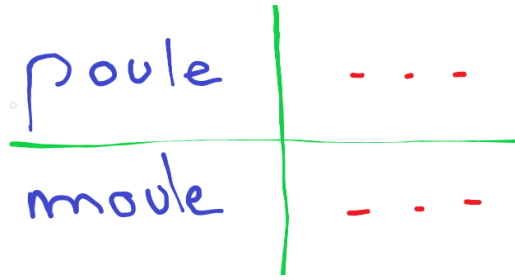


FIGURE 2 – Illustration si on choisit moule

Le programme s'occupe du reste et trouve ces deux autre mots manquant.
ex : "ta poule rame !" → "ta moule rape !"

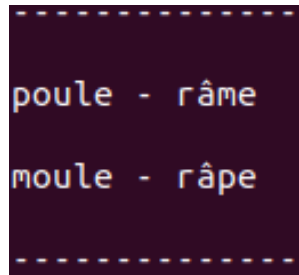


FIGURE 3 – Exemple de quadruplet possible

Ainsi à partir des résultats on peut avoir des idées de phrases contrepétrables (oui c'est un terme inventé par notre équipe) intéressantes.

3 Implémentation

3.1 De la recherche

3.1.1 Nos sources

Nous avons plusieurs sources :

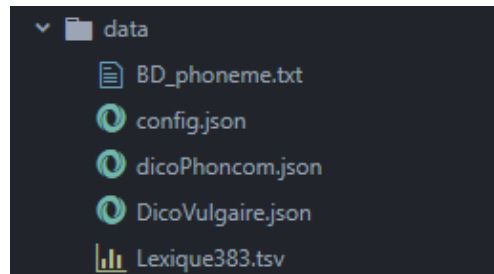


FIGURE 4 – data

- BDphoneme : Liste tous les phonèmes (utiles pour les remplacements)
- config : Un dictionnaire qui porte les infos sur les filtres (il se modifie seul)
- dicoPhoncom : Un immense dictionnaire qui résoud un problème qu'on expliquera plus tard
- DicoVulgaire : Un immense dictionnaire qui contient tous les mots vulgaires de la langue française
- Lexique383.tsv : TOUS les mots de la langue française

3.1.2 Arbre

Afin de ne pas perdre des nombreuses secondes à chaque fois qu'on recherche un mot dans le dictionnaire nommé Lexique383.tsv nous avons utilisé une structure d'arbre binaire de recherche (c'est arbin.py).

Arbin comporte 3 choses utiles à savoir, tout d'abord la class "Tree" qui fonctionne comme l'arbre si dessous sauf qu'on classe des mots (python classe leurs valeurs ASCII)

