Corruption Littéraire

Lors de cette résidence j'ai voulu expérimenter la corruption sémantique de textes en remplaçant certains mots par des synonymes. Les synonymes sont récupérés dynamiquement à travers l'API du site http://thesaurus.altervista.org. Pourquoi j'ai choisi ce site ? Ben les dictionnaires de synonymes en langue française proposant une API ne courent pas les cyber-rues, le choix a été rapide. Il faut noter que chaque API a ses propres règles de communication. Il devrait donc être possible d'utiliser le programme de corruption de texte avec un autre dictionnaire en ligne à condition d'adapter le programme aux règles de la nouvelle API.
Le programme est composé de deux modules maladroitement intégrés ensemble (ça s'est fait à la va vite sans réelle vision d'ensemble, et puis il y avait une tireuse avec bière à volonté...). Le premier module, écrit en Python, se charge de récupérer les synonymes en lignes et de sauvegarder le texte corrompu dans un nouveau fichier. Le second module, en Processing (Java), fait défiler les textes à l'écran, ajoute des effets de corruption graphique et fait régulièrement appel au premier module.

Le module de corruption de texte

Il peut s'utiliser de façon autonome, à condition d'avoir installé Python3 sur votre machine.
N'oubliez pas de remplacer la clé d'API du site http://thesaurus.altervista.org par votre propre clé (après avoir crée votre compte) dans la variable key.

Usage:
`$ python3 textcorrupt.py` *`fichier_original fichier_de_sortie`* `[`*`nombre_d'itérations`*`]`

textcorrupt.py (cliquer pour afficher le code)

textcorrupt.py

```
import requests
import random
import os.path
import sys

syn_dict = "syn_dict.txt"
filter_out = "french_filter_out.txt"

endpoint = "http://thesaurus.altervista.org/thesaurus/v1"
key = "à remplacer par votre clé personnnelle"
language = "fr_FR"
output = "json"


def loadFilterOut(filename):
    filterOut = []
    if not os.path.isfile(filename):
        pass
    else:
        with open(filename, 'r') as file:
            for line in file.readlines():
                if line:
                    filterOut.append(line.strip())
    return filterOut

def saveFilterOut(filename, filterOut):
    with open(filename, 'w') as file:
        for word in filterOut:
            file.write(word+'\n')


def loadSynDict(filename):
    synDict = dict()
    if not os.path.isfile(filename):
        pass
    else:
        with open(filename, 'r') as file:
            for line in file.readlines():
                if line:
                    words = line.split('\t')
                    synDict[words[0]] = list(map(cleanWord, words[1:]))
    return synDict

def saveSynDict(filename, synDict):
```

```
    with open(filename, 'w') as file:
        for word in synDict:
            file.write(word)
            for syn in synDict[word]:
                file.write('\t' + syn)
            file.write('\n')


def getSynonyms(word, synDict, filterOut):
    word = cleanWord(word)
    if word in synDict:
        return synDict[word]

    sys.stderr.write("Fetching synonym of '{}'...\n".format(word))
    url = f"{endpoint}?word={word}&key={key}&language={language}&output={output}"
    r = requests.get(url)
    try:
        r.raise_for_status()
        json = r.json()
        synList = json["response"][0]["list"]["synonyms"].split('|')
        synList = list(map(cleanWord, synList))
        synDict[word.lower()] = synList
        return synList
    except:
        sys.stderr.write("Synonym not found\n")
        filterOut.append(word)
        return None
    finally:
        sys.stderr.flush()


def chooseSynonym(word, synDict, filterOut):
    syns = getSynonyms(word, synDict, filterOut)
    if syns:
        return random.choice(syns)
    else:
        return word


def validate(word, filterOut):
    return len(word)>1 and word.isalpha() and not word.casefold() in filterOut


def cleanWord(word):
    return word.casefold().strip()


def decomposeWord(word):
    iStart = 0
    iEnd = len(word)
    while(iEnd>0 and not word[iEnd-1].isalpha()): iEnd -= 1
    if len(word)>2 and word[1] == "'": iStart=2

    return word[:iStart], word[iStart:iEnd], word[iEnd:]


def parseText(text, synDict, filterOut):
    lines = text.split('\n')
    corruptedLines = []
    for line in lines:
        words = line.split()
        corruptedWords = []
        for word in words:
            prefix, radix, suffix = decomposeWord(word)
            # The corruption probability is 0.2 for each word
            if random.random()<0.2 and validate(radix, filterOut):
                corruptedWord = chooseSynonym(radix, synDict, filterOut)
                # Keep capital letters
                if word.istitle():
                    corruptedWord = corruptedWord[0].upper() + corruptedWord[1:]
                corruptedWords.append(prefix+corruptedWord+suffix)
            else:
                # Word is kept unchanged
                corruptedWords.append(word)
        corruptedLines.append(' '.join(corruptedWords))
    return '\n'.join(corruptedLines)


if __name__ == "__main__":
    sys.stderr.flush()
    synDict = loadSynDict(syn_dict)
    filterOut = loadFilterOut(filter_out)

    iteration = 1
    if len(sys.argv) > 3: iteration = int(sys.argv[3])

    text = ""
    with open(sys.argv[1], 'r') as file:
        text = file.read()
    for i in range(iteration):
        text = parseText(text, synDict, filterOut)

    # Save to file
    with open(sys.argv[2], 'w') as file:
```

```
            file.write(text)

    saveSynDict(syn_dict, synDict)
    saveFilterOut(filter_out, filterOut)
```

Le sketch Processing de présentation