Rapport sur le test des chunkers sur les données écrites en temps réel

Réalisé par

Diego ROSSINI

Sandra JAGODZINSKA

Sarah ALMEIDA BARRETO

Yingzi LIU

Enseignante : Iris Taravalle

Master TAL Traitement automatique des langues

Cours: Enrichissement de corpus

Année : 2022-2023

1. Introduction

Dans le cadre de notre projet, nous avons entrepris une étude sur l'utilisation des chunkers pour l'analyse de données écrites en temps réel. Nous avons travaillé avec un corpus composé de données enregistrées en temps réel, comprenant des informations telles que la position du curseur, les caractères écrits ou supprimés, les espaces et les textes finaux. Ces données présentent des défis spécifiques en raison de leur nature particulière et ont donc été peu explorées dans le domaine du Traitement Automatique des Langues (TAL).

Notre objectif principal était d'explorer la tâche de chunking appliquée à ces données en temps réel. Le chunking est une technique d'analyse linguistique qui vise à identifier et regrouper des mots en unités linguistiques plus larges, telles que des groupes nominaux ou des groupes verbaux. Cependant, en raison des caractéristiques spécifiques des données en temps réel, cette tâche présente des défis supplémentaires nécessitant une exploration approfondie.

Nous avons structuré notre rapport en plusieurs étapes. Tout d'abord, nous avons formulé des questions et des hypothèses concernant les performances des chunkers sur les données en temps réel. Ensuite, nous avons présenté en détail le corpus de données utilisé, mettant en évidence ses caractéristiques spécifiques. Par la suite, nous avons effectué une recherche approfondie afin de sélectionner les chunkers disponibles les mieux adaptés à notre corpus.

Notre projet vise à améliorer la compréhension de l'efficacité des chunkers appliqués aux données écrites en temps réel. Les résultats obtenus nous permettent d'identifier les défis et les opportunités liés à la segmentation de ces données. Ces connaissances peuvent être utilisées pour améliorer les techniques de traitement automatique des langues adaptées aux données en temps réel.

Dans ce rapport, nous détaillerons chacune des étapes de notre étude, en mettant en évidence les défis rencontrés et les résultats obtenus.

2. Hypothèses et questions de recherche

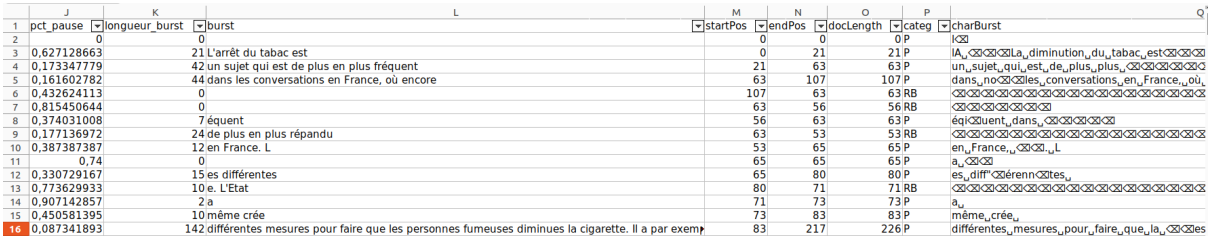
En relation avec notre corpus, nous avons entrepris une recherche et constaté que la segmentation précise des données textuelles est essentielle pour notre étude sur les chunkers. Cependant, il est crucial de comprendre la répartition des pauses dans les données textuelles segmentées et leur relation avec les différentes parties du texte. Nous avons donc proposé quelques questions et hypothèses préliminaires pour orienter notre recherche :

Question 1	Est-ce que les outils disponibles parviennent à bien segmenter les données ?
Hypothèse 1	Les pauses se produisent aux frontières des groupes de mots (chunks).
Question 2	Si oui, est-ce que les pauses se produisent entre des types de groupes de mots particuliers, par exemple après le groupe verbal ?
Hypothèse 2	Les différents processus de textualisation présentent des corrélations similaires entre les pauses et les groupes de mots.

3. Aperçu des données

Les données que nous avons reçues de Mme. Taravella comprend une documentation détaillée sur l'enregistrement en temps réel, le traitement et le processus de rédaction formelle. En plus de cela, nous disposons de trois tableaux au format CSV portant les noms de "formulation", "planification" et "révision", qui représentent les trois étapes du processus de textualisation.

Ces tableaux CSV fournissent des informations spatiales et temporelles sur l'écriture réalisée entre les deux pauses. Ils ont été collectés dans le cadre d'une étude visant à formaliser et analyser les données produites lors de différentes expériences. Nous avons également reçu un ensemble de fichiers au format TXT qui représentent les textes finaux et ce qui a été obtenu à la fin du processus d'écriture. Les figures 1 et 2 suivantes présentent une partie d'un tableau, ainsi que le texte final correspondant.



	J	K	L	M	N	O	P	Q
1	pct_pause	longueur_burst	burst	startPos	endPos	docLength	categ	charBurst
2	0	0		0	0	0	P	l⌫
3	0,627128663	21	L'arrêt du tabac est	0	21	21	P	lA␣⌫⌫⌫⌫La␣diminution␣du␣tabac␣est⌫⌫⌫⌫
4	0,173347779	42	un sujet qui est de plus en plus fréquent	21	63	63	P	un␣sujet␣qui␣est␣de␣plus␣plus␣⌫⌫⌫⌫⌫⌫⌫
5	0,161602782	44	dans les conversations en France, où encore	63	107	107	P	dans␣no⌫⌫les␣conversations␣en␣France,␣où␣
6	0,432624113	0		107	63	63	RB	⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫
7	0,815450644	0		63	56	56	RB	⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫
8	0,374031008	7	équent	56	63	63	P	équ⌫⌫uent␣dans␣⌫⌫⌫⌫⌫⌫⌫
9	0,177136972	24	de plus en plus répandu	63	53	53	RB	⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫
10	0,387387387	12	en France. L	53	65	65	P	en␣France.␣⌫⌫⌫.␣L
11	0,74	0		65	65	65	P	a␣⌫⌫⌫
12	0,330729167	15	es différentes	65	80	80	P	es␣diff⌫⌫érenn⌫⌫tes␣
13	0,773629933	10	e. L'Etat	80	71	71	RB	⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫⌫
14	0,907142857	2	a	71	73	73	P	a␣
15	0,450581395	10	même crée	73	83	83	P	même␣crée␣
16	0,087341893	142	différentes mesures pour faire que les personnes fumeuses diminues la cigarette. Il a par exemp	83	217	226	P	différentes␣mesures␣pour␣faire␣que␣la␣⌫⌫⌫es

Figure 1 Tableau “formulation”

L'arrêt du tabac est un sujet de plus en plus répandu en France. L'Etat a même crée différentes mesures pour faire que les personnes fumeuses diminues la cigarette. Il a par exemple interdi la cigarette et la cigarette électronique dans les lieux publics. Mais il y a aussi une augmentation du prix du paquet ou encore la création de paquet neutre.