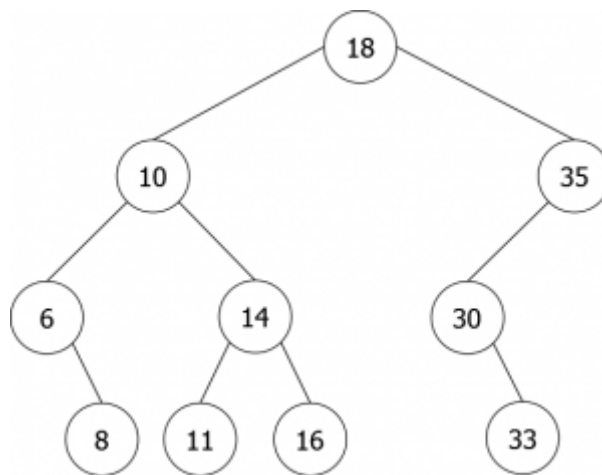


FIGURE 5 – Calsse Tree de arbin.py

Après avoir créé le moule à arbre, il nous faut faire deux arbres : un qui classent selon l'écriture française et l'autre qui classe selon l'écriture phonétique.

```
def Constructeur_Arbre_Mot():
    a = Tree("aaa", None, None)
    tsv_file = open("data/Lexique383.tsv", encoding="utf-8")
    read_tsv = csv.reader(tsv_file, delimiter="\t")
    for lignes in read_tsv:
        if " " not in lignes[0]:
            mot = lignes[0] + "," + lignes[1] + "," + \
                lignes[3][:3] + "," + lignes[4] + "," + lignes[5]
            insert(a, mot)
    tsv_file.close()
    return a

# -----

def Constructeur_Arbre_Phon():
    a = Tree("aaa", None, None)
    tsv_file = open("data/Lexique383.tsv", encoding="utf-8")
    read_tsv = csv.reader(tsv_file, delimiter="\t")
    for lignes in read_tsv:
        if " " not in lignes[0]:
            mot = lignes[1] + "," + lignes[0] + "," + lignes[3] + lignes[4] + lignes[5]
            insert(a, mot)
    tsv_file.close()
    return a

# -----

arbre_mot = Constructeur_Arbre_Mot()
arbre_phon = Constructeur_Arbre_Phon()
```

FIGURE 6 – Fonction constructeur d'arbre de arbin.py

Puis des fonctions afin de récupérer les valeurs dans l'arbre il en existe plusieurs mais elles marchent avec la même méthode récursive (en voici deux exemples) :

```
def Phon_to_Mot(tree, value):
    if tree is None:
        return False
    if (tree.value.split(",")[0]) == value:
        return (tree.value.split(",")[1] + "," + tree.value.split(",")[2])
    if (tree.value.split(",")[0]) is not None and value < (tree.value.split(",")[0]):
        return Mot_to_Phon(tree.left, value)
    elif (tree.value.split(",")[0]) is not None:
        return Mot_to_Phon(tree.right, value)

# -----

def Mot_to_Phon(tree, value):
    if tree is None:
        return False
    if (tree.value.split(",")[0]) == value:
        return (tree.value.split(",")[1] + "," + tree.value.split(",")[2])
    if (tree.value.split(",")[0]) is not None and value < (tree.value.split(",")[0]):
        return Mot_to_Phon(tree.left, value)
    elif (tree.value.split(",")[0]) is not None:
        return Mot_to_Phon(tree.right, value)
```

FIGURE 7 – recherche dans l'arbre

3.1.3 Fabrication de Contrepètrie

Pour rechercher une contrepètrie il faut faire des inversions puis tester si on obtient un résultat français prenons un exemple :

Les femmes laissent leur **coeur** aux vaincus.

Ici en inversant les lettres, nous n'obtiendrons jamais le contrepètrie attendu car "cus" n'est pas un mot français :

Les femmes laissent leur **cus** aux vainqueurs.

Il faut donc utiliser les phonèmes qui en inversant les sons retrouveront la bonne réponse. Pour cela nous avons fait deux méthodes de recherche : une sur les sons et une sur les lettres.

Nous ne travaillons pas sur une lettre mais sur un groupe de lettres comme il faut le faire afin de retrouver pour cette contrepètrie. Nous fonctionnons avec la même méthode pour les sons . Ces méthodes sont dans echSyllabes.py

Leur fonctionnement est assez simples, elles prennent la phrase rentrée :

Les femmes laissent leur **coeur** aux vaincus.

Puis tentent absolument tous les échanges possibles puis testent l'appartenance, regardent s'ils sont dans l'arbre (Mot to Phon ou Phon to Mot) de chacun des mots des combinaisons .

