

# Rapport Projet Chunker

Qinyue Liu M1 IDL

## 1. Structure des fichiers

Fichiers de données:

- dict.csv : contient les mot-clés de chaque catégorie
- regle.csv: contient les règles pour construire les chunks
- source.txt: contient les textes entrées
- tokens.txt: contient les tokens fait par spacy

Fichier de sortie:

- chunker.txt: contient les chunks créés par les codes en python

Fichier de traitement:

- utils.py: contient les fonctions pour la tokenisation etc
- main.py: contient l'algorithme de base pour réaliser le chunker

## 2. Principe de mon algorithme:

- Partie tokenisation:

1. Lire le fichier source.txt, et importer le libraire “*Spacy*”
2. Tokenizer le texte par *Spacy* et sauvegarder les tokens dans fichier “*tokens.txt*”
3. Lire le fichier “*tokens.txt*” et sauvegarder les tokens dans une variable dict

- Partie chunker:

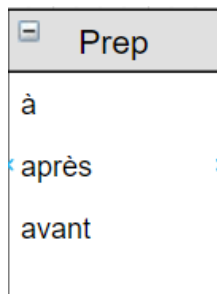
1. Lire les fichiers csv, et fournir deux dictionnaires: “category” et “regle”

Dans category, la clé est le nom de la catégorie et la valeur( les mots de cette catégorie ) est une liste.

{'prep': ['à', 'après', 'avant',...], 'det': ['le', 'la', 'les',...], ...etc.}

Dans regle, la clé est le nom de la catégorie de token précédent, et les valeurs sont aussi les dictionnaires, avec les catégories possibles de faire un chunk comme la clé, et les catégories des chunks comme la valeur.

{'prep': {'prep': 'NULL', 'det': 'PN', 'pro': 'PN', 'pref': 'PV', 'default': 'PN'}, 'det': {'prep': 'SN', 'det': 'N', 'pro': 'NULL', 'pref': 'NULL', 'default': 'N'}, etc.



Prep	
à	
après	
avant	

Prep	
Prep	NULL
Det	PN
Pro	PN
Pref	PV

Le règle de “prep” est la suivante :

Token-avant    token-précédent    ->    catégorie de chunk

Prep    prep    ->    NULL

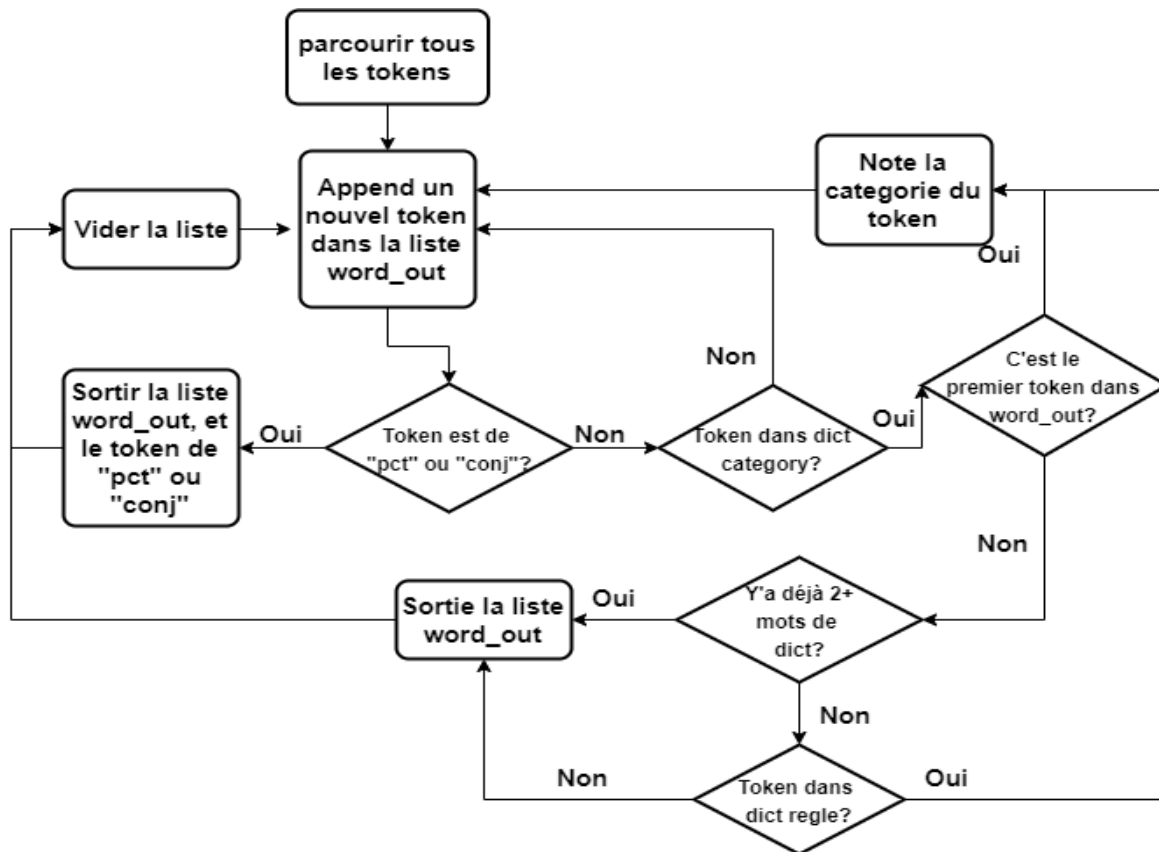
Prep    det    ->    PN

Prep    pro    ->    PN

Prep    pref    ->    PV

Prep    ->    PN

2. Parcourir tous les tokens, et faire les chunks selon la logique suivante:  
("word\_out" contient les tokens pour construire un chunk)



