Correction TP lexique

Compte des occurrences de mots

Exemple :

{ (la : 1); (vache : 1); (et : 2); (le : 2); (veau : 1); (chien : 1); (chat : 1) }

clé
mot du texte

valeur
nombre de phrases dans lesquelles le mot apparaît

HashMap<String, Integer>

Idée de countSentencesWithWord

- Découper le texte en phrases
- Pour chaque phrase :
    - Découper la phrase en mots
    - En faire un ensemble sans répétition
    - Parcourir les mots, et pour chaque :
        - Ajouter 1 au nombre d’occurrences de ce mot

Idée de countSentencesWithWord

- Découper le texte en phrases

méthode split sur les string (cf main)
lire la javadoc (dont l'exemple)

- Pour chaque phrase :
    - Découper la phrase en mots
    - En faire un ensemble sans répétition
    - Parcourir les mots, et pour chaque :
        - Ajouter 1 au nombre d'occurrences de ce mot

Idée de countSentencesWithWord

- Découper le texte en phrases

méthode split sur les string (cf main)
lire la javadoc (dont l'exemple)
String[] phrases = txt.split("/n");

- Pour chaque phrase :
    - Découper la phrase en mots
    - En faire un ensemble sans répétition
    - Parcourir les mots, et pour chaque :
        - Ajouter 1 au nombre d'occurrences de ce mot

Idée de countSentencesWithWord

- Découper le texte en phrases String[] phrases
- Pour chaque phrase : *boucle for ou for each sur* phrases (*variable* phrase)
  - Découper la phrase en mots *split d'une phrase en mots* String[] mots = phrase.split(" ");
  - En faire un ensemble sans répétition
  - Parcourir les mots, et pour chaque :
    - Ajouter 1 au nombre d'occurrences de ce mot

Idée de countSentencesWithWord

- Découper le texte en phrases String[] phrases
- Pour chaque phrase : *boucle sur* phrases (*variable* phrase)
  - Découper la phrase en mots String[] mots
  - En faire un ensemble sans répétition

    cf sujet du TP :
    HashSet<String> uniqWord =
    new HashSet(Arrays.asList(phrase));
  - Parcourir les mots, et pour chaque :
    - Ajouter 1 au nombre d'occurrences de ce mot

Idée de countSentencesWithWord

- Découper le texte en phrases String[] phrases
- Pour chaque phrase : *boucle sur* phrases (*variable* phrase)
  - Découper la phrase en mots String[] mots
  - En faire un ensemble sans répétition `HashSet<String> motsUniqueDansPhrase = new HashSet(Arrays.asList(phrase));`
  - Parcourir les mots, et pour chaque :
    - Ajouter 1 au nombre d'occurrences de ce mot

Ensemble sans répétition de String :
Pas de doublon, pas d'ordre
Se parcourt par for each

Constructeur qui demande une liste
-> on convertit

Idée de countSentencesWithWord

- Découper le texte en phrases String[] phrases
- Pour chaque phrase : *boucle sur* phrases (*variable* phrase)
  - Découper la phrase en mots String[] mots
  - En faire un ensemble sans répétition HashSet<String> motsUniqueDansPhrase;
  - Parcourir les mots, et pour chaque : for (String unMot : motsUniqueDansPhrase);
    - Ajouter 1 au nombre d'occurrences de ce mot

Ensemble sans répétition de String :
Pas de doublon, pas d'ordre
Se parcourt par for each

Idée de countSentencesWithWord

- Découper le texte en phrases String[] phrases
- Pour chaque phrase : *boucle sur* phrases (*variable* phrase)
  - Découper la phrase en mots String[] mots
  - En faire un ensemble sans répétition HashSet<String> motsUniqueDansPhrase;
  - Parcourir les mots, et pour chaque : for (String unMot : motsUniqueDansPhrase);
    - Ajouter 1 au nombre d'occurrences de ce mot