Pour tenter tous les échanges possibles nous avons fait un fastidieux travail de découpage sur les chaines de caractères (l'expliquer n'avance à rien dans le cadre de la compréhension du projet).

```
Phrase à sonder :  
Les femmes laissent leur coeur aux vaincus  
  
Traitement en cours ...  
  
Activer filtre Grammaticale  
(1:Oui/0:Non/n'importe quelle clef:défaut):0  
  
Activer filtre Grossier  
(1:Oui/0:Non/n'importe quelle clef:défaut):0  
  
Les contrepétries possibles sont :  
  
Les leurs laissent femme coeur aux vaincus  
Les coeurs laissent leur femme aux vaincus  
Les femmes laissant leur coeur eux vaincus  
Les femmes laissent vaincu coeur aux leurs  
Les femmes laissent incus coeur aux valeur  
Les femmes laissent leur vaincu aux coeurs  
  
Nombre de résultats : 6  
0 : Quitter / 1 : Retour au début
```

FIGURE 8 – Avec les lettres(sans filtres)

Comme prévu, le contrepét n'est pas dans l'échange de lettres mais il est dans l'échange de sons

```
les femmes laissent leur coeur aux vaincus
Traitement en cours ...
Activer filtre Grammaticale
(1:Oui/0:Non/n'importe quelle clef:défaut):0
Activer filtre Grossier
(1:Oui/0:Non/n'importe quelle clef:défaut):0
0 --> Fée lames laissent leur coeur aux vaincus
1 --> Famée l' laissent leur coeur aux vaincus
2 --> Lames fée laissent leur coeur aux vaincus
3 --> Laissez femmes l' leur coeur aux vaincus
4 --> Leurrez femmes laissent l' coeur aux vaincus
5 --> Heur femmes laissent leur clefs aux vaincus
6 --> Leurre femmes laissent leur quais aux vaincus
7 --> Ohé femmes laissent leur coeur l' vaincus
8 --> Lot femmes laissent leur coeur é vaincus
9 --> Vainquez femmes laissent leur coeur aux lût
10 --> Lins femmes laissent leur coeur aux vécues
11 --> Lût femmes laissent leur coeur aux vainquez
12 --> Les lames fèces leur coeur aux vaincus
13 --> Les lemme faces leur coeur aux vaincus
14 --> Les flemme as leur coeur aux vaincus
15 --> Les fèces lames leur coeur aux vaincus
16 --> Les faces lemme leur coeur aux vaincus
17 --> Les lames laissent feurre coeur aux vaincus
18 --> Les fleurs laissent âme coeur aux vaincus
19 --> Les feurre laissent lames coeur aux vaincus
20 --> Les cames laissent leur feurre aux vaincus
21 --> Les ram laissent leur keufs aux vaincus
22 --> Les feurre laissent leur cames aux vaincus
23 --> Les fac laissent leur meurent aux vaincus
24 --> Les homes laissent leur coeur fa vaincus
25 --> Les faux laissent leur coeur âme vaincus
26 --> Les femmes leurre leur caisses aux vaincus
27 --> Les femmes lek leur soeurs aux vaincus
28 --> Les femmes hausse leur coeur lait vaincus
29 --> Les femmes lot leur coeur esse vaincus
30 --> Les femmes laissent lot coeur heur vaincus
31 --> Les femmes laissent vainqueurs coeur aux lût
32 --> Les femmes laissent lût coeur aux vainqueurs
33 --> Les femmes laissent leur co heur vaincus
35 --> Les femmes laissent leur culs aux vainqueurs
```

FIGURE 9 – Avec les sons(sans filtres)

3.1.4 Menu

Dans le cadre d'un projet comme celui-là, la gestion des fichiers et les connections entre eux est un problème. C'est pourquoi la fonction `input.py` gère le menu principal, de plus elle fait appel à `aideContre.py` qui gère le menu de l'aide (c'est la partie suivante).

