

To Build a Chunker

[Sous-titre du document]



JEREMY BOURDILLAT

M1 Sciences du Langage parcours Industries de la Langue

Table des matières

[Introduction : Les *chunks*, une réalité psycholinguistique 2](#_Toc134541674)

[Protocole : Texte de référence et *Gold Standard* 4](#_Toc134541675)

[Protocole et objectifs 4](#_Toc134541676)

[Texte de référence 4](#_Toc134541677)

[Evaluation 4](#_Toc134541678)

[Base de connaissances et fonctionnement général 5](#_Toc134541679)

[Mises en place du moteur 6](#_Toc134541680)

[Implémentation « naïve » / aveugle 6](#_Toc134541681)

[Implémentation à fenêtre glissante *(perdue à tout jamais…)* 7](#_Toc134541682)

[Le cas des expressions régulières 7](#_Toc134541683)

[Annexes 9](#_Toc134541684)

[Texte de référence 9](#_Toc134541685)

[Segmentation de référence 10](#_Toc134541686)

[Bibliographie 12](#_Toc134541687)

# Introduction : Les *chunks*, une réalité psycholinguistique

« *I begin with an intuition* ». Tels étaient les premiers mots d’Abney, non à sa naissance (encore que) mais dans son article Parsing By Chunks de 1991. L’intuition à laquelle se réfère l’universitaire est une intuition orale, celle d’un schéma prosodique ressenti par le locuteur (Abney, 1991, p. 1). Ce schéma prosodique est, de plus, spécifique à l’anglais puisque basé sur son système d’accentuation (un accent lexical par chunk, et seuls les mots lexicaux portent l’accent prosodique en neutral tonicity).

Un chunk est donc un segment ; le plus souvent, il s’agira d’une segmentation de la phrase, c’est-à-dire un découpage des unités dans les limites de cette dernière. Abney déjà les décrivait comme une unité ayant une structure et s’inscrivant dans une structure plus grande : « *I assume that a chunk has syntactic structure which comprises a connected subgraph of the sentence’s parse-tree* »(Abney, 1991, p. 2). Blache les décrit comme des structures partielles assimilables à des structures syntaxiques mais décrites différemment : il considère un syntagme comme un ensemble de propriétés qui, sous une certaine combinaison, activent une catégorie en particulier plus qu’une autre (Blache, 2013, p. 231). Il ne s’agit donc pas, d’après lui, de les décrire avec des liens syntaxiques traditionnels.

Or, le découpage en segments est presque toujours effectué sur de la langue écrite ou bien transcrite ; et dans le cas de l’étude du français, dont le système accentuel est différent de celui de l’anglais, doit-on abandonner pour autant cette idée ?

Se pose alors la question de la légitimité d’un tel formalisme dans ce cadre-ci.   
Il s’avère que des études subséquentes du début du millénaire tendent à prouver la validité d’un même phénomène à l’écrit, en relevant que, dans l’élaboration de théories cognitivistes, le découpage en chunks est intéressant d’un point de vue *mémoire* car il permet son optimisation dans le traitement cognitif de la langue (Blache, 2013, p. 231), sans distinction écrit/oral, via un système de mémoires tampons. D’ailleurs, en sciences cognitives, le *chunk* est une « unité d’information » qui peut être gardée en mémoire à court terme, avec une moyenne définie par George Miller à 7 buffers pour un être humain, appelé « empan mnésique » (Weil-Barais, 2011, p. 343). On peut alors supposer un isomorphisme entre le *chunk* cognitif, et le *chunk* linguistique. Le cognitivisme étant une approche de la pensée comme traitement de l’information, il n’est pas ridicule de stipuler un lien entre informatique, cognition et langage.

Cependant, l’utilité d’un traitement automatique par chunk n’apparaît pas tout de suite clairement : si effectivement l’être humain pourrait fonctionner de cette manière, où réside l’intérêt dans le fait de l’imiter ?