On peut voir que certaines de ces mesures permettent aux personnes de diminuer vraiment la cigarette. Comme par exemple, l'augmentation du prix du tabac, car les personnes a plus faibles revenu ou ne voulant pas mettre trop d'argent dans les cigarettes vont diminuer ou alors passer des cigarettes pré-faites aux cigarettes à roulées.

Figure 2 Texte final sur tableau “formulation”

Dans le tableau, chaque colonne offre des perspectives différentes sur les caractéristiques de l'écriture. Par exemple, la colonne "longueur_burst" indique le nombre de caractères présents dans le texte à la fin de l'intervalle de pause, ce qui permet d'évaluer la quantité de texte produite pendant chaque intervalle et donne un aperçu de la dynamique de l'écriture.

La colonne "burst" indique ce qui a été écrit à la fin de chaque intervalle de pause. Cette information est utile pour examiner le contenu spécifique qui a été produit et peut être utilisée pour analyser les choix linguistiques ou les motifs thématiques dans le texte.

Les colonnes "startPos" et "endPos" fournissent la position de départ et de fin du curseur au moment où l'écriture commence et se termine entre deux pauses. Ces informations spatiales permettent de comprendre la distribution et les mouvements du curseur pendant le processus d'écriture, ce qui peut être pertinent pour étudier la planification ou la révision du texte.

Enfin, la colonne "charBurst" enregistre toutes les actions effectuées sur le clavier, y compris les frappes, les effacements et les espaces. Cette colonne offre une vision détaillée des actions effectuées pendant chaque intervalle de pause et permet d'analyser les erreurs de frappe, les corrections ou les comportements spécifiques liés à l'écriture.

4. Choix du chunker

Nous avons procédé à des tests et à l'évaluation de quatre chunkers différents : SEM, TreeTagger, SpaCy et NLTK. L'objectif était de déterminer lequel de ces chunkers était le mieux adapté aux données dont nous disposions. Pour mener à bien ce test, nous avons suivi la méthodologie dans la figure 3 suivante:

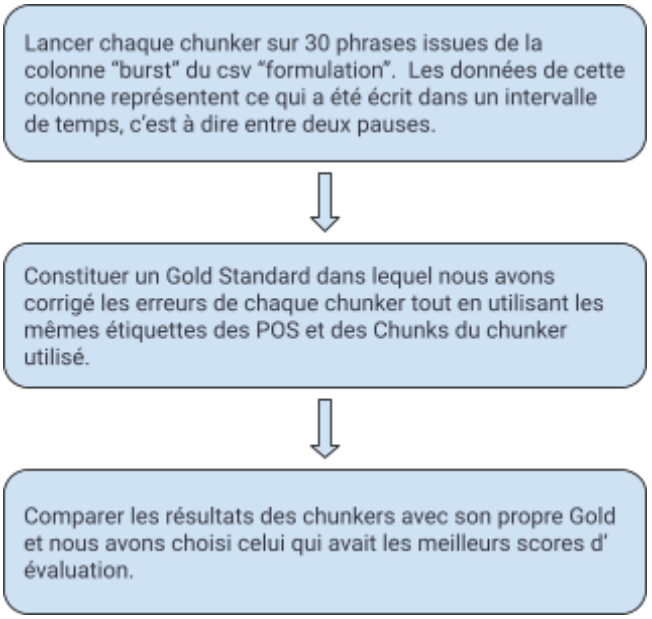


Figure 3 Méthodologie de la recherche

4.1 SEM

SEM est un logiciel développé par Yoann Dupont et Isabelle Tellier. Il permet l'annotation du texte en parties du discours (POS) et effectue également le chunking. Au début, l'outil procède à l'annotation du texte en attribuant des étiquettes de POS à chaque mot. Ensuite, en se basant sur ces annotations, SEM regroupe les mots en chunks, qui sont des unités linguistiques plus grandes et significatives.

SEM était depuis le début notre Saint-Graal. Il s'est avéré être un outil extrêmement efficace et précis pour l'annotation de POS. Forts de ces résultats impressionnants en POS tagging, nous avions de grandes attentes pour l'application de SEM au chunking. Nous étions curieux de savoir comment il traiterait nos données spécifiques et si sa précision se maintiendrait dans cette tâche plus complexe.

4.1.2 Evaluation de SEM

Nous avons évalué les performances du POS tagging et du chunking en utilisant les mesures de rappel, précision et F-mesure. Ces évaluations ont été réalisées sur les 30 premières lignes de données. Les tableaux 1 et 2 ci-dessous présentent respectivement les résultats de l'évaluation de SEM pour le POS tagging et le chunking :

Mesure	**Rappel**	**Précision**	**F-mesure**
POS tagging	98%	94%	96%

Tableau 1 : Résultats de l'évaluation de SEM pour le POS tagging

Mesure	**Rappel**	**Précision**	**F-mesure**
chunking	98%	92%	95%

Tableau 2 : Résultats de l'évaluation de SEM pour le chunking

Il est à noter que les résultats obtenus lors des évaluations du POS tagging et du chunking réalisées avec SEM ont démontré des performances prometteuses.

Pour le POS tagging, SEM a obtenu un rappel de 98% et une précision de 94%, ce qui indique sa capacité à identifier correctement les parties du discours dans les phrases annotées. La F-mesure de 96% souligne un équilibre harmonieux entre le rappel et la précision.

Dans le cas du chunking, SEM a atteint un rappel de 98% et une précision de 92%, montrant ainsi sa capacité à regrouper correctement les mots en chunks et à identifier avec précision les limites de ces groupes. La F-mesure de 95% indique un bon équilibre global entre le rappel et la précision pour cette tâche.

Dans l'ensemble, les résultats obtenus démontrent que SEM est un outil efficace et fiable pour l'annotation et l'analyse linguistique. Il a montré de solides performances à la fois pour le POS tagging et le chunking des données écrites en temps réel. Ces résultats encouragent l'utilisation de SEM dans des applications nécessitant une compréhension précise de la structure linguistique des textes.

4.2 TreeTagger

TreeTagger est un outil développé par Helmut Schmid à l'Université de Stuttgart, qui permet d'annoter le texte en parties du discours (POS) et en lemmes. Il dispose également d'un module de chunking qui peut être appliqué à un fichier texte contenant tous les tokens des phrases que l'on souhaite parser en chunks. Cependant, après avoir comparé les résultats de TreeTagger avec un Gold Standard, nous avons constaté que sa division en chunks n'était pas optimale et ne semblait pas exploitable. Nous avons observé que TreeTagger rencontrait des difficultés avec certains signes de ponctuation et imbriquait les chunks NP dans des chunks PP. En conséquence, nous avons pris la décision de rejeter TreeTagger pour notre étude en raison de ces limitations dans sa performance de chunking.