Ensemble sans répétition de String :
Pas de doublon, pas d'ordre
Se parcourt par for each

Suppose d'avoir déclarer au début le résultat sur lequel on veut travailler

Idée de countSentencesWithWord

Type de retour de la fonction (= méthode statique)

HashMap<String, Integer> occ = new HashMap<String, Integer>();

Commence vide (aucun couple clé-valeur), on va les ajouter au fur et à mesure

- Découper le texte en phrases — String[] phrases
- Pour chaque phrase : *boucle sur* phrases (*variable* phrase)
  - Découper la phrase en mots — String[] mots
  - En faire un ensemble sans répétition — HashSet<String> motsUniqueDansPhrase;

Ensemble sans répétition de String :
Pas de doublon, pas d'ordre
Se parcourt par for each

  - Parcourir les mots, et pour chaque : for (String unMot : motsUniqueDansPhrase);
    - Ajouter 1 au nombre d'occurrences de ce mot

Suppose d'avoir déclarer au début le résultat sur lequel on veut travailler

Idée de countSentencesWithWord

Type de retour de la fonction (= méthode statique)

HashMap<String, Integer> occ = new HashMap<String, Integer>();

Commence vide (aucun couple clé-valeur), on va les ajouter au fur et à mesure

- Découper le texte en phrases — String[] phrases
- Pour chaque phrase : *boucle sur* phrases (*variable* phrase)
  - Découper la phrase en mots — String[] mots
  - En faire un ensemble sans répétition — HashSet<String> motsUniqueDansPhrase;

Ensemble sans répétition de String :
Pas de doublon, pas d'ordre
Se parcourt par for each

  - Parcourir les mots, et pour chaque : for (String unMot : motsUniqueDansPhrase);
    - Ajouter 1 au nombre d'occurrences de ce mot

Ici, le mot peut avoir déjà été rencontré (et avoir une entrée dans occ) ou pas encore (et ne pas en avoir)

Suppose d'avoir déclarer au début le résultat sur lequel on veut travailler

Idée de countSentencesWithWord

HashMap<String, Integer> occ

- Découper le texte en phrases String[] phrases
- Pour chaque phrase : *boucle sur* phrases (*variable* phrase)
  - Découper la phrase en mots String[] mots
  - En faire un ensemble sans répétition HashSet<String> motsUniqueDansPhrase;
  - Parcourir les mots, et pour chaque : for (String unMot : motsUniqueDansPhrase);
    - Ajouter 1 au nombre d'occurrences de ce mot

Ensemble sans répétition de String :
Pas de doublon, pas d'ordre
Se parcourt par for each

Ici, le mot peut avoir déjà été rencontré (et avoir une entrée dans occ) ou pas encore (et ne pas en avoir)

Idée de countSentencesWithWord

HashMap<String, Integer> occ

- [...]
    - Ajouter 1 au nombre d'occurrences de ce mot
        - on cherche mot dans occ :
            - s'il existe, on ajoute 1 à la valeur correspondante
            - s'il n'existe pas, on commence avec une valeur de 1 (on vient de croiser le mot)
        - on ajoute/remplace (mot, nouvelle valeur) à HashMap

Ici, le mot peut avoir déjà été rencontré (et avoir une entrée dans occ) ou pas encore (et ne pas en avoir)

Idée de countSentencesWithWord

HashMap<String, Integer> occ

- [...]
    - Ajouter 1 au nombre d'occurrences de ce mot
        - on cherche mot dans occ : Integer nb = occ.get(mot);
            - s'il existe, on ajoute 1 à la valeur correspondante
            - s'il n'existe pas, on commence avec une valeur de 1 (on vient de croiser le mot)
        - on ajoute/remplace (mot, nouvelle valeur) à HashMap

Idée de countSentencesWithWord

HashMap<String, Integer> occ

- [...]
    - Ajouter 1 au nombre d'occurrences de ce mot
        - on cherche mot dans occ : `Integer nb = occ.get(mot);`