De surcroit, toutes les entrées sont vérifiées afin de ne pas faire planter le programme.

```
while test:

    print(
        """\nSelectionnez le mode que vous souhaitez : \n
1. Aide à la contrepéterie
2. Recherche de contrepéterie
3. Configuration des filtres
0. Quitter\n""")

    try:
        n = int(input())
    except ValueError:
        print("Vous n'avez pas saisi un nombre.\n")

    if n == 0:
        sys.exit()
    elif n in range(1, 4):
        test = False
    else:
        print("Votre saisie n'est pas valide\n")
```

FIGURE 10 – Vérificateur

3.1.5 Affichage

Dans certains cas, le nombre de contrepètries est énorme c'est pourquoi nous avons fait des affichages adaptés au nombre de contrepets attendus, l'affichage sera donc différent en fonction de l'échange sur les lettres et les sons car leur nombre est différent.

Dans tout les cas l'affichage de la fonction recherche est dans le dossier filtre car l'affichage est lié aux filtres (`affiRechFiltre` dans `filtre.py`).

Il prend entrée un dictionnaire puis s'occupe de l'affichage en fonction des choix de l'utilisateur.

Lorsque nous travaillons avec les phonèmes beaucoup de mots ont la même écriture phonétique mais pas la même écriture orthographique ex :

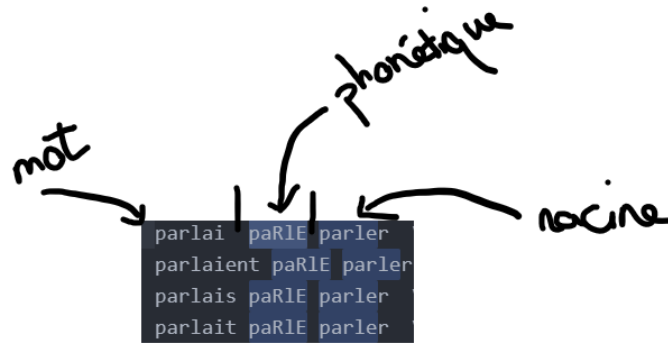


FIGURE 11 – exemple de difficulté d’affichage

Dans ce cas, la fonction de fabrication nous retournera les 4 même phrases (ici il y en a 4 mais souvent c’est 10 ou 15)

C’est pourquoi elle affiche comme vu à la figure 8, un exemple d’orthographe et donne accès aux autres orthographes sur demande de l’utilisateur.

3.2 Aide

3.2.1 Nos sources

Nous avons plusieurs sources :

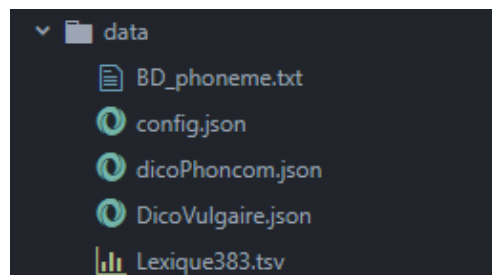


FIGURE 12 – data

- BDphoneme : Liste tous les phonèmes (utiles pour les remplacements)
- config : Un dictionnaire qui porte les infos sur les filtres (il se modifie seul)
- dicoPhoncom : Un immense dictionnaire qui résout un problème qu’on expliquera plus tard
- DicoVulgaire : Un immense dictionnaire qui contient tous les mots vulgaires de la langue française
- Lexique383.tsv : TOUS les mots de la langue française

3.2.2 Arbre

Afin de ne pas perdre des nombreuses secondes à chaque fois qu’on recherche un mot dans le dictionnaire de le Lexique383.tsv nous avons utilisé une structure d’arbre binaire de recherche (c’est arbin.py). Arbin comporte 3 choses utiles à savoir, tout d’abord la class "Tree" qui fonctionne comme l’arbre si dessous sauf qu’on classe des mots (python classe leurs valeurs ASCII) Nous utilisons les mêmes formes de données car elles sont effiaces

3.2.3 Outil d'aide

Tout d'abord, il y a 4 méthodes selon lesquelles on peut créer des contrepètries.

```
print("""Voulez-vous faire une recherche sur :  
1- une lettre  
2- un son  
3- plusieurs lettres  
4- plusieurs sons  
0- quitter l'aide""")
```