4.3 SpaCy

SpaCy est une bibliothèque de traitement automatique des langues en Python, développée par Matt Honnibal et Ines Montani. Dans SpaCy, le chunking est généralement réalisé en analysant les phrases sous forme d'arbres de dépendance. Pour obtenir les dépendances et les POS, il est nécessaire de commencer par la tokenisation, puis les annotations. Cependant, nous avons rejeté l'utilisation de SpaCy dans notre étude en raison de certaines limitations dans sa capacité à identifier précisément les limites entre les différentes parties lors de la segmentation des phrases en chunks. Nous avons observé plusieurs erreurs syntaxiques dans les résultats, telles que l'utilisation de chunks sans signification, la combinaison incorrecte de mots dans un chunk, ainsi que des divisions incorrectes de phrases en chunks.

4.4 NLTK

Nous avons rejeté NLTK comme premier choix pour le chunking en raison de sa nécessité de définir manuellement nos propres règles de chunking à l'aide d'expressions régulières basées sur les étiquettes POS. Cette approche manuelle ne nous permettait pas d'effectuer le chunking de manière automatisée, ce qui rendait difficile la collecte de mesures d'évaluation fiables pour NLTK dans notre étude.
Cependant, il convient de noter que NLTK reste un outil précieux pour d'autres tâches de traitement du langage naturel, notamment l'étiquetage des parties du discours (POS tagging). Il est largement utilisé dans la communauté de recherche linguistique en raison de sa flexibilité et de ses fonctionnalités avancées.

4.5 Résumé

Après avoir évalué les performances de plusieurs outils, à savoir SEM, NLTK, SpaCy et TreeTagger, nous avons pris la décision finale de choisir SEM comme l'outil privilégié pour notre étude.

SEM s'est distingué des autres outils en obtenant les meilleures performances dans nos évaluations, tant pour le POS tagging que pour le chunking. Sa capacité à fournir des annotations précises et cohérentes, ainsi que sa facilité d'utilisation et son automatisation dans le processus de chunking, en font l'outil idéal pour nos besoins.

En comparaison, TreeTagger s'est classé en deuxième position en termes de performances. Bien qu'il dispose de fonctionnalités de chunking, ses résultats en termes de précision et de cohérence n'ont pas été satisfaisants. De même, SpaCy a rencontré des difficultés dans l'identification précise des limites des chunks. Quant à NLTK, bien qu'il offre une flexibilité et des fonctionnalités avancées, il présente des limitations en termes de chunking automatisé.

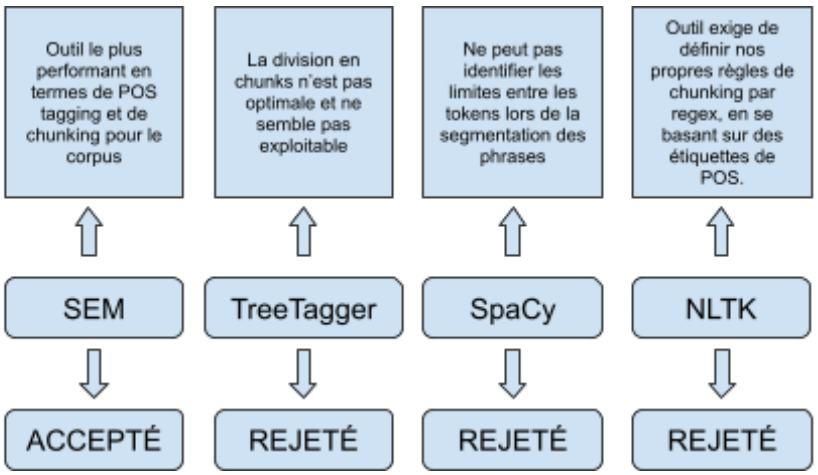


Figure 5 Résumé des résultats du choix du chunker

5. Collecte des données en format exploitable

5.1 Essai numéro 1 : Reconstruction du texte depuis la colonne "charBurst"

Ce premier essai consistait à reconstruire le texte écrit entre chaque intervalle afin de pouvoir analyser le texte ajouté. Pour ce faire, nous avions utilisé la colonne "charBurst" et avions réalisé un programme qui, caractère par caractère, reconstituait le texte.

Il nous a aussi fallu prendre en compte la colonne "startPos", qui indique à quel index se trouve le curseur au début de la séquence de frappe afin d'ajouter le texte produit au bon endroit dans le texte reconstruit.

Néanmoins, cette approche s'est trouvée être infructueuse. Si nous avions la position du curseur au début de chaque séquence, il semble qu'à l'intérieur même des séquences celui-ci se déplace parfois. Ces informations n'étant pas disponibles, le texte reconstruit posait quelques problèmes (mots collés entre eux, mots coupés en deux, etc.) qui rendait difficile son annotation. Ainsi, nous avons abandonné cette méthode.

```
#---TREATMENT---

    outputLines=[]
    outputLine=[]
    outputIndex=0
    correctionIndex=0
    for i,j in enumerate(fileInput):
        if j[charBurstIndex][0]=="\"" and j[charBurstIndex][len(j[charBurstIndex])-1]=="\"":
            fileInput[i][charBurstIndex]=j[charBurstIndex][1:len(j[charBurstIndex])-1].replace("\"\"","\"")
    for i in fileInput:
        toAddIndex=0
        for j in i[charBurstIndex]:
            if j!="⌫":
                while True:
                    if int(i[startPosIndex])+toAddIndex-correctionIndex>len(outputLine):
                        correctionIndex+=1
                    else:
                        break
                outputLine.insert(int(i[startPosIndex])+toAddIndex-correctionIndex,j)
                toAddIndex+=1
            else:
                while True:
                    try:
                        outputLine.pop(int(i[startPosIndex])-1+toAddIndex-correctionIndex)
                        break
                    except:
                        correctionIndex+=1
                toAddIndex-=1
        outputLines.append(outputLine)
        outputLine=outputLines[outputIndex][:]
        outputIndex+=1

    fileOutput=open(f"output_{fileToOpen}.csv","w",encoding="utf-8")
    fileOutput.write("charBurst\twholeTextTyped\n")
    for i,j in zip(fileInput,outputLines):
        fileOutput.write(f"{i[charBurstIndex]}\t{''.join(j).replace('␣',' ')}\n")
    fileOutput.close()
```

5.2 Essai numéro 2 : Les bigrammes de phrases à partir de la colonne “burst”

Une fois compris que la colonne “charBurst” ne contenait pas des données exploitables, nous nous sommes tournés vers la colonne “burst”. La colonne “burst” des csv qui constituent notre corpus contient le texte final de chaque intervalle. Afin de pouvoir chunker chaque ligne et de vérifier si les pauses se trouvent ou pas à la frontière entre deux chunks, nous avons pensé que la constitution de bigrammes aurait pu être une solution. L’idée était de prendre deux phrases successives, les coller ensemble, les diviser en chunk et ensuite insérer le symbole de pause “@” entre eux et vérifier si cette dernière se trouve ou non à la frontière entre deux chunks. Un traitement d’une telle sorte nous permet aussi d’observer si des patterns sont identifiables, c’est-à-dire si des types de chunks particuliers semblent provoquer la pause. Toutefois deux grandes difficultés se sont présentées face à nous :

1) Comment chunker les bigrammes de phrases sur SEM étant données que nous ne disposions que de la version en ligne de SEM ?