          Soit un nombre soit null.
          null si l'entrée n'existait pas
            - s'il existe, on ajoute 1 à la valeur correspondante
            - s'il n'existe pas, on commence avec une valeur de 1 (on vient de croiser le mot)
        - on ajoute/remplace (mot, nouvelle valeur) à HashMap

Idée de countSentencesWithWord

HashMap<String, Integer> occ

- [...]
    - Ajouter 1 au nombre d'occurrences de ce mot
        - on cherche mot dans occ : `Integer nb = occ.get(mot);` *Soit un nombre soit null. null si l'entrée n'existait pas*
            - s'il existe, on ajoute 1 à la valeur correspondante `nb ++;`
            - s'il n'existe pas, on commence avec une valeur de 1 (on vient de croiser le mot)

              `if (nb == null)` `nb = 1;`
        - on ajoute/remplace (mot, nouvelle valeur) à HashMap

          `occ.put(mot,nb);`

          il y a une correspondance clé <-> valeur,
          s'il y avait déjà une entrée pour mot, elle est remplacée (comme pour tab[i] = val;)
          sinon elle est créée

Table de cooccurrence

Table de cooccurrence : Exemple

{'chat': {'and': 1, 'the': 1, 'dog': 1, 'cat': 1},
'chien': {'and': 1, 'the': 1, 'dog': 1, 'cat': 1},
'et': {'and': 2, 'calf': 1, 'cat': 1, 'cow': 1, 'dog': 1, 'the': 2},
'la': {'and': 1, 'the': 1, 'cow': 1, 'calf': 1},
'le': {'and': 2, 'calf': 1, 'cat': 1, 'cow': 1, 'dog': 1, 'the': 2},
'vache': {'and': 1, 'calf': 1, 'cow': 1, 'the': 1},
'veau': {'and': 1, 'the': 1, 'cow': 1, 'calf': 1}}

HashMap<String,Integer>

traductions possibles déjà rencontrées

HashMap<String, HashMap<String, Integer>>

mot francais
clé

trad. possibles
valeur

Idée de buildCoocTable

- Découper les 2 textes en phrases
- Parcourir **simultanément** chacun des 2 tableaux de phrases :
    - Découper la phrase fr en mots et en faire un ensemble sans répétition
    - Découper la phrase en en mots et en faire un ensemble sans répétition
    - Parcourir les mots fr et pour chaque :
        - Récupérer l'ensemble des trad. possibles déjà calculées
        - Parcourir les mots en et pour chaque
        - Ajouter 1 au nombre d'occurrences de ce mot

Idée de buildCoocTable

- Découper les 2 textes en phrases *split*
- Parcourir **simultanément** chacun des 2 tableaux de phrases :

Force l'utilisation d'une boucle for où l'on gère nous-même les indices

  - Découper la phrase fr en mots et en faire un ensemble sans répétition
  - Découper la phrase en en mots et en faire un ensemble sans répétition
  - Parcourir les mots fr et pour chaque :
    - Récupérer l'ensemble des trad. possibles déjà calculées
    - Parcourir les mots en et pour chaque
    - Ajouter 1 au nombre d'occurrences de ce mot

Idée de buildCoocTable

- Découper les 2 textes en phrases *split*

- Parcourir **simultanément** chacun des 2 tableaux de phrases :

Force l'utilisation d'une boucle for où l'on gère nous-même les indices

  - Découper la phrase fr en mots et en faire un ensemble sans répétition
  - Découper la phrase en en mots et en faire un ensemble sans répétition

  - Parcourir les mots fr et pour chaque :

    - Récupérer l'ensemble des trad. possibles déjà calculées

Si on a encore aucune traduction possible dans la table de cooccurrence, il faut créer une nouvelle entrée, initialement vide

```
mapCurTradCanddidateFromFr = new HashMap<String, Integer>();
```

    - Parcourir les mots en et pour chaque

    - Ajouter 1 au nombre d'occurrences de ce mot