Université de Montréal

IFT6285 – Traitement automatique des langues naturelles

Devoir 8

---

par:

Bassirou Ndao
(0803389)
Eugénie Yockell
(20071932)

Date de remise: 20 novembre 2020

Dans ce devoir on s'intéresse aux grammaire hors-contextes à l'aide de librairie `NLTK` et du corpus Penn Tree Bank. Tout d'abord, il est nécessaire de générer une grammaire probabiliste induite par les données d'entraînement issues du `PTB`. Ces données sont au format `CFG`. Elles sont composées d'arbres pour différentes phrases. Une dernière étape consiste a générer une grammaire `PCFG` grâce a la fonction `induce_pcfg` de `NLTK`.

Une fois que la grammaire est générée, il est possible restreindre la taille de celle-ci, en sélectionnant les règles à retenir. Les règles les plus fréquentes ont été gardés. À noter que, par exemple, lorsqu'on précise qu'on utilise 50% des règles, cette quantité est prise sur les règles les plus fréquentes. Plus tard nous illustrerons l'impact de cet hyperparamètre au niveau de la qualité du modèle.

En poursuivant notre analyse, nous voulions tester les grammaires obtenues en analysants les phrases test du `PTB` à l'aide de l'analyseur `ViterbiParser`. Nous avons vite fait face à un problème lier la présence de mots inconnus, qui lorsque rencontré par notre parseur, lève une exception. La solution choisie pour y palier, est celle suggéré dans l'énoncé. Pour s'y faire, on regroupe tous les termes de gauche non terminaux possible et pour chaque mot inconnu, on créer des règles avec ceux-ci où le terme de droite est "UNK". Cependant, en addition à celle-ci, une modification des données de test pour remplacer ces même mots inconnus par "UNK" à été appliqué. Il aurait aussi été possible d'ajouter des règles pour chaque mot inconnu, soit utilisé le mot directement au lieu d'utiliser "UNK". Cette solution n'a pas été choisi, car elle générait une énorme quantité de règles et comme il est montré plus loin, l'algorithme d'analyse a une forte dépendance temporelle avec la taille de la grammaire.

En terme d'évaluation, on remarque que les mesures `parseval` ne sont pas disponible dans `NLTK`. Il faut trouver une solution de replis afin d'évaluer notre analyseur syntaxique. La solution proposée dans l'énoncé de ce devoir nous impose de comparer nos données d'entraînement et de test sous le prisme d'étiquetage morpho-syntaxique. Celui-ci est obtenu grâce à la fonction `pos` appliquée à l'arbre résultant du `ViterbiParser` ainsi qu'aux arbres de données test. Des mesures de justesse, précision, de rappel, etc sont produites. La mesure de performance préférée est la justesse (*accuracy*). Les différentes grammaires sont analysées selon le temps nécessaire pour appliqué Viterbi et la justesse de ce dernier.

À la figure 1, on mesure l'impact de la longueur des phrases sur le temps nécessaire pour appliquer l'algorithme de Viterbi avec `ViterbiParser`. En effet, il était pertinent de s'intéresser au temps, puisque c'est un algorithme très lent. La littérature affirme que l'algorithme de Viterbi à une complexité $\mathcal{O}(N^2T)$ où N est le nombre d'états et T est la longueur de la séquence observée. [1] Par contre, comme on le voit à la figure, le temps augmente exponentiellement selon la longueur des phrases. L'algorithme de `NLTK` est implémenté d'une manière qui est exponentielle sur la longueur de la phrase. De plus, comme on s'y attend, l'algorithme est beaucoup plus lent lorsque le nombre de règle augmente. Ce résultat concorde avec la complexité théorique.

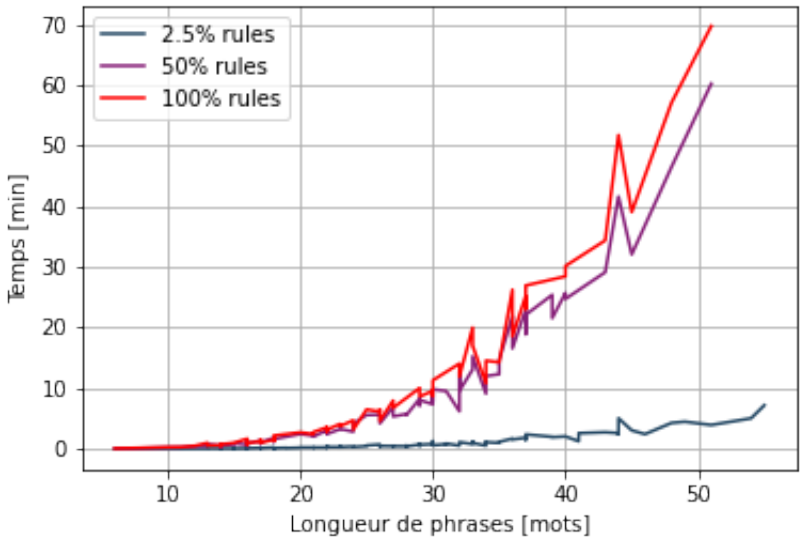

Figure 1: Temps en fonction de la longueur des phrases sur différente taille de grammaire où le nombre de règle total est 205 644.