2) Comment insérer les pauses une fois obtenus les bigrammes de phrases ?

Un long script python a été écrit à ce propos (nom du script “Bigrams_chunking_and_pausesInsertion.py”). Dans un premier temps, ce script récupère les textes des phrases présentes dans la colonne “burst” et crée des bigrammes qui sont ensuite stockés dans une variable “burst_clean_with_pause” sur la base du modèle suivant :

- Phrase 1 + symbole de pause “@” + Phrase 2
- Phrase 2 + symbole de pause “@” + Phrase 3
- Phrase 3 + symbole de pause “@” + Phrase 4
- Etc…

Le but est d’enregistrer la position de la pause entre deux phrases, c’est-à-dire à l’intérieur de chaque bigramme.
En même temps, une autre liste de bigrammes est créée et stockée dans une variable “burst_clean_no_pause” mais cette fois-ci sans les pauses et encore sur le modèle précédent :

- Phrase 1 + Phrase 2
- Phrase 2 + Phrase 3
- Phrase 3 + Phrase 4
- Etc…

C’est cette deuxième liste qui sera passée, bigramme par bigramme, aux chunker SEM en ligne.

Pour répondre à la question 1 de ce paragraphe, le script passe à SEM le contenu de notre liste de bigrammes sans les pauses. Pour cela, nous avons utilisé la librairie python Selenium qui permet de simuler la navigation humaine sur les sites web. Grâce à une fonction que nous avons appelée “write_chunks_from_bigrams_burst”, nous avons passé en boucle tous les bigrammes de notre liste en obtenant des bigrammes divisé en chunks en output. Les résultats ont été ensuite écrits dans un fichier texte pour chacun des csv dont nous disposons (formulation, planification et révision).

Voici la fonction “write_chunks_from_bigrams_burst” dans sa totalité et les importations nécessaire à son fonctionnement :

```
# Importation de la librairie "Selenium" et des fonctions qui permettent de passer au chunker SEM en ligne chaque
# bigramme.

import selenium
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.wait import WebDriverWait
from selenium.webdriver.common.alert import Alert
from selenium.webdriver.firefox.options import Options
```

```
# Cette fonction écrit ligne par ligne sur un document texte l'output du chunker SEM

def write_chunks_from_bigrams_burst(burst_clean_no_pause):

    # On fait en sorte qu'aucune fenetre de navigation n'apparaisse
    options = Options()
    options.headless = True

    # On fait une boucle sur chaque bigramme contenu dans "burst_clean_no_pause" et on prend en considération que les
    # bigrammes qui contiennent du texte
    for idx,bigram in enumerate(burst_clean_no_pause):
        bigram = str(bigram)
        if bigram == " ":
            with open("../Selenium_SEM_output//bigram_chunks_complete_formulation","a+", encoding="utf-8") as file:
                file.write("\n")
            file.close()
            continue

        else:
            # On ouvre le browser FireFox
            driver = webdriver.Firefox(options=options)

            # On navige vers le site de SEM
            url = "https://apps.lattice.cnrs.fr/sem/index"
            driver.get(url)
            driver.implicitly_wait(5)

            # On accepte les cookies et les fenetres de popup
            driver.switch_to.alert.dismiss()
            time.sleep(2)
            time.sleep(1)

            # On écrit le bigrammes sur la zone de texte et on lance la division en chunks
            driver.find_element(By.XPATH,"//textarea").send_keys(bigram)
            time.sleep(1)
            driver.find_element(By.ID,"inlineCheckbox2").click()
            time.sleep(1)
            driver.find_element(By.XPATH,"//button").click()
            time.sleep(1)
            driver.switch_to.alert.dismiss()
            time.sleep(1)
            driver.find_element(By.XPATH,"//label[contains(.,'Chunking')]").click()
            time.sleep(1)

            # On telecharge le fichier texte conténant le bigramme divisé en chunk
            driver.find_element(By.XPATH,"//a[contains(.,'t text')]").click()
```

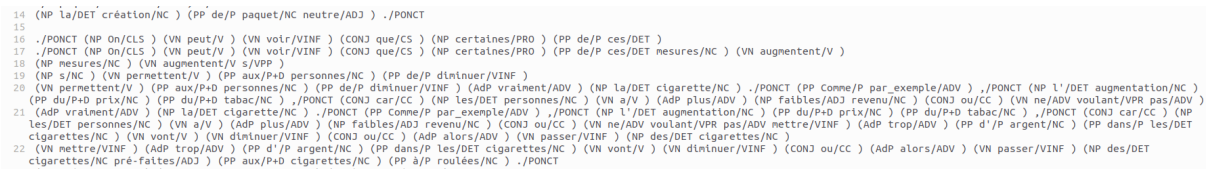
```
            with open("../Selenium_SEM_output//bigram_chunks_complete_formulation","a+", encoding="utf-8") as file:

                # On ouvre le fichier texte télécharge dupis le site de SEM contenant le bigramme, on écrit son conte
                # sur un fichier appelé "bigram_chunks_complete_" et on éfface le fichier téléchargé auparavant
                file_path = glob.glob("/home/diego/Downloads/sem_*.text")
                file_path = str(file_path)[2:len(str(file_path))-2]
                with open(file_path,"r",encoding="utf-8") as f:
                    unique_line = ""
                    for line in f.readlines():
                        line = line.strip('\n')
                        unique_line = unique_line+" "+line
                file.write(unique_line)
                file.write("\n")
                os.remove(file_path)
            file.close()
            driver.close()
```

```
# On lance notre focntion sur nos bigrammes

write_chunks_from_bigrams_burst(burst_clean_no_pause)
```

Le fichier obtenu en output grâce à la fonction ci-dessus est le suivant (aperçu) :



A partir du fichier output en format texte nous avons inséré le symbole de pause “@” via des expressions régulières. Le pattern choisi pour la construction des expressions régulières se base sur la position de l’indice du symbole “@” qui se trouve dans la variable “burst_clean_with_pause” et récupère le mot à gauche et le mot à droite du symbole “@”. Le regex vérifie ensuite si ces deux mots se trouvent bien dans les bigrammes chunkées et ajoute le symbole “@” au milieu. Si le symbole se trouve entre deux parenthèses dont la première fermante (“)”) et la deuxième ouvrante (“(“), alors cela signifie que la pause est à la frontière entre deux chunks. Si ce n’est pas le cas alors le symbole de pause “@” se trouve à l’intérieur d’un chunk.

