TP7, 8 et 9: Projet de traduction automatique Français-Anglais

Objectifs et Notation L'objectif de ce projet est de programmer une application de traduction automatique du français vers l'anglais. Ce projet comporte 4 niveaux mais

seul le **le niveau 1** et **le compte rendu** seront **évalués** cette année.

Les niveaux suivants sont optionnels. Vous devez terminer le niveau i avant de traiter le niveau $i+1$, etc. **Conservez le code de la fonction de traduction de niveau i avant de commencer la fonction de traduction de niveau $i+1$.**

Niveau 1 Une seule table associative simple (sans recherche dichotomique) : un **lexique** qui associe à un mot français un mot anglais ;

Niveau 2 On complète le lexique par une seconde table associative simple (sans recherche dichotomique) : une **table des racines** qui associe un mot conjugué ou accordé sa racine. Par exemple on associe au mot "travaux" (accordé) le mot "travail". On associe au mot "cherchais" (conjugué) le mot "chercher". Cela permet, ensuite, de traduire le mot "travaux" par "work" et "cherchais" par "look for" alors qu'ils ne figurent pas dans le lexique. On peut utiliser des bornes pour le nombre maximal des entrées dans ces tables : 17000 pour le lexique et 290000 pour les racines ;

Niveau 3 On accélère les performances de la traduction en utilisant la recherche dichotomique dans les tables associatives utilisées pour le lexique et la table des racines. On utilise toujours les mêmes bornes pour les tailles maximales des tables ;

Niveau 4 On rend les tables dynamiques, sans imposer de taille maximale a priori.

Préliminaires Importez le projet Eclipse `/share/l1ie/SI1/ProjetTraduction/ProjetTraduction.zip`. Dans le répertoire `lib`, le fichier `frenchEnglish.txt` contient un lexique français-anglais. Le principe de ce lexique est : pour un mot français situé sur la ligne i, sa traduction est sur la ligne $i+1$.

1 Les tables d'association

1.1 Introduction

Les tables d'association, aussi nommées "tableaux associatifs" ou "map" en anglais, sont des structures de données associant un ensemble de clés à un ensemble de valeurs. Par exemple :

— un lexique français-anglais est une **table d'association** qui donne pour un mot français (la clé, de type `String`) sa traduction (la valeur, de type `String`) ;

clé	valeur
bateau	boat
abîme	abyss
boisson	beverage

— l'affectation des étudiants de SI1 en groupe de TD est une table qui associe à chaque nom et prénom d'étudiant (la clé, de type `String`) le numéro de son groupe (la valeur, de type `int`).

clé	valeur
Jean-Paul Taitinger	2
Éloïse Mumm	1
Kévin Mercier	6
Hélène Ruinart	1

Les clés sont uniques : chaque clé apparaît au plus **une fois** dans une table. En revanche, les valeurs ne sont pas nécessairement uniques comme on le voit avec la valeur 1 dans l'exemple de table précédent. Les trois opérations élémentaires sur les tables d'association sont :

La recherche d'une clé dans la table Par exemple, dans la table précédente la clé "Kévin Mercier" apparaît dans la table. La clé "Max Duchêne" n'apparaît pas.

La restitution d'une valeur associée à une clé Dans la table précédente, la valeur associée à la clé "Kévin Mercier" est 6. La valeur associée à la clé "Max Duchêne" n'existe pas.

L'ajout d'un couple clé-valeur dans une table Si on ajoute le couple clé-valeur ("Margot Pommery",2) dans la table, on obtient la table suivante :

clé	valeur
Jean-Paul Taitinger	2
Éloïse Mumm	1
Kévin Mercier	6
Hélène Ruinart	1
Margot Pommery	2

1.2 Différences entre tableaux et tables d'association

On peut voir les tableaux (Java) comme une forme **particulière** de table d'association dans laquelle les clés sont des entiers (positifs et rangés par ordre croissant dans la table), et les valeurs sont du type des éléments du tableau. Par exemple le tableau `String[] t= {"joie","abc","marcel"}`

	t
0	joie
1	abc
2	marcel

peut être vu comme la table :

clé	valeur
0	joie
1	abc
2	marcel

En revanche, avec un tableau, il n'est pas possible d'utiliser autre chose qu'un entier positif comme clé. Voyons comment représenter **une** table d'association à l'aide de **deux** tableaux.

1.3 Représentation des tables d'association à l'aide de deux tableaux

Reprenons la table d'association précédente, représentant un lexique français-anglais :

clé	valeur
bateau	boat
abîme	abyss
boisson	beverage

Nous allons représenter cette table à l'aide de **deux** tableaux : un tableau `String[] cles` pour les clés et un tableau de `String[] valeurs` pour les valeurs.

	`cles`
0	bateau
1	abîme
2	boisson

	`valeurs`
0	boat
1	abyss
2	beverage

Cependant, les tableaux sont de tailles fixes alors que la dimension des tables peut, elle, évoluer (par exemple lors de l'ajout d'un couple clé-valeur). Nous allons donc initialiser nos tableaux en sur-estimant le nombre maximal d'éléments pouvant être ajoutés à une table et utiliser un index `nbAssoc` donnant le nombre d'associations effectivement utilisées dans les deux tableaux. En particulier, les valeurs présentes à partir des index `nbAssoc` et au delà n'ont pas d'importance, on les représente ici par " ? ?". La représentation devient donc :

`nbAssoc=3`

	`cles`
0	bateau
1	abîme
2	boisson
3	? ?
4	? ?
⋮	⋮

	`valeurs`
0	boat
1	abyss
2	beverage
3	? ?
4	? ?
⋮	⋮

Sur cette représentation, les trois opérations élémentaires deviennent :

La recherche d'une clé dans la table Pour savoir si la clé "abîme" apparaît dans la table d'association, il suffit de vérifier si la chaîne "abîme" est présente dans le tableau `cles` entre les indices 0 et `nbAssoc-1`.