On peut trouver un début de réponse à cette question en relevant le fait que, dans l’hypothèse stipulée par Philippe Blache d’un « traitement incrémental du langage » (Blache, 2013, p. 230), on peut mesurer la quantité d’information linguistique apportée par un mot via les différents traits sémantiques, syntactiques, ou autres qu’il porte, restreignant ainsi « l’espace de recherche », soit la combinatoire de ce qui peut suivre. Sous cet angle de théorie de l’information, on se rapproche ainsi des travaux de Blache sur les grammaires de propriétés et l’intérêt de fonctionner par contraintes. Tout prend alors sens : en découpant le texte en unités plus petites ayant une charge sémantico-syntaxique (bref, de l’information quantifiable, les *propriétés* (Blache, 2001, p. 107), on réduit les chemins possibles, et donc l’ambiguïté. De plus, l’apport d’informations et la combinaison de ces dernières peut permettre d’appliquer de nouvelles contraintes : un déterminant peut dénoter un syntagme nominal (ou ce qui s’en rapproche), or s’il est suivi d’une préposition (donc possible syntagme prépositionnel) on peut émettre l’hypothèse d’une relation « complément du nom » entre les deux (entre autres). En suivant la théorie de Blache, obtenir un nom permet de satisfaire des propriétés de constituance, d’obligation et d’unicité vis-à-vis d’un potentiel syntagme nominal. Nous y reviendrons, mais en définissant une hiérarchie claire des possibles (et donc un choix évident, fût-il correct ou non) cela permet une certaine robustesse en écartant la combinatoire, et permet aussi un traitement par point fixe (*fixed-point computation*) (Cooper, 2023, p. 53) souvent utilisé dans les moteurs d’inférence et a fortiori, les compilateurs.

Il est à noter que la théorie de Blache va encore plus loin : ce dernier rapporte le nombre de propriétés présentes au nombre de celles possibles, dites « pertinentes », faisant de son formalisme un hybride de système expert et statistique (Blache, 2013, p. 237).

**Note : la plupart des documents sont disponibles sur le** [**github du projet**](https://github.com/Jerem-dY/T-GeStaMESP)**.** Dans ce dossier, toute référence à un fichier indique la présence de ce dernier dans le dépôt (qui n’est pas très bien rangé, d’ailleurs…).

# Protocole : Texte de référence et *Gold Standard*

## Protocole et objectifs

Nous souhaitons donc réaliser une ébauche de chunker, soit un système qui va ordonner des unités contigües en groupes, avec :

* Pour unités, des **tokens** en français
* Pour règle générale de groupement, une approximation des **syntagmes** de la grammaire traditionnelle

On souhaite aussi construire une structure arborescente, assez proche d’une analyse en constituants, dont la sortie lisible éventuelle sera au format XML.

Il s’agira d’un **moteur d’inférence**, c’est-à-dire un système qui s’appuie sur la logique de premier ordre (notions de variables mises en relation par des prédicats) pour prendre des décisions sur la segmentation et la catégorisation des chunks. Notre ordinateur étant une machine logique avant tout, cela en est d’autant plus facile à implémenter.

Un chunk sera introduit par un token identifiable qui s’avérera la plupart du temps être un mot grammatical, suivant les observations faites sur le nombre de fixations oculaires de lecteurs sur des mots et à « l’indice de surprise » de Hale (Blache, 2013, p. 232).

## Texte de référence

Pour à la fois se donner une direction à prendre et évaluer les performances de nos essais d’implémentation, nous avons choisi un texte de référence (p.5) et défini une segmentation de référence (p.6, seulement au premier niveau), disponibles en annexe.

## Evaluation

Les résultats sont évalués à la main, en comparant chaque chunk sorti par le programme avec la référence. Une note est alors attribuée, de 0 à 2 avec 0 : insatisfaisant et 2 : très satisfaisant. Sont évalués le type de chunk et sa constitution (quels tokens ont ou n’ont pas été ajoutés) ; il est possible d’avoir un résultat satisfaisant même si différent quand le chunk résultant porte une charge sémantique ou syntaxique suffisante, à l’appréciation du validateur (une jolie façon de dire que j’ai fait au *feeling*).