Voici la partie du code qui s’occupe de l'insertion du symbole de pause “@”. Ce script emploie les fonctions “try” et “except” car dans les corpus sont présents des cas où le regex n’arrive pas immédiatement à identifier les bons mot qui se trouvent à gauche et à droite du symbole de pause “@”.

```
# Insertion du symbole de pause "@" entre le bigrammes divisés en chunks. Les données bruité sont traitées avec un
# enchainement de "try" "except"

with open("../Selenium_SEM_output/bigram_chunks_complete_formulation","r",encoding="utf-8") as f:
    with open("../Selenium_SEM_output/bigram_chunks_complete_formulation_with_pauses","a+",encoding="utf-8") as final
        for idx,line in enumerate(f.readlines()):
            mot_1 = pauses_entre_mot[idx][0]
            mot_2 = pauses_entre_mot[idx][2]
            try:
                match_1 = re.search(mot_1,line)
                match_2 = re.search(mot_2,line)
                set_pause = line[match_1.span()[1]:match_2.span()[0]]
                between_chunks = re.search("\)\s\(",set_pause)
            except:
                try:
                    match_1 = re.search("\\"+mot_1,line)
                    match_2 = re.search(mot_2,line)
                    set_pause = line[match_1.span()[1]:match_2.span()[0]]
                    between_chunks = re.search("\)\s\(",set_pause)
                except:
                    match_1 = re.search('\"',line)
                    match_2 = re.search(mot_2,line)
                    set_pause = line[match_1.span()[1]:match_2.span()[0]]
                    between_chunks = re.search("\)\s\(",set_pause)


            if between_chunks is None:
                try:
                    match_1 = re.search(mot_1,line)
                    match_2 = re.search(mot_2,line)
                    set_pause = line[match_1.span()[1]:match_2.span()[0]+match_1.span()[0]]
                    between_chunks = re.search("\)\s\(",set_pause)
                except:
                    try:
                        match_1 = re.search("\\"+mot_1,line)
                        match_2 = re.search(mot_2,line)
                        set_pause = line[match_1.span()[1]:match_2.span()[0]+match_1.span()[0]]
                        between_chunks = re.search("\)\s\(",set_pause)
                    except:
                        match_1 = re.search('\"',line)
                        match_2 = re.search(mot_2,line)
                        set_pause = line[match_1.span()[1]:match_2.span()[0]+match_1.span()[0]]
                        between_chunks = re.search("\)\s\(",set_pause)
                if between_chunks is None:
                    final_file.write(line[:match_1.span()[1]])
                    final_file.write('@')
                    final_file.write(line[match_1.span()[1]:])
                elif between_chunks[0] == ') (':
                    final_file.write(line[:match_1.span()[1]+1])
                    final_file.write(')@(')
                    final_file.write(line[match_2.span()[0]-3:])
            elif between_chunks[0] == ') (':
                final_file.write(line[:match_1.span()[1]+1])
                final_file.write(')@(')
                final_file.write(line[match_2.span()[0]-3:])
```

Le document finale sur lequel seront calculé ensuite les statistique se présente donc ainsi, avec le symbole de pause “@” inséré au bon indice :

```
1 (NP L'/DET arrêt/NC ) (PP du/P+D tabac/NC ) (VN est/)@(NP un/DET sujet/NC ) (NP qui/PROREL ) (VN est/V ) (AdP  de_plus_en_plus/ADV ) (AdP fréquent/ADV )
2 (NP un/DET sujet/NC ) (NP qui/PROREL ) (VN est/V ) (AdP  de_plus_en_plus/ADV ) (AdP fréquent/)@(PP dans/P les/DET conversations/NC ) (PP en/P France/NPP ) ,/PONCT (PP où/
  PROREL ) (NP encore/ADV )
3 (PP dans/P les/DET conversations/NC ) (PP en/P France/NPP ) ,/PONCT (PP où/PROREL ) (AdP encore/)@(AdP équent/ADV )
4 (NP équent@/NC de_plus_en_plus/ADV ) (NP répandu/VPP )
5 (AdP  de_plus_en_plus/ADV ) (VN répandu/)@(us_en_plus/ADV ) (VN répandu/VPP ) (PP en/P France/NPP ) ./PONCT (NP L/NC )
6 (PP en/P France/NPP ) ./PONCT (NP L@/DET es/NC différentes/ADJ )
```

6. Statistiques

Le corpus-driven linguistique rejette la caractérisation de la linguistique de corpus en tant que méthode et prétend à la place que le corpus lui-même devrait être la seule source de nos

hypothèses sur le langage. On prétend ainsi que le corpus lui-même incarne une théorie du langage[1].

La nature de ce projet ainsi que nos hypothèses et nos questions principales exigeaient de recueillir des preuves des faits linguistiques. Il était donc nécessaire de collecter le plus de preuves possibles sous la forme de nombres et de statistiques des phénomènes observés dans le corpus. Ces grandes masses de données catégorisées et représentées par des nombres exacts nous permettaient de répondre aux questions et formuler des conclusions sur des productions des locuteurs.

Notre but était de collecter le maximum de données statistiques plus ou moins fiables pour répondre aux questions et hypothèses qui nous tracassent. Afin d'automatiser cette partie de recherche au plus possible, nous avons créé des fonctions et des scriptes qui nous ont aidé d'émerger des différentes caractéristiques des productions traitées.
Grâce à l'insertion d'une arobase qui marquent des pauses produites par le locuteur dans le texte traitée, nous obtenons accès facile aux informations concernant leurs emplacement. Dès lors, il est possible d'effectuer l'extraction automatique des chunks (ses étiquettes et ses éléments regroupés) ainsi que d'observer les corrélation entre les chunks et l'emplacement des pauses. Pour repérer les motifs et les statistiques qui sont au coeur de notre analyse du corpus, nous avons créé un script python qui parcourt le texte, crée une liste de chunks, et ensuite en se référant aux arobases renvoi les informations si la pause et produit dedans le chunk ou non, et quelle sont les étiquettes de chunks concernées. Ce script a été appliqué aux trois corpus (formulation, planification et révision) et construit à partir de chaque .csv un tableau de résultat.

6.1. Explication du script

En premier étape de notre script, nous constituons une liste de chunks visibles dans le texte. Les arobases qui se trouvent en extérieur des chunks, c'est-à-dire qui se trouvent entre une balise fermante et une balise ouvrante - ")@(", sont gardées et ajoutées dans une liste comme des éléments indépendants. Ensuite, tous les signés de ponctuation avec ses étiquettes sont supprimés de cette liste, de telle façon qu'elle contient seulement des chunks et des arobases. Cette approche nous permet de traiter des cas, où les pauses sont produites entre les chunks.

[1] (Tognini-Bonelli 2001 : 84-5)

Une arobase en tant qu'élément séparé de la liste sera un indicateur de ce phénomène pour le programme, et la ponctuation, qui dans cette étude n'est pas pertinente, une fois supprimée ne nous empêchera pas d'extraire des étiquettes des chunks précédents ou suivants.

