République Algérienne Démocratique et Populaire
الجمهورية الجزائرية الديـموقراطية الشعبية
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
وزارة التعليم العالي والبحث العلمي

المدرسة الوطنية للإعلام الآلي
(المعهد الوطني للتكوين في الإعلام الآلي سابقا)
École nationale Supérieure d'Informatique
ex. INI (Institut National de formation en Informatique)

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

2022 / 2023

Table des matières

Table des figures

Algorithmes et extraits de code

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.

Les fréquences de ces erreurs (pour les 32 premiers mots qui ont 327 modifications) sont présentées dans la figure 1.1.

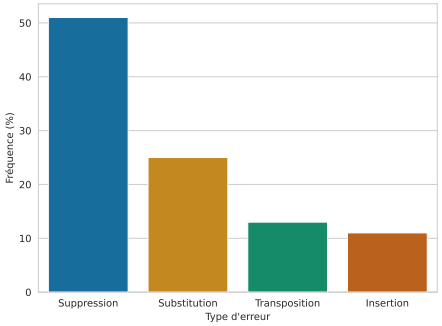

FIGURE 1.1 – Fréquences des catégories d'erreurs

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).

```python
def corrupt_word(
    word,
    p_remove=0.51,
    p_substitute=0.25,
    p_transpose=0.13,
    p_insert=0.11,
    p_skip=0.5,
    all_syllables=None,
):
    from random import seed, randint, random
    from hyphen import Hyphenator

    hyphenator = Hyphenator("fr_FR")
    syls = hyphenator.syllables(word)

    # skip words that are too short
    if len(syls) < 3:
        return word

    # skip all words with probability p_skip
    if random() < p_skip:
        return word

    # remove a syllable with probability p_remove
    if random() < p_remove:
        idx = randint(0, len(syls) - 1)
        del syls[idx]

    # substitute a syllable with probability p_substitute
    if random() < p_substitute:
        idx1 = randint(0, len(syls) - 1)
        syls[idx] = choice(all_syllables)

    # transpose two syllables with probability p_transpose
    if random() < p_transpose:
        idx1 = randint(0, len(syls) - 1)
        idx2 = randint(0, len(syls) - 1)
        syls[idx1], syls[idx2] = syls[idx2], syls[idx1]

    # insert a syllable with probability p_insert
    if random() < p_insert:
        idx = randint(0, len(syls) - 1)
        syls.insert(idx, choice(all_syllables))

    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
```