# Base de connaissances et fonctionnement général

Tout moteur d’inférence possède une base de connaissances, puisque c’est ce qui lui permet de tirer des conclusions. Les raisonnements effectués seront des déductions ayant pour prémisse un token, et comme conclusion la catégorie du chunk ayant pour premier élément ce token ; dans deux des trois implémentations présentées, la base de règle prendra la forme d’une table de hachage associant la prémisse à la conclusion.

Le fait de s’appuyer sur une base de règles mise en place par quelqu’un d’éduqué sur le sujet fait du système un « système expert ».

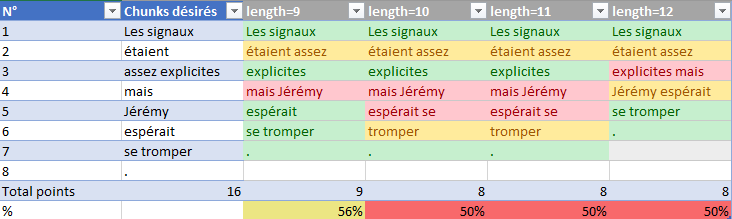
# Mises en place du moteur

Nous allons maintenant évoquer les trois implémentations du moteur : avantages, inconvénients, résultats.

## Implémentation « naïve » / aveugle

Pour ce premier essai, nous avons tenté de segmenter **sans base de règles** et **sans tokenisation** : on définit un « centre » *n* (en nombre de caractères), si l’on tombe sur un caractère de token on avance jusqu’au prochain séparateur, puis on coupe.

Evidemment, les résultats sont plutôt aléatoires, même en faisant varier *n*. Passer par un *n* dynamique ne change pas vraiment la donne. Nous n’allons pas épiloguer dessus, mais cela permet d’évaluer l’importance de la base de connaissance.



Puisque les résultats ne sont ni prévisibles, ni liés à une quelconque sémantique ou syntaxe, ils sont inutilisables pour le traitement. Passer par une base de règles résout ce problème et donne un *sens* aux chunks, une ***charge***, utilisable dans les traitements en aval.

## Implémentation à fenêtre glissante *(perdue à tout jamais…)*

Pour découper en chunks, l’une des (nombreuses) possibilités est de mettre en place une fenêtre glissante à empan variable. Cette fenêtre, par exemple d’une taille de trois, va se déplacer, tenter l’application d’une règle composite sur les trois éléments sélectionnés : si ça marche, alors les trois vont constituer un chunk. Autrement, on essaye à nouveau en réduisant la fenêtre à deux, puis un.

Exemple :

*Soient*

|  |  |
| --- | --- |
| JOINT | [que] |
| SV | [cela faisait partie] |
| N | [des symptômes] |

Une fois le curseur sur JOINT, on peut prendre JOINT+SV+N (encodés de cette manière) et voir que, effectivement, il existe une règle JOINT+SV+N 🡺 PROP.

Le deuxième essai de ce système fonctionnait « en arrière » :

Essai 2 : oo[ooooooI] ; ooo[oooooI] ; oooo[ooooI] ... oooooooo[I]  
Essai 3 : oo[Ioooooooo]oo ; oo[Iooooooo]ooo ... oo[I]ooooooooo

Avec ‘I’ le token sur lequel se trouve le curseur. En tous les cas, l’intérêt est de donner la priorité aux règles composite, plus intéressantes que les règles unaires car plus chargées.

Les règles sont issues d’un filtrage réalisé au fur-et-à-mesure des essais, selon deux critères :

* Les **occurrences** de la règle lors des tests (une règle peu ou pas utilisée peut être supprimée), en passant par des n-grammes (p.12) sur la liste d’application des règles.
* La **cohésion sémantico-syntaxique**, ou l’aspect « instinctif » de ce qui devrait aller ensemble au niveau psycholinguistique. Nous pourrions l’ignorer et ne s’appuyer que sur l’aspect quantitatif, ce qui ferait de notre moteur un système partiellement statistique. Cela conviendrait à un système d’apprentissage automatique, particulièrement non-supervisé, ce qui est hors de la portée de ce dossier.