FIGURE 13 – Les axes

En fonction du choix de l'utilisateur, nous proposons à l'utilisateur une liste de mots complémentaires avec lesquels faire sa contrepètrie. Ces mots ne sont pas choisis au hasard, ils viennent d'une inversion de la forme :

poule → moule
lancer → labouer

FIGURE 14 – Exemple d'inversion

Cela marche pareil avec les phonèmes mais c'est bien moins visuel. Une fois les deux mots choisis, le programme récupère les groupes de lettres ou de phonèmes inversés pour obtenir les deux mots.

p -> m

ou

an -> bour

Et il teste s'il existe deux autres mots de la langue française qui en forment deux autres avec le même échange.

pue -> mue

ou

labourés -> lancés

Bien qu'elle soit laide : lancer labourés -> labourer lancés est une contrepétie (au sens algorithmique)

3.3 Menu

Le menu est géré par `aideContre.py`.

Il synchronise les demandes de l'utilisateur avec la fonction de génération et la fonction d'affichage.

3.4 Affichage

Dans certains cas le nombre de contrepéties est énorme c'est pourquoi nous avons fait des affichages adaptés au nombre de contrepets attendus, il est différent en fonction de l'échange sur les lettres et les sons car leur nombre est différent.

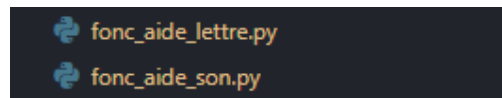


FIGURE 15 – Les fonctions affichage sont avec celles de fabrication

Il prend en entrée une liste à afficher puis s'occupe de l'affichage en fonction des choix de l'utilisateur.

Plutôt que de proposer 1500 mots (ce qui correspond à 1/100 de la langue française) On propose une pré-selection en permettant à l'utilisateur de choisir sur quelle partie du mot il désire son intervention.

```
Écriture phonétique : swaR
1 s -----> 90 mots
2 sw -----> 216 mots
3 swa -----> 472 mots
4 w -----> 32 mots
5 wa -----> 145 mots
6 waR -----> 560 mots
7 a -----> 5 mots
8 aR -----> 27 mots
9 R -----> 25 mots
```

FIGURE 16 – Méthode d'affichage pour les groupes de lettres et sons (p1)

Si l'on choisit waR (6)

```

1 s2          ex: ce          2 s2la        ex: ceux-là
3 s2si        ex: ceux-ci     4 s5          ex: sains
5 s5Z         ex: singent      6 s5Z@        ex: singeant
7 s5ZE        ex: singeait     8 s5Za        ex: singea
9 s5Ze        ex: singer       10 s5d         ex: scinde
11 s5d@        ex: scindant     12 s5dE        ex: scindait
13 s5dR        ex: ceindre     14 s5de        ex: scindée
15 s5du        ex: saindoux     16 s5d$        ex: sindon
17 s5gi        ex: saint-guy    18 s5gl        ex: cinglent
19 s5k         ex: cinq        20 s5pa        ex: sympas
21 s5pl        ex: simples     22 s5t         ex: ceintes
23 s5tR        ex: cintre      24 s5te        ex: synthé
25 s5s         ex: suint       26 s5st        ex: suinte
27 s5sR        ex: sueurs      28 s5@         ex: suants
29 s5@t        ex: suantes     30 s5E         ex: suais
31 s5ER        ex: suaire      32 s5Ed        ex: suède
33 s5a         ex: suât        34 s5av        ex: suaves
35 s5e         ex: suée        36 s5i         ex: suies
37 s5if        ex: suif        38 s5is        ex: suisses
39 s5it        ex: suite       40 s5iv        ex: suiv
41 s5R         ex: soeurs      42 s5Rf        ex: surfes
43 s5j         ex: seuils      44 s5l         ex: seule
45 s5lE        ex: seulet      46 s@          ex: cent
47 s@ba        ex: samba       48 s@bl        ex: semblent
49 s@b$        ex: sent-bon    50 s@dR        ex: sandre
51 s@do        ex: sandows     52 s@g5        ex: sanguin
53 s@gl        ex: sangle      54 s@p@        ex: sampan
55 s@pl        ex: sample      56 s@s         ex: sens
57 s@se        ex: sensées     58 s@sy        ex: sangsue
59 s@t         ex: sentent     60 s@t@        ex: sentant

Les mots obtenables en remplaçant 'waR' dans 'swaR' ('soir')

(0 : quitter l'aide/-3: revenir à selection précédente /-4: revenir au début de l'aide)
(-1:Gauche / -2:Droite) ou saisissez numéro du mot :