- Partie règle :

Quand il faut sortir le chunk :

1. Quand le token est dans catégorie de "pct" ou "conj".
2. Quand il y a plus de 2 tokens qui se trouvent dans le dictionnaire category.
3. Quand la catégorie du token précédent ne peut pas être trouvée dans dictionnaire regle, cela veut dire:  
list\_word [previous\_key][key] == NULL  
S'il y a plus qu'une relation entre les mots dans le dictionnaire, je prends au final seulement la première relation.

Par exemple:

**"Tout le monde en france"**

Je prends seulement la relation entre *Tout* et *le* pour définir la catégorie de ce chunker.

Dans le fichier "regle.csv": la colonne *index* inclut les catégories qui peuvent suivre la catégorie "prep".

index	prep
prep	
det	PN
pro	PN
pref	PV
sujet	
default	PN

Prep	prep	->	NULL
Prep	det	->	PN
Prep	pro	->	PN
Prep	pref	->	PV
Prep		->	PN

Les limites de mes règles et les erreurs faites par mon chunker:

Ma règle ne peut pas bien distinguer PV et PN, SN et SV, car je n'ai aucune catégorie de verbe dans mon dictionnaire, et pour chaque catégorie, il y a une valeur par défaut, ce qui affecte aussi la précision.

puis -> CONJ

d' ajouter -> PN,

Ceci est un exemple qui montre que PV est confondu avec PN

Certains chunks sont un peu longs car le nombre de mots dans le dictionnaire est limité, par conséquent certaines collocations de phrase n'ont pas été bien reconnues.

ceux qui ne bossent pas autant que moi -> SV,

Ceci est un exemple d'échec de reconnaissance de "autant que"

A cause de la règle où j'ai limité le nombre de mots ( dans le dictionnaire ) dans une phrase, certains chunks ont été mal séparés.

l' instant que de ->SN

l' ordre d' un euro -> SN,

Ceci est un exemple de mauvaise séparation à cause de la règle pour limiter le nombre de mots

La qualité d'analyse fait par mon chunker:  
 Pour le fichier **chunker.txt**: (fichier en français)

	PN	PV	SN	SV	N	V	CONJ
PN	24	1		3	1		
PV		0					
SN			2	1			
SV	1			8			
N				7	9		1
V						3	
CONJ							9

Nombre de Chunks dans le texte: 109  
 Nombre de Chunks trouvés par le chunker: 100  
 Chunks bien catégorisés: 85

La précision pour chunker.txt est  $85/109 = 77.98\%$ , mais la précision pour la catégorie SV est seulement  $8 / (3 + 1 + 8 + 7) = 8/19 = 42.1\%$ , ceci s'explique car il n'y a aucun traitement pour les verbes.

La qualité d'analyse fait par mon chunker:  
 Pour le fichier **chunker1.txt**: (fichier en français)

	PN	PV	SN	SV	N	V	CONJ
PN	9			1	1	1	
PV		0					
SN	1		3				
SV				0			
N	1				2	1	
V						0	
CONJ							1

Nombre de chunks dans le texte: 37  
 Nombre de chunks trouvés par le chunker: 35

Chunks bien catégorisés: 26

La précision pour chunker1.txt est  $26/35 = 74.28\%$ , ce qui ressemble au résultat de chunker.txt, la précision pour les verbes est toujours plus basse.

La qualité d'analyse fait par mon chunker:

Pour le fichier **chunker\_en.txt**: (fichier en anglais)

	PN	PV	SN	SV	N	V	CONJ
PN	8			1			
PV		1					
SN			1		3		
SV	0			1			
N	2				6	1	1
V						0	
CONJ							1

Nombre de chunks dans le texte: 36

Nombre de chunks trouvés par le chunker: 35

Chunks bien catégorisés: 25

La précision pour chunker\_en.txt est  $25/35 = 71.4\%$ , pour l'anglais, la catégorie N est souvent confondue avec SN, c'est à cause des différences des grammaires entre deux langues.

## Annexe:

### jeu de règles

Prep	prep	->	NULL	Det	prep	->	SN
Prep	det	->	PN	Det	det	->	N
Prep	pro	->	PN	Det	pro	->	NULL
Prep	pref	->	PV	Det	pref	->	NULL
Prep	sujet	->	PV	Det	sujet	->	NULL
Prep		->	PN	Det		->	N

Pro	prep	->	NULL
Pro	det	->	SV
Pro	pro	->	NULL
Pro	pref	->	SV
Pro	sujet	->	NULL
Pro		->	SV

Pref	prep	->	NULL
Pref	det	->	NULL
Pref	pro	->	NULL
Pref	pref	->	NULL
Pref	sujet	->	NULL
Pref		->	NULL

Sujet	prep	->	NULL
Sujet	det	->	NULL
Sujet	pro	->	NULL
Sujet	pref	->	NULL
Sujet	sujet	->	NULL
Sujet		->	NULL