### Le cas des expressions régulières

Ces essais permettaient l’utilisation de regex sur les éléments en tant que « règles spéciales ». Si cela fonctionnait plutôt bien, il n’est évidemment pas souhaitable d’offrir à l’utilisateur (peu fiable, il faut le dire) la possibilité de ralentir le système pour ensuite venir se plaindre des performances de ce dernier. En revanche, cela nous permet d’intégrer des traits morphologiques à la base de règles, ce qui est intéressant. La conception d’un subset d’expression régulière, à la manière de SQL, serait une solution envisageable pour garder cet outil puissant sans le coût important qu’il peut occasionner. Reste le problème des empans : que celui-ci soit manuel (décidé par l’utilisateur) ou automatique (deviné par le système) – les deux ayant été implémentés – il est impossible pour le système de savoir à l’avance l’empan nécessaire pour telle ou telle expression régulière. Ainsi, le seul moyen d’être certain de la justesse du fonctionnement est de préparer un empan suffisamment grand, ce qui réduit évidemment les performances de l’outil.

Ensuite, un certain nombre de problèmes persistent :

### Résultats

Le premier essai, prototype sans prétention, n’obtenait que 74% des points (p.11). Après avoir raffiné les règles et le fonctionnement du moteur, nous montons à 93% (p.12) ce qui est bien meilleur. Les améliorations ne permettent cependant pas de se débarrasser du problème récurrent avec ce type de système : les agrégats fautifs. Il peut s’agir, par exemple, de la concaténation d’une ponctuation avec un mot non reconnu et donc n’ouvrant pas de chunk, qui se retrouve alors mis avec ladite ponctuation. Pallier à ce problème n’est pas aisé, car cela nécessiterait la création d’un chunk sans en connaître le type. Cela nous pousse donc à remettre en question le modèle que l’on a choisit de suivre, même s’il permet une majorité des comportements désirés.

## Implémentation par automate à états finis

La dernière implémentation réalisée, bien qu’encore inachevée, est un automate à pile qui reconnaît un langage créé pour l’occasion et le transforme en base de règles.

L’objectif était de se substituer au problème d’empan et de permettre une grande flexibilité dans les décisions, sans pour autant impacter les performances, et de permettre la tokenisation **avec le même système** (mais des règles différentes, inspirées du transducteur de tokenisation réalisé au semestre 1, « le carré »).

Il permet aussi d’éliminer l’étape de lexicalisation grâce à son système de set, qui permet l’association directe "les" 🡪 det 🡪 NP. Enfin, la possibilité de choisir si, lors d’une transition, l’on souhaite écrire et où l’on souhaite écrire (chunk précédent ou nouveau chunk ?) permet de préciser son modèle de règles de manière assez précise. Petit bémol cependant : cela peut vite devenir verbeux, notamment lorsqu’il faut que chaque set soit présent dans chaque état (ce qui, honnêtement, sera le cas la plupart du temps).

Un exemple pour la tokenisation est disponible p.15.

# Annexes

### Texte de référence

|  |
| --- |
| Les signaux étaient assez explicites mais Jérémy espérait se tromper. “ J’avais lu sur Doctissimo que cela faisait partie des symptômes mais je n’ai pas voulu m’alarmer. Les gens se moquaient de moi et me disaient mais non, t’en fais pas, t’es pas Français, t’es juste un peu surmené en ce moment.” témoigne-t-il alors qu’il râle dans les embouteillages.  Mais malheureusement, hier, le couperet tombe, après une prise de sang, Jérémy est positif à la nationalité française, une affection longue durée très handicapante remboursée à 100% par la Sécurité sociale. “ Toute votre vie bascule en un instant, vous savez qu’il y a des choses que vous ne pourrez plus jamais apprécier : un voyage au soleil, un film, un mariage. Vous trouverez toujours quelque chose qui ne va pas. Vous savez que quand on vous demandera “ça va “ vous répondrez désormais des phrases comme “ écoute, on fait comme on peut” nous glisse le primo Français qui broie du noir depuis qu’il a vu la météo ce matin.  Il n’existe aucun traitement pour soigner cette terrible maladie alors que près de 65 millions de personnes sont atteintes de nationalité française. “ Pour l’instant la seule chose qui marche pour les soulager, c’est le vin. Mais ce n’est qu’un soin palliatif qui ne traite pas la cause de la maladie ” déclare le docteur Bernard Moutier lui-même atteint par ce terrible mal. |