La restitution d'une valeur associée à une clé Pour restituer la valeur associée à la clé "abîme", on recherche "abîme" dans le tableau `cles` entre les indices 0 et `nbAssoc-1`. Cette chaîne est présente dans `cles` à la position 1. On sait que la valeur associée est donc `valeurs[1]`, ici la chaîne "abyss".

L'ajout d'un couple clé-valeur dans une table Pour ajouter le couple clé-valeur ("bienvenue","welcome") dans la table, on modifie notre représentation en :

`nbAssoc=4`

	`cles`
0	bateau
1	abîme
2	boisson
3	bienvenue
4	? ?
⋮	⋮

	`valeurs`
0	boat
1	abyss
2	beverage
3	welcome
4	? ?
⋮	⋮

2 Traduction Automatique, niveau 1

2.1 Construction de la table d'association

Dans le fichier `Main.java` du package `main` du projet Eclipse, définissez la table d'association représentant le lexique français/anglais. Vous utiliserez pour cela la représentation des tables expliquée précédemment et la fonction `String[] ACX.lectureDico(String nomFichier)`.
Remarque : Vous pouvez isoler cette construction dans une *fonction* qui initialise (voire crée) la table.

2.2 Fonction de traduction d'un texte

Dans le fichier `Main.java` du package `main` du projet Eclipse, définissez une fonction de traduction qui, étant donné un tableau de mots `texte` donné en entrée (un tableau de `String`) rend un tableau de `String` de même longueur. Dans le tableau résultat, à l'indice `i`, on trouve la traduction du mot `texte[i]` s'il figure dans le lexique ou le mot non traduit `texte[i]` sinon. La fonction aura la signature suivante :

```
public static String[] traduire(String[] texte)
```

Comme dans le projet de correction, le lexique ne doit pas être passé en paramètre de la fonction `traduire`, et vous chargerez le lexique une seule fois (et non à chaque appel de `traduire`).

2.3 Tests unitaires avec JUnit

Dans le répertoire `testsJUnit`, vous trouverez un fichier `TraducteurTest.java` dans lequel vous devez écrire vos tests **pour toutes les fonctions !** A vous de déterminer quels sont les tests pertinents.

2.4 Intégration dans l'interface graphique

On vous fournit une interface graphique pour l'application de traduction automatique. Pour la lancer, dans la fonction `main` du fichier `Main.java` faites un appel à la fonction `ACX.interfaceTraduction`. Par exemple, `ACX.interfaceTraduction("traduire")`, si le nom de votre fonction de traduction est `traduire`. Au lancement, une fenêtre apparaît. Vous pouvez entrer du texte au clavier dans cette fenêtre, le texte sera automatiquement traduit avec la fonction de traduction dont le nom est passé en paramètre.

2.5 Limitation du niveau 1

Dans l'interface graphique, si vous demandez à traduire la phrase : "Je cherche mes belles chaussures", le résultat pour le niveau 1 devrait être "I cherche my belles chaussures"... c'est un début. Le point bloquant est ici que les mots "cherche", "belles" et "chaussures" ne sont pas dans le lexique : ce sont des formes conjuguées et accordées de mots qui, eux, figurent bien dans le lexique. D'ailleurs, la traduction de "Je chercher mon beau chaussure" est plus satisfaisante : "I look for my beautiful shoe". Passons au niveau 2.

3 Traduction automatique, niveau 2

Pour un mot absent du lexique, on va chercher à traduire, non pas ce mot exactement, mais sa racine (sa forme non conjuguée, non accordée). Dans le répertoire `lib` du projet Eclipse, le fichier `racines.txt` donne pour chaque mot accordé et conjugué un certain nombre d'informations, dont sa racine. Pour exploiter ces informations, utilisez les fonctions `ACX.lectureDico` et `ACX.mots`. La fonction `String[] ACX.mots(String texte)` retourne, à partir d'un `String` contenant des mots séparés par des espaces, tabulations..., le tableau contenant chacun de ces mots, dans l'ordre. Ex : `ACX.mots("blah blah")` retourne {`"blah","blah"`}.

Test de la traduction automatique de niveau 2 En tapant la phrase “Je cherche mes belles chaussures”, la traduction automatique de niveau 2 doit maintenant vous donner “I look for my beautiful shoe”. Par contre si vous copiez-collez dans l'interface un texte important, comme par exemple des passages de `germinalExtrait.txt`, celle-ci met un certain temps à répondre. Que diriez-vous de passer au niveau 3 ?

4 Traduction automatique, niveau 3

Améliorer l'efficacité de la recherche dans les tables associatives en triant les tables de clés et en utilisant la recherche dichotomique à la place de la recherche simple.

5 Traduction automatique, niveau 4

Fixer des bornes sur la taille des tables n'est pas toujours possible. Comment exploiter un fichier lexique dont on ne connaîtrait pas la taille a priori ? Proposez une implantation de la traduction automatique de niveau 3 mais dans laquelle la taille des tables n'est pas fixe et peut évoluer au cours du temps en fonction du nombre d'associations ajoutées dans la table. On attend une solution basée sur les tables d'associations de taille variable et non bornée et non sur les ArrayList de Java.