En parcourant la liste de chunks, quand le programme trouve le signé spécial tout seul sous l'index *i*, il récupère l'élément qui le précède avec un index *i*-1, et aussi celui qui le suit avec un index *i*+1. Depuis ces deux chunks, il extrait le premier élément, qui est son étiquette. Si les étiquettes de deux chunks ne sont pas conformes au standard d'étiquetage de chunks par SEM, ils ne sont pas pris dans les étapes suivantes et nous les considérons et les comptons comme des données bruitées. Ensuite, les deux étiquettes conformés sont mises dans un tuple que le programme ajoute à la deuxième liste qui nous sert à rassembler toutes les occurrences de tous les étiquettes de chunks entre lesquels il y a des pauses. À la fin, nous sommes capables de compter automatiquement le nombre de tuples identiques dans cette liste et créer une sorte de dictionnaire qui représente les données recueillies par le programme.

Une méthodologie similaire a été appliquée dans le cas de l'examen des pauses qui sont produites au milieu d'un chunk. Avec une telle différence que tous les éléments d'une liste qui ont une longueur plus petite que 7 caractères sont supprimés, ce qui nous permet de supprimer les éléments arobases indépendants et d'alléger le programme. En parcourant cette liste de chunks nettoyé à nouveau, une fois l'élément qui contient l'arobase est rencontré le programme extrait son étiquette qui doit être conforme au standard de SEM pour être pris en compte dans les statistiques. Le reste de la méthodologie est identique à l'expérience précédente.

6.2 Présentation des données collectées et analyse

Dans cette partie, nous allons présenter les données recueillies en appliquant notre programme aux trois corpus : planification, formulation et révision dont chacun représente un processus de la textualisation. Nous allons les analyser, pour déduire des caractéristiques et des corrélations présentes au sein de nos ressources et les discuter brièvement. Il nous semble nécessaire de souligner que notre programme n'est pas parfait ni la qualité des données. Il était prévu d'avoir des résultats imparfaits avec telle nature des données, alors la recherche se focalise plutôt sur des données plus sûres et propres que d'essayer de minimiser le bruit.

6.2.1 Planification

L'étape de la planification est le moment de la production des idées par des locuteurs et leur organisation dans le texte. Dans les résultats du programme appliqué à ce corpus, il est remarquable qu'il y a une grande partie des données bruitées. Quand même, nous pouvons remarquer des caractéristiques qui émergent.

Dans la plupart des cas, les pauses produits par des locuteurs sont arrivés entre un chunk prépositionnel et un chunk nominal (ex. (PP de/P médecine/NC traditionnelle/ADJ) @ (NP Ce/DET centre/NC), (NP l'/DET éfficacité/NC) @ (PP de/P cette/DET démarche/NC)). Un peu moins entre deux chunks nominaux (ex. (NP un/DET avantage/NC) @ (NP Cette/DET médecine/NC)) ou entre un chunk nominal et un chunk verbal (ex. (NP sa/DET disposition/NC) @ (VN place/V)).

Nature	Nombre	Exemple
PP, NP	164	(PPde/P médecine/NC traditionnelle/ADJ)@(NPCe/DET centre/NC)
NP, NP	164	(NPun/DET avantage/NC)@(NPCette/DET médecine/NC)
NP, VN	118	(NPsa/DET disposition/NC)@(VNplace/V)
NP, PP	118	(NPl'/DET éfficacité/NC)@(PPde/P cette/DET démarche/NC)
VN, NP	109	(VNvoient/V)@(NPun/DET renouveau/NC)
PP, PP	103	(PPdans/P le/DET reacteur/NC)@(PPdes/P+D avions/NC)
PP, VN	47	(PPen/P meilleure/ADJ santé/VPP)@(VNRéduire/VINF)
VN, PP	65	(VNpermetterait/V)@(PPdes/P+D gaz/NC)
AdP, NP	57	(AdP De_plus/ADV)@(NPelle/CLS)
CONJ, NP	50	(CONJque/CS)@(NPtoutes/ADJ les/DET personnes/NC)
VN, VN	39	(VNdevra/V)@(VNêtre/VINF mis/VPP)
AdP, VN	27	(AdPégalement/ADV)@(VNêtre/VINF)
AdP, PP	22	(AdPrégulierement/ADV)@(PPde/P problème/NC)
AP, NP	21	(APfatale/ADJ)@(NPLa/DET médecine/NC alternative/ADJ)
AP, PP	14	(APproches/ADJ)@(PPdes/P+D villes/NC)
CONJ, VN	10	(CONJMais/CC)@(VNne/ADV pensons/V pas/ADV)
AdP, AP	8	(AdPplus/ADV)@(APrapide/ADJ)
VN, AP	5	(VNsont/V)@(APnombreux/ADJ)
CONJ, PP	4	(CONJmais/CC)@(PPdans/P l'/DET ensemble/NC tout/ADJ le/DET monde/NC)
PP, AdP	4	(PPdans/P le/DET froid/NC)@(AdP De_plus/ADV)
NP, CONJ	3	(NPsa/DET prescription/NC)@(CONJEt/CC)
NP, AdP	3	(NPleur/DET réussité/NC scolaire/ADJ)@(AdPréellement/ADV)
AP, VN	2	(APfatale/ADJ)@(NPLa/DET médecine/NC alternative/ADJ)
VN, CONJ	2	(VNprospérer/VINF)@(CONJet/CC)
PP, AP	2	(PPsur/P le/DET long/ADJ terme/NC)@(APnégatifs/ADJ)
NP, AP	2	(NPdes/DET remèdes/@/NC naturels/ADJ)@(APnaturels/ADJ)
PP, CONJ	2	(PPde/P fumer/VINF)@(CONJet/CC)
AP, CONJ	1	(APnécéssitées/ADJ)@(CONJet/CC)
CONJ, AP	1	(CONJque/CS)@(APcertain/ADJ)
AdP, AdP	1	(AdPetc/ADV)@(AdPetc/ADV)
VN, AdP	1	(VNne/ADV fumant/VPR pas/@/ADV pas/ADV)@(AdPpas/ADV)
Bruit	1135	

Tableau 3 : Pauses produits entre les chunks dans le processus de la planification avec des exemples

Concernant les pauses produits dans les chunks, nous pouvons déjà remarquer, en regardant la tableau 4, qu'elles constituent une minorité importante. Il y en a nettement moins que ceux entre les chunks dont la grande majorité, avec un gros avantage quantitatif par rapport aux autres, étaient produits dans les chunks nominaux ou prépositionnels (ex. (NP la/DET mont/@/NC é/ADJ plus/ADV rapide/ADJ , (PP en/@/P sa/DET disposition/NC)).

Nature	Nombre	Exemple
NP	434	(NPla/DET mont/@/NC é/ADJ plus/ADV rapide/ADJ)
PP	372	(PPen/@/P sa/DET disposition/NC)
VN	89	(VNsont/@/V reconnu/VPP)
AdP	47	(AdP Prem_@_ièrement/ADV)
CONJ	25	(CONJMais/CC que/@/CS qu'_est-ce_que/CS)
AP	24	(APpeu@uplé/ADJ)
Bruit	13	

Tableau 4 : Pauses produits dans les chunks dans le processus de la planification avec des exemples

6.2.2. Formulation

L'étape suivante est la formulation où les locuteurs traduisent leurs idées et leurs concepts en langage naturel et les incluent dans le texte. Il est remarquable comme précédemment qu'il y a une grande partie des données bruitées.