1 https://www.legorafi.fr/2022/05/25/deprime-et-jamais-content-un-homme-de-37-ans-se-voit-diagnostiquer-francais/

### Segmentation de référence

|  |
| --- |
| Les signaux  étaient  assez explicites  mais  Jérémy  espérait  se tromper  .  “  J’avais lu  sur Doctissimo  que  cela faisait  partie  des symptômes  mais  je n’ai pas voulu  m’alarmer  .  Les gens  se moquaient  de moi  et  me disaient  mais  non  ,  t’en fais pas  ,  t’es pas  Français  ,  t’es juste  un peu  surmené  en ce moment  .  ”  témoigne-t-il  alors qu’  il râle  dans les embouteillages  .  Mais  malheureusement  ,  hier  ,  le couperet  tombe  ,  après une prise  de sang  ,  Jérémy  est positif  à la nationalité française  ,  une affection  longue durée  très handicapante  remboursée  à 100%  par la Sécurité sociale  .  “  Toute votre vie  bascule  en un instant  ,  vous savez  qu’  il y a  des choses  que  vous ne pourrez plus jamais apprécier  :  un voyage  au soleil  ,  un film  ,  un mariage  .  Vous trouverez toujours  quelque chose  qui ne va pas  .  Vous savez  que  quand  on vous demandera  “  ça va  “  vous répondrez  désormais  des phrases comme  “  écoute  ,  on fait  comme  on peut  ”  nous glisse  le primo Français  qui broie  broie  du noir  depuis  qu’  il a vu  la météo  ce matin  .  Il n’existe  aucun traitement  pour soigner  cette terrible maladie  alors que  près de 65 millions  de personnes  sont atteintes  de nationalité française  .  “  Pour l’instant  la seule chose  qui marche  pour les soulager  ,  c’est  le vin  .  Mais  ce n’est  qu’  un soin palliatif  qui ne traite pas  la cause  de la maladie  ”  déclare  le docteur  Bernard Moutier  lui-même atteint  par ce terrible mal  . |

2 texte de référence chunké à la main

### Résultats des essais

### 

Figure 3 Résultats essai 1 (zoomer !) <https://github.com/Jerem-dY/T-GeStaMESP/blob/main/documents/chunks_latest.xlsx>

