Projet 3: acteurs poètes

- Échéance : soumettre le projet avant le 14 avril à 10h40
- Coder dans un style fonctionnel est nécessaire pour réussir
- Personne de contact : Cédric Libert

Introduction

Dans cette seconde partie du projet 2, on va créer des poèmes collaboratifs grâce aux acteurs.

1 Tutoriel à suivre absolument

Vous allez être amenés à utiliser la bibliothèque Akka. Pour ce faire, suivez d'abord le tutoriel suivant : `http://doc.akka.io/docs/akka/2.0/intro/getting-started-first-scala.html`. Veillez à bien comprendre chaque étape et à avoir un code qui s'exécute à la fin.

2 Objectifs d'apprentissage

Après ce projet, vous devrez être capables de concevoir un programme comme une communauté d'acteurs qui interagissent en s'envoyant des messages, qui créent d'autres acteurs et les stoppent.

3 Poème collaboratif : les règles du jeu

- Chaque Poète est un acteur qui possède un corpus particulier
- Chaque poète, au début, envoie une phrase au hasard au Juge.
- Le Juge est un acteur qui envoie chacune des phrases reçues à chacun des Poètes.
- Quand un poète reçoit une phrase, il essaie de trouver celle, qui dans son corpus, va le mieux avec la phrase qu'il a reçue...et **qu'il n'a pas encore envoyée** ! Il l'envoie ensuite au juge.
- S'il ne trouve pas de phrase, il a fini.
- Le juge qui reçoit de nouvelles phrase les renvoie aux Poètes qui ne se sont pas encore arrêtés. Et ainsi de suite. Au fur et à mesure, le juge affiche les phrases qu'il reçoit.
- Jusqu'à ce qu'il n'y ait plus de Poète.

4 Deux phrases qui vont bien ensemble ?

Jusqu'à présent, on n'a pris en compte que les phrases qui riment et qui ont un nombre de syllabes suffisamment proche. On veut aussi tenir compte du sens des mots pour voir si deux phrases vont bien ensemble.

Soit p une phrase, constituée des mots $m_1, m_2, ..., m_n$.

Soit $sens(m)$ le sens du mot m. Une approximation simple du sens de m est l'ensemble des mots (ou le « sac de mots ») qui font partie de l'article Wikipédia de m.

Soit $sens(p)$ le sens de la phrase p. Ce sens est l'union du sens de tous ses mots : $sens(m_1) \cup sens(m_2) \cup ... \cup sens(m_n)$.

Deux phrases p_1 et p_2 qui vont bien ensemble ont un grand score de similarité, calculé comme suit : $\frac{|sens(p_1) \cap sens(p_2)|}{|sens(p_1) \cup sens(p_2)|}$. À vous de décider ce que vous considérez être un grand score de similarité.

5 Récupérer un article Wikipédia

Vous pouvez utiliser le code suivant pour récupérer un article, correspondant à un certain mot, sous la forme d'une chaine de caractères :

```
import scala.xml._
//...
(XML.load("https://fr.wikipedia.org/w/api.php?action=query&titles="
+ mot +
"&prop=revisions&rvprop=content&format=xml")\\ "rev").toString
```

Un acteur Poète devrait, idéalement, lancer un acteur par mot de la phrase dont il veut trouver le sens. Chacun serait chargé de récupérer la page Wikipédia correspondant à ce mot et à la renvoyer au Poète.

5.1 Évaluer ses pairs

Vous devrez évaluer le projet de trois autres étudiants. Cette évaluation fait partie de l'exercice et est obligatoire pour le réussir. Les critères d'évaluation sont les suivants :

- Compile
- Sans warning
- s'exécute, est correct (fait ce qui est demandé), se termine
- définition de différents acteurs, de différents messages. Envoi de messages entre acteurs avec `!`. Réception de messages avec `receive`.

Ajoutez en plus de ça un **feedback**. Cette seconde partie est vraiment destinée à dire à vos condisciples comment, d'après vous, ils pourraient s'améliorer et ce que vous trouvez intéressant dans ce qu'ils ont fait. Fournissez un feedback complet et précis. C'est l'une des parties les plus importantes de l'exercice. Ce feedback aura deux partie :

- commentaires concernant tous les critères définis ci-dessus (ça ne compile pas, pourquoi ? ça ne s'exécute pas correctement, qu'est-ce que ça fait à la place de ce qui est demandé ? D'où vient le problèmes ? Quels concepts n'ont pas été implémentés ? ...)
- suggestions et commentaires : commentaires négatifs **et** positifs, questions (pourquoi tu as fait ceci ou cela ?), suggestions.

Veillez, dans les deux parties, à utiliser des **termes précis et objectifs** pour évaluer la qualité (contre exemple : bien, super, excellent ne sont pas des termes précis et objectifs) ainsi que pour décrire le programme (utilisez les termes qui conviennent en Scala : classe, case class, object, application partielle, fonction, argument,...).

J'insiste encore une fois sur le fait que le feedback porte sur la qualité du **code**, et pas uniquement sur les résultats à l'exécution ou à la compilation. Ça signifie que, même si ça ne compile pas ou si ça ne fait pas ce qui est demandé, il faut fournir un feedback. À l'inverse, même si le code vous semble très

bon, justifiez pourquoi vous le trouvez si intéressant (en citant des bouts de code qui vous ont étonnés positivement, par exemple), et proposez, le cas échéant, quelques améliorations possibles.

5.2 Dernières consignes importantes

- envoyez vos fichiers **en UTF8 sans BOM**
- mettez des **chemins relatifs** pour les fichiers, et considérez que le dictionnaire et le corpus sont dans le même dossier que le fichier de code. Ça facilitera la correction.
- envoyez du code qui compile
- essayez de nettoyer votre code pour ne pas avoir de warning
- **N'indiquez pas de package**. Ça facilitera la correction.
- Pas besoin de faire de l'interaction avec l'utilisateur.