Twit'Oz: un prédicateur de texte

Peter Van Roy Gorby Kabasele
Antoine Vanderschueren Thomas Wirtgen

April 17, 2020

1 Contexte

Les articles écrits par des *bots* en cet âge de l'intelligence artificielle ont attiré l'attention du président des Etats-Unis. En cette période de crise il ne trouve plus le temps pour combiner son golf quotidien avec son obsession pour la *Twittosphère*. Le président a entendu ce matin à la télé qu'Oz était un langage fort efficace car utilisant la *récursion terminale* et un *threading* efficace. Il veut donc vous engager pour lui fournir une application graphique capable de prédire le prochain mot qu'il va écrire à partir des précédents mots. Votre mission, si vous l'acceptez, sera donc d'automatiser l'écriture de ses *tweets* en vous basant sur un historique fourni.

2 Données et objectifs

Ce projet se divise en 2 grandes parties:

- La lecture des données de manière parallèle
- La prédiction par interface graphique

Toute la documentation se trouve ici (ce qui était valable pour Oz 1.4, l'est en grande partie pour Oz 2.0) et dans ce livre de référence vous pouvez aussi trouver certains exemples qui vous faciliteront la vie.

2.1 Traitement des données

Vous avez 208 fichiers texte de chacun maximum 100 lignes contenant tous les tweets sur lesquels vous devrez vous baser. La lecture ligne par ligne d'un fichier texte peut être trouvée ici. Vous choisirez vous-même le nombre de threads (minimum 2) que vous voulez assigner à la lecture des fichiers tant que celles-ci se fait de manière parallèle. Il faut ensuite parser et sauvegarder les données lues. Nous vous conseillons d'utiliser au moins autant de threads de parsing qu'il y a de threads de lecture et un thread pour la sauvegarde des données comme dans la figure 1.

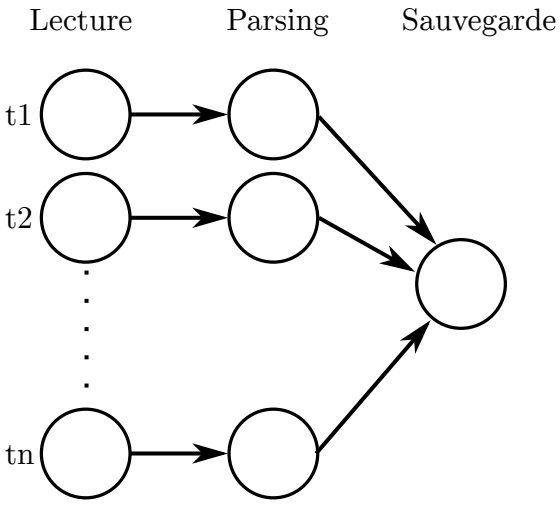

Figure 1: La structure conseillée du threading pour ce projet

2.2 Prédiction

Afin de prédire le mot suivant d'une phrase vous devrez utiliser des N-grammes. Nous ne vous demandons pas un parsing très compliqué des mots pour extraire leur type syntaxique, les utiliser tels quels sera suffisant. Il faudra néanmoins réfléchir au rupture de phrase (points, virgule, hashtags etc.) et au majuscules. Soyez cependant attentif car cette partie **devra** être récursive terminale.

Le principe d'un O-gramme est le suivant: vous prédisez toujours le mot le plus fréquent. Celui d'un 1-gramme est que vous prédisez le mot le plus fréquent en sachant quel était le mot précédent. Par exemple si dans tous les tweets le mot *you* est majoritairement suivi de *are* c'est ce dernier que vous devrez prédire si on input *you* dans la zone de texte. Un 2-gramme étend cette logique aux 2 mots précédents. Donc si on input *you are* le mot le plus fréquent pourrait être *fake* car une phrase souvent utilisée.

Vous remarquez donc que la capacité de mémoire nécessaire croit exponentiellement avec le N du N-gramme. Limitez-vous donc à maximum un 2-gramme en sachant que **nous attendons de vous au minimum un 1-gramme fonctionnel**! Pour le 2-gramme vous devrez déjà limiter les choix de mot ou en extraire les racines pour limiter l'utilisation de la mémoire à vous de voir comment vous voulez gérer ce cas-là. Nous vous conseillons d'utiliser des dictionnaires.

Pour le côté graphique, une bête fenêtre avec une zone de texte et éventuellement un bouton pour demander la prochaine prédiction. Mais vous êtes libres d'améliorer cela à votre guise, votre note finale ne pourra qu'en être plus favorable.

3 Soumission

Le devoir se fera par groupe de 2 et à la fin nous attendons de vous un lien vers votre repo partagé sur Github qui contiendra les 2 choses suivantes:

1. Tout votre code. Celui-ci **doit** contenir un `ReadMe` qui détaille la structure des fichiers ainsi qu'un `Makefile` afin de pouvoir compiler et exécuter votre code. Si vous travaillez sous Windows vous pouvez ajouter les instructions de compilation dans le ReadMe.

2. Un rapport très bref qui contient les détails de vos choix d'implémentation ainsi que quelques captures d'écran de votre interface graphique avec quelques indications d'utilisation.

Gardez vos repos **privés** et ajoutez-nous en collaborateurs afin qu'on puisse aller voir votre code. Plus de détails suivront à ce propos.

Si jamais vous ne trouvez pas de partenaire avec qui faire le projet, vous pouvez le faire seul à condition d'avoir une bonne raison (nombre impair d'étudiant ou personne n'a répondu à vos messages pour trouver un binôme sur le forum). La date limite du projet est le **dimanche 17 mai à 23h55**. Vous enverrez le lien vers votre repo github sur Moodle dans la section prévue à cet égard. **Vous devrez être inscrit dans un groupe** pour soumettre, même si vous êtes seul, faites attention à cela.

4 Questions-réponses

Pour faciliter l'interaction nous allons ouvrir un channel sur teams (afin que tout soit regroupé). Si vous les souhaitez, nous pouvons aussi organiser une session questions-réponses.