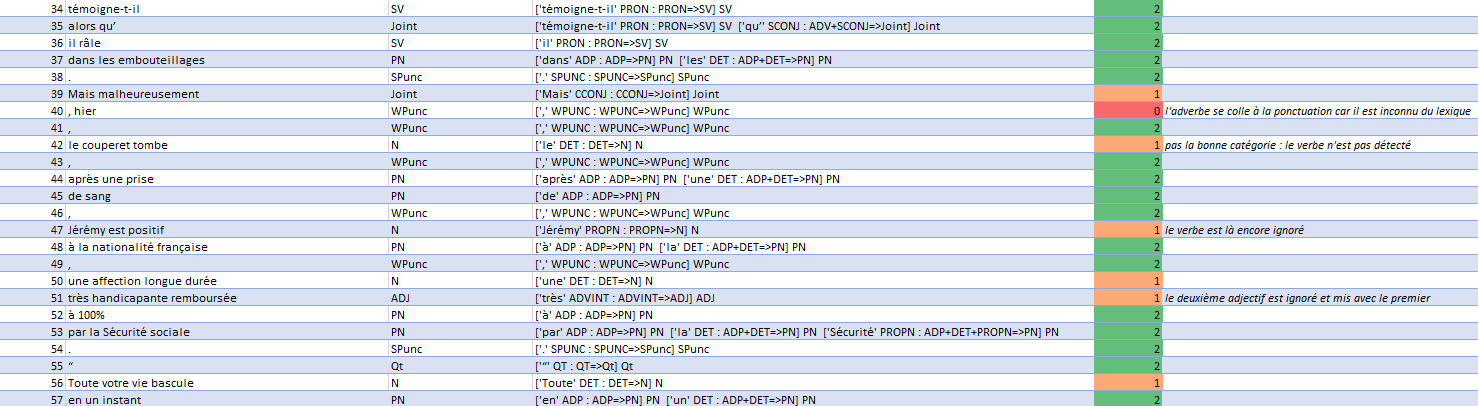


Figure 4 Résultats essai 2 (zoomer !) <https://github.com/Jerem-dY/T-GeStaMESP/blob/main/documents/chunks_latest.xlsx>

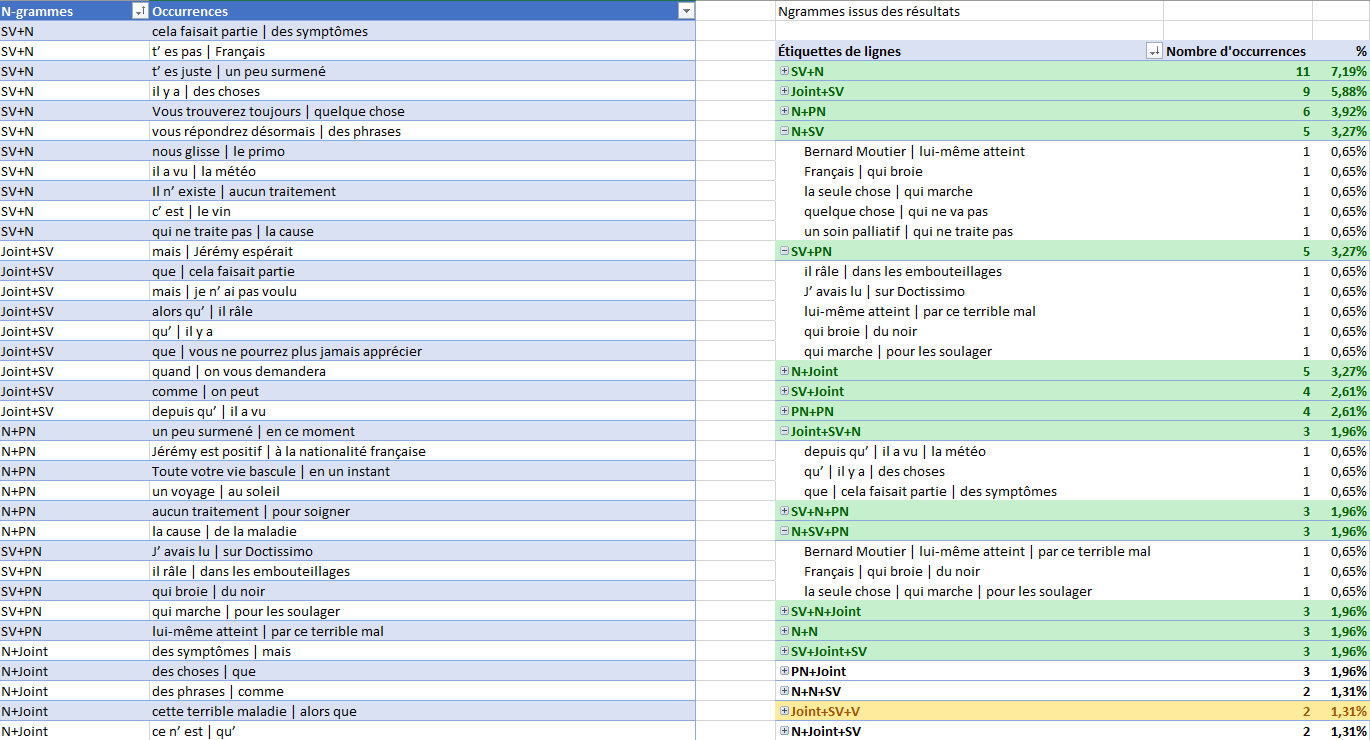


Figure 5 Exploitation essai 2 (zoomer !) <https://github.com/Jerem-dY/T-GeStaMESP/blob/main/documents/chunks_latest.xlsx>

