

Mémoire de fin d'études
Pour l'obtention du diplôme d'Ingénieur d'État en Informatique
Option : Systèmes Informatiques

Création d'un corpus de l'aphasie de Broca et
développement d'un système Speech-to-speech de
réhabilitation de la parole

Réalisé par :
BELGOUMRI Mohammed
Djameleddine
im_belgoumri@esi.dz

Encadré par :
Pr. SMAILI Kamel
smaili@loria.fr
Dr. LANGLOIS David
david.langlois@loria.fr
Dr. ZAKARIA Chahnez
c_zakaria@esi.dz

Table des matières

Page de garde	i
Table des matières	i
Table des figures	ii
Algorithmes et extraits de code	iii
1 Tests et résultats	1
1.1 Génération des erreurs	1
1.2 Corpus	3
A Dépendances et bibliothèques	4

Table des figures

1.1	Fréquences des catégories d'erreurs	1
-----	---	---

Algorithmes et extraits de code

1.1	Génération des erreurs pour un mot	2
-----	--	---

Chapitre 1

Tests et résultats

Le premier chapitre de cette partie est consacré à la présentation de la conception de notre solution. Le deuxième résume les étapes de sa réalisation. Dans ce chapitre, nous présentons les résultats obtenus en les analysant et en les commentant.

1.1 Génération des erreurs

Parmi les erreurs générées par chatGPT, celles qui ressemblent le plus à des erreurs humaines ont été manuellement sélectionnées. Le résultat de cette sélection est une liste de 217 mots avec une moyenne de 5 erreurs retenues par mot (1104 erreurs en termes absolus). Ces erreurs ont été analysées et classées en 4 catégories :

- des suppressions : de lettres ou de syllabes,
- des ajouts : de lettres ou de syllabes,
- des substitutions : de lettres ou de syllabes,
- des transpositions : de lettres ou de syllabes.

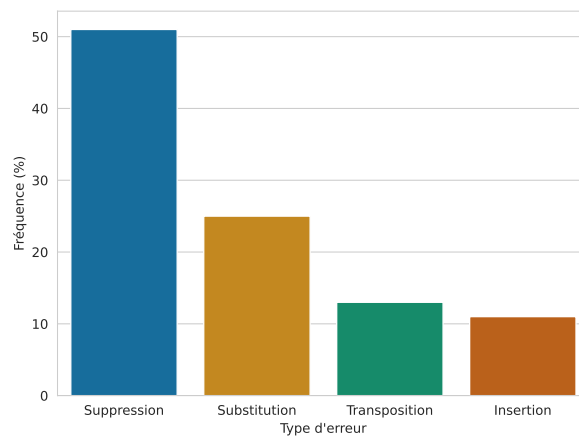


FIGURE 1.1 – Fréquences des catégories d'erreurs

Les fréquences de ces erreurs (pour les 32 premiers mots qui ont 327 modifications) sont présentées dans la Figure 1.1. Sur la base de ces fréquences, nous avons créé une

fonction qui génère pour un mot donné, des erreurs qui suivent les mêmes fréquences (voir le code 1.1).

```
1 def corrupt_word(  
2     word,  
3     p_remove=0.51,  
4     p_substitute=0.25,  
5     p_transpose=0.13,  
6     p_insert=0.11,  
7     p_skip=0.5,  
8     all_syllables=None,  
9 ):  
10     from random import seed, randint, random  
11     from hyphen import Hyphenator  
12  
13     hyphenator = Hyphenator("fr_FR")  
14     syls = hyphenator.syllables(word)  
15  
16     # skip words that are too short  
17     if len(syls) < 3:  
18         return word  
19  
20     # skip all words with probability p_skip  
21     if random() < p_skip:  
22         return word  
23  
24     # remove a syllable with probability p_remove  
25     if random() < p_remove:  
26         idx = randint(0, len(syls) - 1)  
27         del syls[idx]  
28  
29     # substitute a syllable with probability p_substitute  
30     if random() < p_substitute:  
31         idx1 = randint(0, len(syls) - 1)  
32         syls[idx] = choice(all_syllables)  
33  
34     # transpose two syllables with probability p_transpose  
35     if random() < p_transpose:  
36         idx1 = randint(0, len(syls) - 1)  
37         idx2 = randint(0, len(syls) - 1)  
38         syls[idx1], syls[idx2] = syls[idx2], syls[idx1]  
39  
40     # insert a syllable with probability p_insert  
41     if random() < p_insert:  
42         idx = randint(0, len(syls) - 1)  
43         syls.insert(idx, choice(all_syllables))  
44     return "".join(syls)
```

Extrait de code 1.1 Génération des erreurs pour un mot

Les erreurs générées par cette fonction sont similaires aux erreurs générées par chatGPT (par exemple, maintenant → temain | tenant, entendu → enten | tendu | tenendu.). Cependant, certaines parmi elles ne sont pas prononçables (par exemple, maintenant → nantmain, simplement → mentple). Pour cette raison, nous avons décidé de ne pas les utiliser dans le corpus. Cela étant dit, il nous paraît intéressant d’explorer des méthodes de filtrage de ces erreurs. Si réussies, elles permettent de générer des erreurs plus rapidement et plus facilement que par chatGPT.

1.2 Corpus

Annexe A

Dépendances et bibliothèques

```
lightning==2.0.2
torch==2.0.0
pytorch_memlab==0.2.4
PyYAML==6.0
tokenizers==0.13.3
torchdata==0.6.0
torchmetrics==0.11.4
torchtext==0.15.1
torchview==0.2.6
tqdm==4.64.1
beautifulsoup4==4.11.1
openai==0.27.2
pandas==1.5.3
PyHyphen==4.0.3
python-dotenv==1.0.0
Requests==2.30.0
scikit_learn==1.2.0
tokenizers==0.13.3
tqdm==4.64.1
evaluate==0.4.0
```