Projet 3: poèmes de promesses futures

- Échéance : soumettre le projet avant le 5 mars à 10h40
- Le travail est individuel et coté
- Coder dans un style fonctionnel est nécessaire pour réussir
- Personne de contact : Cédric Libert

Introduction

On va encore améliorer notre générateur de poèmes pour qu'il nous permette de générer des œuvres plus sérieuses : des quatrains. Un quatrain est un poème (ou un morceau de poème) constitué de quatre vers. Pour cette exercice, on va créer uniquement des quatrain en rimes embrassées, c'est-à-dire dont les vers 1 et 4 riment et dont les vers 2 et 3 riment (structure ABBA). Par exemple :

Je lui demandai :
est-elle morte il y a cent ans ?
et je pleurai trois jours amèrement.
Personne ne le savait.

Évidemment, pour que ces quatrains soient plus créatifs, on va chercher à créer les deux vers intérieurs à partir d'un corpus particulier, et les deux vers extérieurs à partir d'un autre corpus. On utilisera aussi deux corpus de secours, au cas où les deux premiers corpus sont occupés, inaccessibles, inexistants... Ici on travail en local, mais imaginez que ces corpus pourraient être des ressources distantes présentes sur des serveurs différents (qui pourraient être temporairement inaccessibles) et qu'il ne serait pas efficace de les concaténer directement avant le traitement. Idem pour le dictionnaire, on prévoie un dictionnaire de secours.

Cela nous permet, par ailleurs d'utiliser des Future et des Promise.

1 Objectifs d'apprentissage

Dans ce projet, vous utiliserez :
- Future
- Promise
- onSuccess
- onFailure
- recover ou recoverWith
- success (sur une Promise)
- (p :Promise).future
- Await.ready
- ...les Future dans des « for comprehension »

2 Modifications périphériques

2.1 À importer

```scala
import scala.concurrent._
import scala.concurrent.duration._
import ExecutionContext.Implicits.global
```

2.2 Modification de `DeuxVers.ecrire`

On aimerait que la méthode `ecrire` renvoie une liste de phrases (en réalité, elle en renverra deux). Ou une exception si rien n'a pu être généré. Plus de Try donc, on va gérer les erreurs ailleurs avec les Future.

2.3 Modification de `extraire_phrases`

Cette fonction doit être modifiée très légèrement pour renvoyer un résultat de type `Future[List[Phrase]]`.

3 Modifications de la fonction `main`

3.1 Le quatrain promis

L'idée derrière la `Promise`, c'est de déclarer une variable qui n'a pas encore de valeur, et de lui assigner une valeur à un moment donné de l'exécution. Notre `Promise` ici, ça sera notre quatrain.

Une fois le quatrain calculé (sous la forme d'une chaine de caractères, avec des retours à la ligne pour une jolie présentation), il faudra l'assigner correctement à cette `Promise`.

Attention : le programme ne devra s'arrêter que quand le `Future` associé à ce quatrain sera terminé. Si vous ne faites pas ça, il est possible que votre programme s'arrête avant que vos poèmes ne soient générés. Et vu qu'il s'agit, en réalité, de threads, il sont arrêtés si le processus dans lequel ils ont été créés est arrêté.

3.2 Extraction de phrases

Vous allez devoir extraire les phrases depuis deux corpus différents. Il est évidemment intelligent de paralléliser cette action. Heureusement, `extraire_phrase` renvoie un `Future`.

Attention ! faites-en sorte que, en cas de souci dans la fonction d'extraction de phrases (le corpus n'existe pas par exemple), un corpus de secours et un dictionnaire de secours soient utilisés.

3.3 Créer deux fois deux vers

Pour chacune des listes de phrases extraites, il faut générer deux phrases qui riment. Attention, n'hésitez pas à faire ça dans un `Future`.

3.4 Si ça fonctionne

Si vous parvenez à créer deux fois deux vers, mettez-les correctement dans la `Promise`

3.5 ...sinon

Sinon mettez dans le quatrain le message d'erreur.

3.6 Finalement

Trouvez comment afficher le contenu du quatrain.

4 Pour avoir des poèmes plus jolis

N'hésitez pas à filtrer la liste de phrases pour ne garder que celles qui ne sont pas trop petites ni trop grandes.