Dans la plupart des cas, les pauses produits par des locuteurs sont arrivés entre un chunk prépositionnel et un chunk nominal (ex. (PP depuis/P plusieurs/DET années/) @ (NP le/DET prix/NC), (NP terrasse/) @ (PP pour/P fumeur/NC)). Un peu moins entre deux chunks nominaux (ex. (NP une/DET voiture/) @ (NP qui/PROREL)) ou entre un chunk nominal et un chunk verbal (ex. (NP curité/NC routière/) @ (VN est/V)). Il convient de souligner qu'il s'agit des mêmes groupes majoritaires que dans l'analyse précédente, et de plus, les nombres représentant les occurrences de ces phénomènes sont presque identiques (une longueur de corpus est très proches aussi).

Nature	Nombre	Exemple
PP, NP	136	(PPdepuis/P plusieurs/DET années/NC)@(NPle/DET prix/NC)
NP, NP	130	(NPune/DET voiture/NC)@(NPqui/PROREL)
NP, VN	114	(NPcurité/NC routière/ADJ)@(VNest/V)
NP, PP	95	(NPune/DET diminution/NC)@(PPdans/P) ; (NPterrasse/NC)@(PPpour/P) fumeur/NC)
PP, PP	92	(PPde/P diminution/NC)@(PPdans/P l'/DET achat/NC)
VN, NP	89	(VNcrée/VPP)@(NPdifférentes/DET mesures/NC)
PP, VN	66	(PPdes/P+D paquets/NC)@(VNprofitent/V)
VN, PP	55	(VNest/V)@(PPà/P cause/NC)
AdP, NP	47	(AdPencore/ADV)@(NPla/DET création/NC)
CNOJ, NP	44	(CONJMais/CC)@(NPla/DET création/NC)
PP, CONJ	44	(PPde/P cigarettes/NC)@(CONJou/CC)
NP, CONJ	35	(NP150/DET ans/NC)@(CONJque/CS)
VN, AdP	30	(VNa/V)@(AdPmême/ADV)
VN, VN	26	(VNdéplacer/VINF)@(VNest/V)
PP, AdP	24	(PP jusqu'_à/P devenir/VINF)@(AdPquasiment/ADV)
AdP, PP	23	(AdPfréquent/ADV)@(PPdans/P les/DET conversations/NC)
NP, AdP	22	(NPdes/DET maladies/NC pulmonaires/ADJ)@(AdPAinsi/ADV)
VN, CONJ	19	(VNobstine/V)@(CONJquand/CS)
AdP, AdP	19	(AdPSeulement/ADV)@(AdPCependant/ADV)
VN, AP	18	(VNserait/V)@(APlong/ADJ)
AP, NP	16	(APintéressant/ADJ)@(NPla/DET sécurité/NC routière/ADJ)
AP, PP	16	(APgratuit/ADJ)@(PPaux/P+D étudiants/NC)
AdP, VN	13	(AdP en_effet/ADV)@(VNdiminuer/VINF)
AP, CONJ	13	(APviolent/ADJ)@(CONJet/CC)
AdP, CONJ	10	(AdPvite/ADJ)@(CONJet/CC)
CONJ, VN	9	(CONJque/CS)@(VNéradiquer/VINF)
CONJ, AdP	8	(CONJsi/CS)@(AdPCependant/ADV)
AdP, AP	6	(AdPaussi/ADV)@(APindisponsable/ADJ)
CONJ, PP	6	(CONJcar/CC)@(PPde/P bus/NC)
AP, VN	4	(APmoindres/ADJ)@(VNSont/V)
NP, AP	3	(NPla/DET population/NC)@(APles/DET plus/ADV pauvre/ADJ)
AP, AdP	3	(APélevé/ADJ)@(AdPCependant/ADV)
CONJ, CONJ	2	(CONJou/CC)@(CONJou/CC)
PP, AP	1	PP d'/P être/VINF)@(APrespectueux/ADJ)
AP, AP	1	(APtous/ADJ)@(APtous/ADJ)
Bruit	811	"(PPde/P tabac/NC)@(P)

Tableau 5 : Pauses produits entre les chunks dans le processus de la formulation avec des exemples

Concernant les pauses produites dans les chunks, nous remarquons dans le tableau 6, qu'elles constituent toujours une minorité importante par rapport à celles qui sont produites entre les chunks. Dans ce processus, les pauses étaient produites le plus souvent dans les chunks nominaux ou les chunks prépositionnels (ex. (NP L/@/DET es/NC différentes/ADJ), (PP à_cause_@_de/P la/DET vitesse/NC)).

Nature	Nombre	Exemple
NP	369	(NPL/@/DET es/NC différentes/ADJ)
PP	329	(PPà/@/P Dans/NPP un/DET)
VN	64	(VN il_@_y_a/V)
AdP	57	(AdP à_@_moins/ADV)
CONJ	17	(CONJ ou_même/@/CC qu'/CS)
AP	25	(APcour@rante/ADJ)
Bruit	7	

Tableau 6 : Pauses produits dans les chunks dans le processus de la formulation avec des exemples

6.2.3 Révision

Le dernier processus de la textualisation est la révision où les locuteurs peuvent modifier soit des éléments du texte soit des concepts ou leurs organisations. Dans cette partie, toutes nos observations précédentes ont été confirmées à nouveau par des chiffres.

Les pauses produits par des locuteurs entre les chunks arrivent régulièrement, selon le schéma entre un chunk prépositionnel et un chunk nominal (ex. (PP de/P filtres/NC) @ (NP Cette/DET idée/NC), (NP la/DET fumée/NC) @ (PP Pour/P réduire/VINF)). Un peu moins entre deux chunks nominaux (ex. (NP quelques/DET années/NC) @ (NP Cette/DET hausse/NC)). Dans cette expérience, cependant, nous voyons une tendance beaucoup plus fréquente à produire des pauses entre un chunk nominal et un chunk verbal (ex. (NP les/DET fabriquants/NC) @ (VN ont/V décidé/VPP), (VN fument/V) @ (NP des/DET cigarettes/NC)).

