IFT6285 – Devoir 5

Analyse Syntaxique avec NLTK/CYK

Maxime MONRAT

Juan Felipe DURAN

Université de Montréal

Automne 2021

Objectifs

Dans ce devoir nous explorons les capacités d'analyse grammaticale offertes par la plateforme *Natural Language Toolkit* (NLTK). Nous utilisons une variante de l'algorithme CYK sur une partie du *Penn Tree Bank* (PTB) afin d'entraîner une grammaire PCFG. Nous utilisons ensuite cette grammaire pour analyser les phrases agrammaticales du corpus de développement de la tâche CoLA du benchmark GLUE.

A) Mots Inconnus

L'analyseur avec lequel nous travaillons, *ViterbiParser*, lève une exception lorsqu'il rencontre un mot inconnu. Afin de remédier à ce problème nous avons extrait tous les terminaux et non-terminaux uniques des productions du PTB. Nous utilisons ensuite ces non-terminaux uniques pour créer des nouvelles productions de la forme A ⟶ UNK. Nous ajoutons ces nouvelles productions à la liste de productions initiale, avant de remplacer tous les mots inconnus (c'est-à-dire non compris dans la liste de terminaux uniques) par UNK.

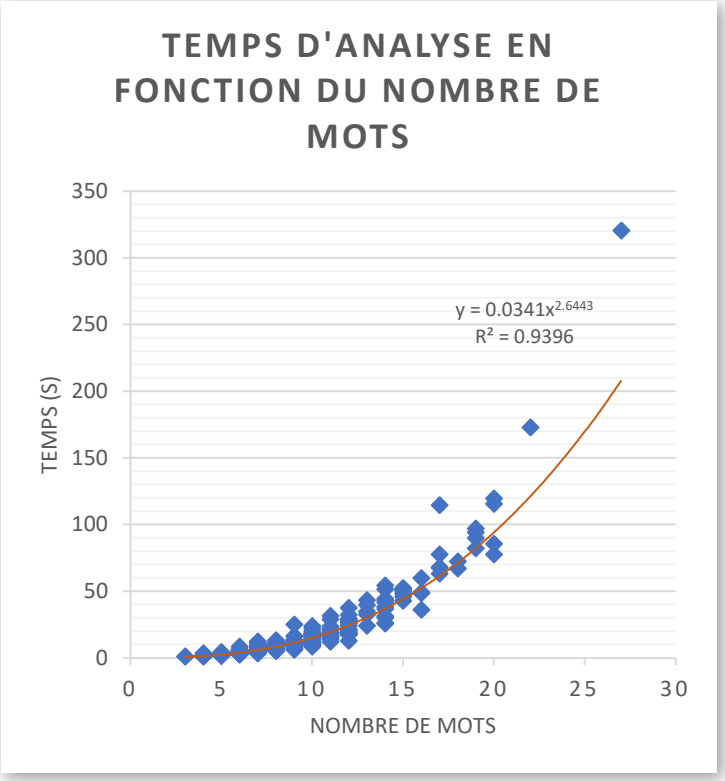
B) Longueur de phrases

NLTK nous permet facilement d'obtenir la longueur d'une phrase du PTB via la fonction *leaves()* qui renvoie la liste des terminaux d'un arbre. Dans notre cas nous obtenons :

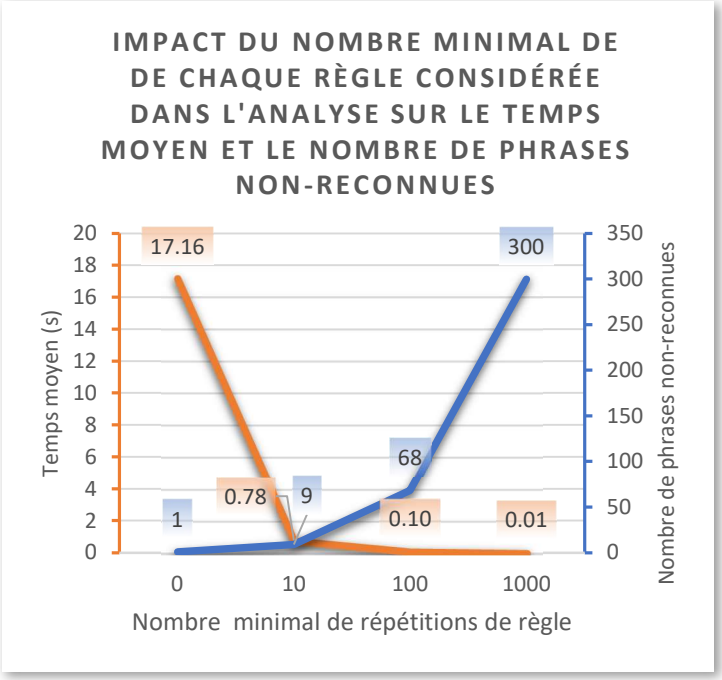
1. Pour le PTB : **25.72 mots**
2. Pour les phrases agrammaticales du dev set de CoLA : **9 mots**

C) Analyse de ViterbiParser

Nous avons effectué notre analyse sur les 300 premières phrases du dev set de CoLA. D'emblée on peut remarquer que le temps de traitement augmente rapidement en fonction du nombre de mots compris dans la phrase, avec un maximum de **5,33 min** pour une phrase de **27 mots**. Le temps de traitement peut néanmoins être diminué par un filtrage de la grammaire, en ne considérant que les règles que l'on peut retrouver un certain nombre de fois (dans notre cas, nous avons testé un filtrage sur les règles se répétant 10, 100 et 1000 fois).