|  |
| --- |
| # Ceci est un fichier de règles pour la tokenisation du français.  # . l'état actuel  # @ les tokens qui ne sont pas dans les sets mentionnées  # ^ n'écrit pas (^. reste sur place et n'écrit pas, ^state n'écrit pas et change d'état)  # % écrit avant de changer d’état  # : état final  # \* état initial  wpunc(',' ';' ':' '“' '”' '"' '«' '»')  spunc('.' '?' '!')  sep(' ' '\n' '\t')  apos("'" '’')  # La ponctuation forte  PoncForte:{  sep = ^Sep  wpunc = PoncFaible  spunc = .  @ = Fo  }  # La ponctuation faible  PoncFaible:{  sep = ^Sep # On ignore les séparateurs  wpunc = PoncFaible  spunc = PoncForte  @ = Fo  }  # Les caractères formant les mots  Fo:{  sep = ^Sep  wpunc = PoncFaible  spunc = PoncForte  apos = %Fo  @ = .  }  # Les séparateurs  Sep:\*{  sep = ^.  wpunc = PoncFaible  spunc = PoncForte  @ = Fo  } |

Figure 6 exemple de fichier pour la tokenisation du français

|  |
| --- |
| <root>  <NP rule="entry → NP">  <leaf>Déprimé</leaf>  </NP>  <CCONJ rule="NP → CCONJ">  <leaf>et</leaf>  <leaf>jamais</leaf>  <leaf>content</leaf>  </CCONJ>  <PUNC rule="CCONJ → PUNC">  <leaf>,</leaf>  </PUNC>  <NP rule="PUNC → NP">  <leaf>un</leaf>  <leaf>homme</leaf>  </NP>  <PP rule="NP → PP">  <leaf>de</leaf>  <leaf>37</leaf>  <leaf>ans</leaf>  </PP>  <VP rule="PP → VP">  <leaf>se</leaf>  <leaf>voit</leaf>  <leaf>diagnostiqué</leaf>  </VP>  <PUNC rule="VP → PUNC">  <leaf>«</leaf>  </PUNC>  <NP rule="PUNC → NP">  <leaf>Français</leaf>  </NP>  <PUNC rule="NP → PUNC">  <leaf>»</leaf>  </PUNC>  <NP rule="PUNC → NP">  <leaf>Les</leaf>  <leaf>signaux</leaf>  <leaf>étaient</leaf>  <leaf>assez</leaf>  <leaf>explicites</leaf>  </NP>  <CCONJ rule="NP → CCONJ">  <leaf>mais</leaf>  <leaf>Jérémy</leaf>  <leaf>espérait</leaf>  </CCONJ>  <VP rule="CCONJ → VP">  <leaf>se</leaf>  <leaf>tromper</leaf>  </VP>  <PUNC rule="VP → PUNC">  <leaf>.</leaf>  </PUNC>  <PUNC rule="PUNC → PUNC">  <leaf>"</leaf>  </PUNC>  ... |

Figure 7 Exemple de sortie XML

# Bibliographie

Abney, S. P. (1991). Parsing By Chunks. In R. C. Berwick, S. P. Abney, & C. Tenny (Éds.), *Principle-Based Parsing* (Vol. 44, p. 257‑278). Springer Netherlands. https://doi.org/10.1007/978-94-011-3474-3\_10

Blache, P. (2001). *Les grammaires de propriétés : Des contraintes pour le traitement automatique des langues naturelles*. Hermès science publications.

Blache, P. (2013). *Chunks et activation : Un modèle de facilitation du traitement linguistique*. 229‑242.

Cooper, D. (2023). *Engineering a Compiler* (Third Edition). Morgan Kaufhann Publishers.

Weil-Barais, A. (2011). *L’homme cognitif* (2e éd). PUF.