Nature	Nombre	Exemple
PP, NP	147	(PPde/P filtres/NC)@(NPCette/DET idée/NC)
NP, NP	143	(NPquelques/DET années/NC)@(NPCette/DET hausse/NC)
NP, VN	134	(NPles/DET fabriquants/NC)@(VNont/V décidé/VPP)
VN, NP	119	(VNfument/V)@(NPdes/DET cigarettes/NC)
NP, PP	109	(NPla/DET fumée/NC)@(PPPour/P réduire/VINF)
PP, PP	107	(PPde/P morts/NC)@(PPsur/P les/DET routes/NC)
PP, VN	63	(PPcomme/P les/DET étudiants/NC)@(VNont/V)
CONJ, NP	58	(CONJet/CC)@(NPLes/DET transports/NC)
AdP, NP	55	(AdPencore/ADV)@(NPla/DET mise/NC)
VN, PP	54	(VNpermet/V)@(PPaux/P+D non-fumeurs/NC)
VN, VN	28	(VNpourraient/V)@(VNcontinuer/VINF)
AdP, PP	24	(AdPici/ADV)@(PPde/P discuter/VINF)
AdP, VN	23	(AdPrégulièrement/ADV)@(VNdiffusées/VPP)
AP, PP	10	(APlibre/ADJ)@(PPde/P toutes/DET restrictions/NC)
AP, NP	10	(APgratuit/ADJ)@(NPles/DET bus/NC)
CONJ, VN	9	(CONJet/CC)@(VNne/ADV mettrait/V pas/ADV)
CONJ, PP	9	(CONJcar/CC)@(PPen/P les/DET exposant/NC)
VN, AP	8	(VNparaître/VINF)@(APutopiste/ADJ)
PP, AP	5	(PPde/P problèmes/NC)@(APcher/ADJ)
AP, VN	3	(APjudicieux/ADJ)@(VNdemander/VINF)
CONJ, AP	3	(CONJet/CC)@(APcondamnable/ADJ)
PP, CONJ	2	(PP au_courant_des/P risques/NC encourus/ADJ)@(CONJpourquoi/CS)
AdP, AdP	2	(AdPnotamment/ADV)@(AdPnotamment/ADV)
AdP, AP	2	(AdPexclus/ADV)@(APNombreux/ADJ)
AP, AP	2	(APsocial/ADJ)@(APsocial/ADJ)
PP, AdP	2	(PPen/P place/NC)@(AdP en_place/ADV)
VN, CONJ	2	(VNrecherché/VPP)@(CONJet/CC)
AP, AdP	1	(APdangereux/ADJ)@(AdPpeut-être/ADV)
NP, CONJ	1	(NPsa/DET sécurité/NC intérieure/ADJ)@(CONJet/CC)
NP, AP	1	(NPLa/DET loi/NC)@(APla/DET plus/ADV récente/ADJ)
VN, AdP	1	(VNapparaitre/VINF)@(AdP Aujourd'_hui/ADV)
AP, CONJ	1	(APdévastateur/ADJ)@(CONJmais/CC)
NP, AdP	1	(NPles/DET)@(AdP C'_est_pourquoi/ADV)
Bruit	1114	

Tableau 7 : Pauses produits entre les chunks dans le processus de la révision avec des exemples

Les pauses produites dans les chunks ont toujours la même tendance très forte à apparaître dans les chunks nominaux ou les chunks prépositionnels (ex. (NP L/@/DET es/NC différentes/ADJ), (PP à_cause_@_de/P la/DET vitesse/NC)) et elles sont aussi beaucoup moins fréquentes que celles produites entre les chunks.

Nature	Nombre	Exemple
NP	389	(NPleur/@/DET habitude/NC)
PP	363	(PPMalgré/@/P les/DET bonnes/ADJ volontées/NC)
VN	96	(VNpourr/@/V aient/VS être/VINF)
AdP	51	(AdP à_premiere_@_vue/ADV)
AP	25	(APr/@/DET plus/ADV élevé/ADJ)
CONJ	17	(CONJMais/@/CC si/CS)
Bruit	22	

Tableau 8 : Pauses produits dans les chunks dans le processus de la révision avec des exemples

6.2.4 Analyse globale et synthèse

En rassemblant les trois corpus, nous voyons en effet les caractéristiques qui se répètent dans les trois expériences. Il est significatif que 6607 pauses se situaient effectivement aux frontières des chunks, tandis que 2835 pauses était située à l'intérieur des chunks. Nous constatons donc une nette prédominance de la tendance à produire des pauses entre les chunks dans les proportions 3:1 avec celles à l'intérieur des chunks.

Statistiques	Tableau “formulation”	Tableau “planification”	Tableau “révision”
Pause entre chunks	2050	2304	2253
Pause à l’intérieur des chunks	868	1004	963

Tableau 9 : Nombre des pauses produits selon leurs emplacement et le processus de la textualisation

À chaque étape de la textualisation, nous pouvons observer que des pauses sont régulièrement produites entre un chunk prépositionnel et un chunk nominal, deux chunks nominaux ou un chunk nominal et un chunk verbal, ainsi que elles sont très souvent produites avant ou après un chunk nominal.

Si les pauses sont produites dans le chunk, il est extrêmement probable que les locuteurs le feront soit dans un chunk nominal, soit dans un chunk verbal. Les pauses dans d'autres groupes se produisent très rarement et il y a une disproportion surprenante entre les pauses

dans les chunks mentionnés et les autres, ce qui donne des preuves tangibles pour confirmer nos hypothèses.

7. Conclusion

Notre évaluation des chunkers a révélé que SEM était l'outil le plus performant en termes de POS tagging et de chunking pour le corpus de données enregistrées en temps réel. SEM a réussi à segmenter les données de manière plus précise et cohérente que les autres outils tels que TreeTagger, SpaCy et NLTK.

De plus, nos observations ont montré que la plupart des pauses se trouvent aux frontières des chunks, ce qui confirme leur rôle essentiel dans la délimitation des unités linguistiques. Les pauses sont généralement présentes entre les chunks PP et NP, NP et NP, NP et VN, et sont fréquemment positionnées avant ou après les chunks NP. Ces constatations soulignent l'importance des pauses en tant que marqueurs pour marquer les transitions entre différentes structures grammaticales et fournir des indices précieux pour la segmentation du texte.

Cette étude souligne donc l'importance d'utiliser des outils performants tels que SEM pour une analyse linguistique précise. Les résultats obtenus renforcent notre compréhension de la structure linguistique des textes et ouvrent la voie à des recherches futures dans le domaine du traitement automatique des langues.

Il convient de noter que bien que SEM ait présenté des performances prometteuses, des améliorations supplémentaires peuvent être apportées pour une segmentation encore plus précise et cohérente. Les recherches futures pourraient se concentrer sur l'affinement des règles de chunking et l'exploration de nouvelles approches pour prendre en compte les spécificités du langage naturel.

En outre, nos observations ont révélé que les différents processus de textualisation, tels que la formulation, la planification et la révision, présentent des corrélations similaires entre les pauses et les chunks. Dans chaque processus, les pauses se manifestent généralement aux frontières des chunks, ce qui indique une cohérence dans l'utilisation des pauses pour marquer les transitions entre les différentes parties du texte.

Ces constatations suggèrent que les pauses jouent un rôle important dans la structuration du texte, quel que soit le processus de textualisation utilisé. Elles servent de repères pour indiquer les limites entre les chunks et facilitent ainsi la compréhension et l'interprétation du texte.