Une étude de la performance en fonction de la longueur de la phrase a aussi été faite à la figure 2. On remarque qu'il n'y a aucune corrélation entre la longueur des phrases et la justesse. En effet, les données sont trop éparpillées pour en déduire une quelconque relation. Il faut noter qu'on arrive seulement à obtenir une justesse de 100% trois fois qu'avec des plus petites phrases.

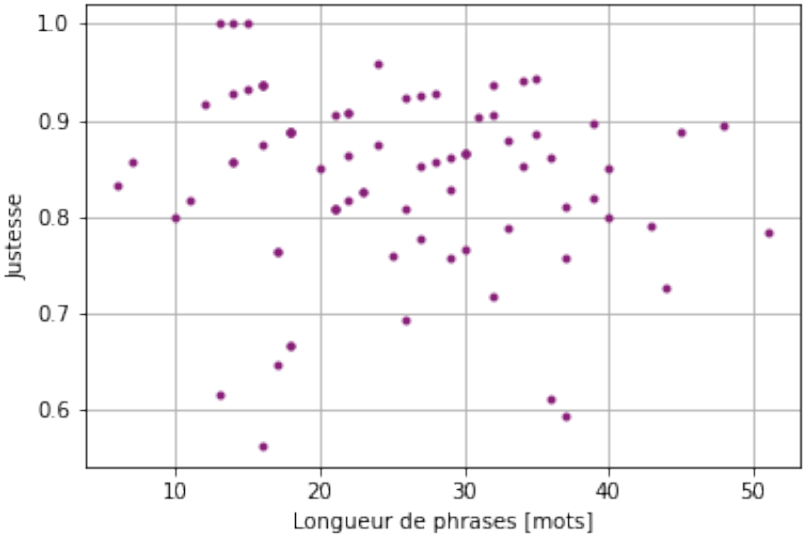

Figure 2: Performance en fonction de la longueur des phrases sur une grammaire de 50% des règles totales.

Intuitivement, notre mesure de précision devrait être impactée à la baisse par le fait de rencontrer des mots inconnus pendant la phase de test. Pour vérifier cette hypothèse on représente la performance du modèle en fonction de la proportion de mots inconnus rencontré par phrase dans la figure 3. Les données observées sur la figure rejoint la pensée initiale que la justesse baisse. On note aussi qu'il y a une grande variabilité dans nos données ce qui a pour effet de produire un nuage de point dans le graphe. On pourrait en déduire qu'il y a une faible corrélation encore la proportion des mots inconnus et la justesse. Cette fluctuation rend un peu plus délicate notre conclusion.

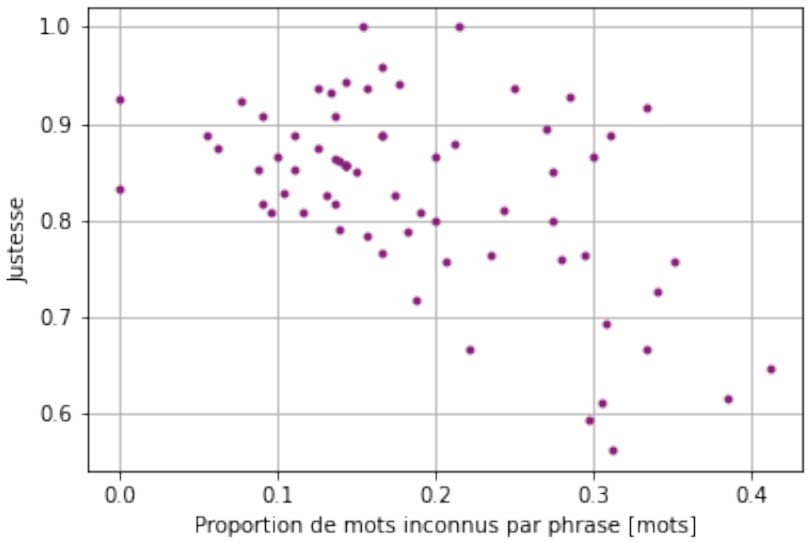

Figure 3: Performance en fonction de la longueur des phrases sur une grammaire de 100% des règles totales.

En conclusion, de l'élaboration de la grammaire à son utilisation dans une tâche d'étiquetage morpho-syntaxique (POS tagging), ce devoir nous a familiarisé avec les techniques d'analyse syntaxique dans `NLTK`. N'ayant pas à notre disposition `parseval`, une solution de replis a cependant été utilisé. Cette solution a permis de définir une méthode d'évaluation adéquat pour notre cas. Il aurait été intéressant de mesurer la différence de performance avec une grammaire qui utilise les mots inconnus directement et avec notre grammaire qui utilise "UNK". De cette façon, on pourrait vérifier si utiliser les mots directement fait augmenter la performance.

References

[1] S. Chatterjee and S. Russell, "A temporally abstracted viterbi algorithm," University of California, Berkeley.