```

FIGURE 17 – Méthode d’affichage pour les groupes de lettres et sons (p2)

Chaque page affiche 60 résultats, on peut changer de page comme dans un livre avec -2 et -1, pour pouvoir afficher toutes les contrepéties avec ces deux mots, il faut sélectionner l’index (ici entre 1 et 561)

Pour comprendre admettons que l’on choisisse "sangle" le 53 (en changeant un groupe de phonème on peut passer de soir à sangle).

On obtient 200 résultats, du type :

```

    Phonèmes
    swaR - vi
171  s@gl - vwaRi
-----
    Un exemple d'orthographe
    |  soir - vit
ex : |  sangle - voirie
  
```

FIGURE 18 – exemple de Résultat

En l'occurrence l'absence de filtres appauvrit les résultats

4 Les filtres

Tout d'abord, les filtres sont placés dans la mesure du possible a des endroits judicieux dans l'objectif de réduire le temps d'exécution et le nombre de résultats affichés, ils sont activables et déactivables facilement.

Les filtres sont liés au fichier config.json qui garde en mémoire vos choix lors de l'exécution.

4.1 Les filtres sur la recherche

4.1.1 Filtre Grossier

Il faut l'activer si vous voulez une contrepétrie grossière, il s'assure qu'il y ai au moins un mot vulgaire dans le résultat.

En théorie le dicoVulgaire contient 500 mots soit 0.33/100 de la langue française, il réduit donc par 150 le nombre de résultats (seuls les deux mots de la sortie sont testés).

Il vaut mieux l'utiliser avec parcimonie.

Il est placé en condition supplémentaire lorsqu'on remplit le dictionnaire d'affichage.

4.1.2 Filtre Grammatical

Nous avons trouvé un outil nommé pytool.

Il vérifie qu'une phrase donnée a une grammaire possiblement française. Il enlève donc les phrases grammaticalement fausses.

Il est placé juste avant l'affichage afin d'être utiliser le moins possible car il est lent, il prend à peu près 0,3 seconde par phrase.

4.2 Les filtres sur l'aide

4.2.1 Filtre Grossier

Il faut l'activer si vous voulez créer une contrepétrie grossière, il s'assure qu'il y ai au moins un mot vulgaire dans le résultat.

En théorie le dicoVulgaire contient 500 mots soit 0.33/100 de la langue française, il réduit donc par 150 le nombre de résultats (les deux mots de la sortie sont testés).

Il vaut mieux l'utiliser avec parcimonie.

Il est placé en condition supplémentaire lorsqu'on remplit la liste d'affichage


Remarque : Il y a aussi un filtre grossier indépendant qui filtre uniquement le second mots dans les modes 3 et 4.

4.2.2 Filtre Grammatical

Il utilise la capacité de Mot to Phon a trouvé la classe grammaticale d'un mot depuis Lexique383.tsv permettant d'assurer que les mots dans lesquels les inversions sont faites sont de la même classe ce qui facilite grandement leur intégration à des phrases par la suite. Il est placé juste avant l'affichage car il ne prend quasiment pas de temps.