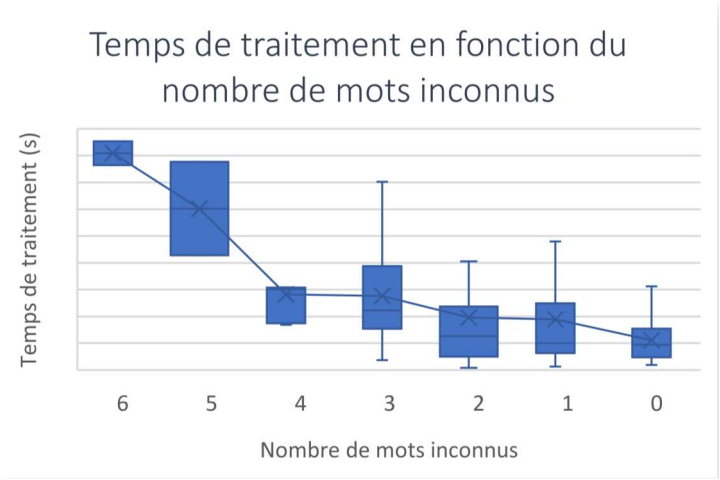


Bien que le filtrage permette d'améliorer drastiquement le temps de traitement, on observe également un impact sur la performance, avec le nombre d'analyses infructueuse augmentant de manière inversement proportionnelle. Bien que cela dépende de l'utilisation prévue, un bon compromis dans notre cas semble être de ne retenir les règles apparaissant au minimum 10 fois dans le corpus d'entraînement : on observe alors une réduction du temps de traitement moyen de plus de **95%** contre une augmentation des phrases non-traitées de seulement **1.67%**.

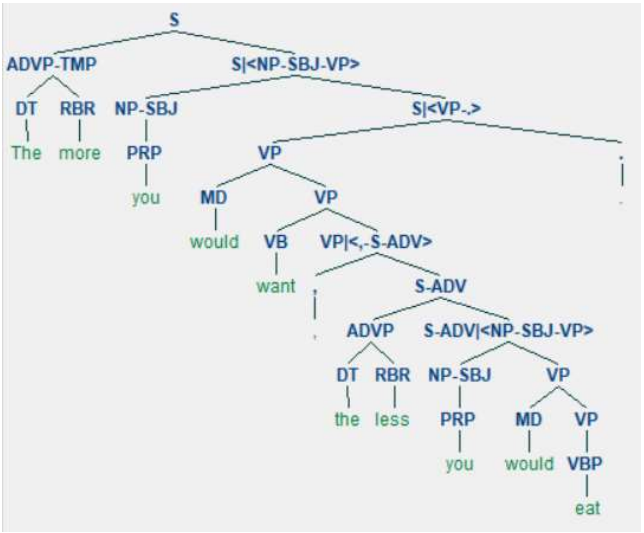


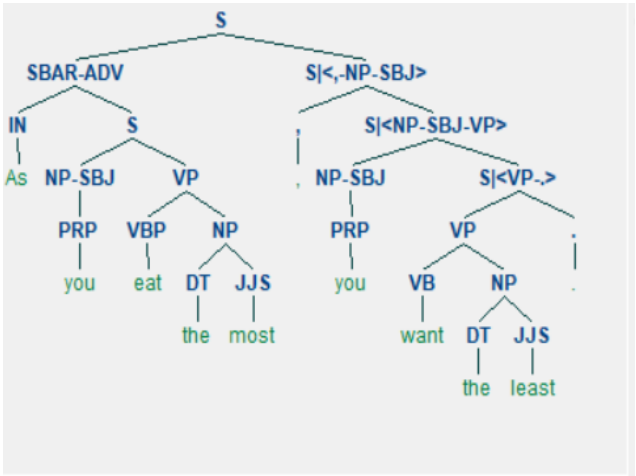
D) Analyse des phrases agrammaticales

De l'ensemble des phrases agrammaticales du dev set de CoLA de maximum 15 mots (290), 10 phrases n'ont pas retourné d'analyse. On observe que le temps de traitement a tendance à augmenter avec le nombre de mots inconnus présents dans la phrase.



Ci-dessous, deux exemples d'analyse grammaticale retournée par NLTK pour deux phrases agrammaticales du dev set de CoLA





E) Comment détecter la justesse grammaticale?

D'après les résultats que nous avons obtenus, illustrés dans les diagrammes ci-après, il semble difficile de détecter les phrases agrammaticales seulement par les probabilités, le nombre de mots ou le temps de traitement. On observe ici pour les phrases agrammaticales une tendance à être traitées plus rapidement et à avoir une probabilité légèrement plus élevée, cependant le faible nombre de données (300 phrases) ne nous permet pas de tirer de conclusions fermes sur ces points.

Nous pourrions néanmoins postuler que les différences de justesse dans la grammaire des phrases proviennent plus vraisemblablement de l'enchaînement des règles. Une analyse plus en profondeur sur ce point pourrait alors permettre une détection plus précise, et plus pertinente.