4.2.3 Exemple d'utilisation :

Nous prendrons pour cette exemple l'utilité du pré-filtre grossier indépendant :



Mot saisie : penser		Mot saisie : penser	
1 p	-----> 8 mots	1 p	-----> 0 mots
2 pe	-----> 10 mots	2 pe	-----> 0 mots
3 pen	-----> 241 mots	3 pen	-----> 5 mots
4 e	-----> 1 mots	4 e	-----> 0 mots
5 en	-----> 31 mots	5 en	-----> 1 mots
6 ens	-----> 256 mots	6 ens	-----> 5 mots
7 n	-----> 2 mots	7 n	-----> 0 mots
8 ns	-----> 30 mots	8 ns	-----> 1 mots
9 nse	-----> 28 mots	9 nse	-----> 1 mots
10 s	-----> 2 mots	10 s	-----> 0 mots
11 se	-----> 3 mots	11 se	-----> 0 mots
12 ser	-----> 111 mots	12 ser	-----> 0 mots
13 er	-----> 2 mots	13 er	-----> 0 mots
14 r	-----> 45 mots	14 r	-----> 0 mots

FIGURE 19 – Avec et Sans préfiltre

Nous prendrons pour cette exemple le filtre grammatical :

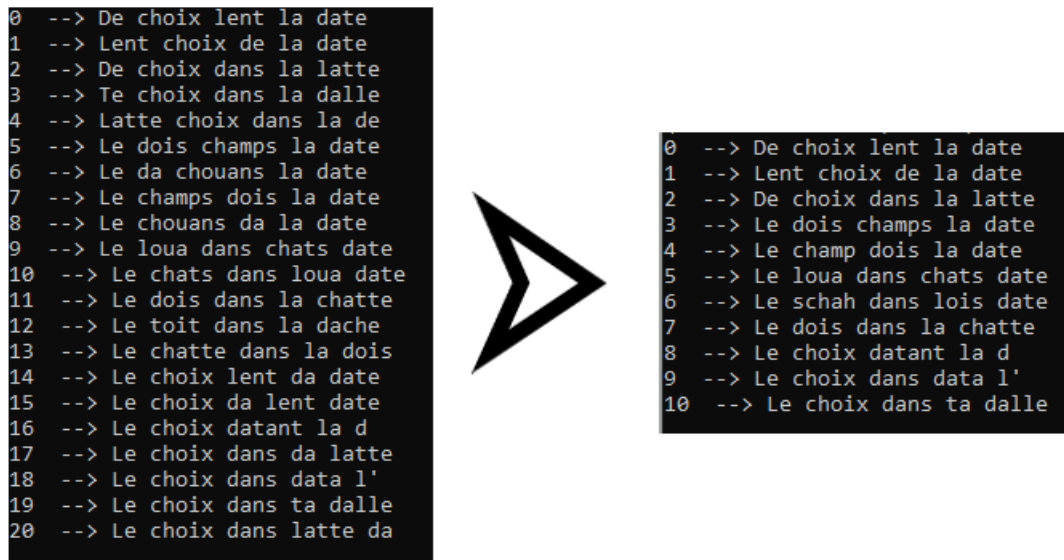


FIGURE 20 – Avec et Sans filtre grammtical

Ici ce n'est dans les exemples d'orthographe, le filtre aura aussi affiner la recherche dans les choix restants (image de droite).

Puis avec un filtre grossier en plus :

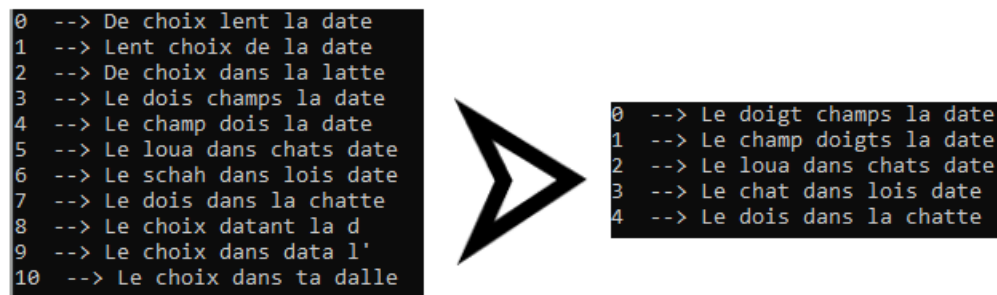


FIGURE 21 – Avec et Sans filtre grossier

5 Les problèmes

5.1 Les problèmes résolus

5.1.1 Le problème avec les phonèmes

Nous avons un problème avec les phonèmes, en effet les arbres trient en fonction des valeurs ASCII du phonème or si deux mots s'écrivent pareils alors il en range à un endroit inaccessible dans l'arbre. c'est pourquoi nous avons eu l'idée de créer un dicoPhoncom qui a pour clé un phonème et pour valeur tout ceux qui s'écrivent pareils(en phonétique).

5.2 Les problèmes non-résolus

5.2.1 Le problème de Classe Grammaticale

Nous avons le même problème qu'au dessus avec les classes grammaticales, car deux mots peuvent avoir la même classe grammaticale. Ce qui fait dysfonctionner Mot to Phon, rendant ainsi le filtre imparfait.

Si nous avions eu plus de temps il aurait fallut que nous fassions un fichier json contenant un dictionnaire avec pour clef chaque mot et pour valeur une liste des classes grammaticales au quel il peut appartenir.

5.2.2 Le problème de découpage par mot

Notre code ne permet pas de trouver les contrepétrie comme :

Le tennis prévisible

Puisque le découpage est fait par mot, afin de le réparer il faudrait changer notre méthode de découpage dans son ensemble.

5.2.3 Contrepétries régulières

De part leur complexité nos fonctions ne peuvent les retrouver ou en créer.

6 Message aux Suivants

6.1 Si on vous demande par hasard un site web

Nous avons à nos heures perdus, chercher comment faire un site à partir du python coté client qui afficherait ses contrepètrie. On vous expose juste les choses à fouiller .

- Si vous êtes des bricoleurs : Aller voir Brypython
`https://brython.info/index.html`
- il y a aussi : Skulpt
`http://skulpt.org/`
- La méthode longue consiste à apprendre et traduire en JavaScript
- Louer un serveur faire tourner le python en coté serveur avec du php (à verifier)

6.2 Si on vous demande d'améliorer l'aglorithmique

- Essayer de bien comprendre comment fonctionne notre partie
- Travaillez proprement (tout dans des fonctions et faire tout les tests) on ne l'a pas fait dès le début par manque d'expérience et ça nous a causé des problèmes par la suite.

6.3 Contact

Thomas : 07 83 35 40 33 – thomascombettes@gmail.com

Corentin : 06 86 06 57 49 - corentin.sabier@etu.uca.com

Nous vous donnons nos coordonnées dans le cas où si vous aviez besoin de conseils sur le projet ou un après-midi pour vous décrire plus en détails les fonctions du projet.

L'objectif serait de vous aidez à vous orienter dans les presque 2000 lignes de code pour que vous perdiez le moins possible votre temps.

7 Conclusion

Ce projet nous aura appris énormément de choses autant au niveau de l'organisation d'un projet de taille plus conséquente que ce qu'on a pu réaliser en prep'isima. De l'utilisation d'outils tels que git à de la résolution de problèmes.

Bien que le résultat puisse ne pas être extrêmement impressionnant visuellement. il est très stimulant du point de vu technique et réalisation puisqu'il nous confronte à énormément de problématiques différentes.

Telles que la gestions de grandes bases de données (et du coup garder un algorithme efficace), de l'algorithmique pure (la recherche et le filtrage de contrepètries de tout les types est un vrais défi), de l'interface utilisateur/machine.