TP 2 : COMPARAISON DE PHRASES

Introduction

Pour le deuxième tp vous allez construire un logiciel qui lit et compare deux phrases. Cette comparaison va afficher une valeur entre 0 et 1 à l'écran. Une valeur de 0 indique que les deux phrases sont différentes alors qu'une valeur de 1 indique qu'elles sont identiques.

Description

Entrées

Votre logiciel doit lire deux phrases au clavier. Pour cette lecture vous allez utiliser un code similaire au suivant :

```
InputStreamReader isr = new InputStreamReader( System.in );
BufferedReader br = new BufferedReader( isr );

String p = null;

try {
    p = br.readLine();
} catch( IOException e ) {
    // placez vos gestions d'erreur ici.
}
```

Un `InputStreamReader` est construit, il est lié au clavier. Ensuite, il est utilisé pour construire un `BufferedReader` qui sera utilisé pour faire les lectures. Finalement, votre logiciel doit lire une valeur que nous noterons 'distance maximale' (δ). Cette lecture se fait sensiblement de la même façon. Vous devrez transformer la chaîne de caractères en entier :

```
int distance = 0;

try {
    String temp = br.readLine();
    distance = Integer.parseInt( temp );
} catch( IOException e ) {
    // placez vos gestions d'erreur ici.
}
```

La distance que vous lisez doit être soit -1, soit 1 ou plus. Une valeur de -1 indiquera que la distance maximale n'est pas utilisée.

Traitement

Votre programme doit ensuite calculer la similarité entre les deux phrases. Cette similarité est calculée comme suit :

1. Extraire les mots des phrases.
2. Construire les sacs de bigrammes espacés des mots.
3. Trouver l'intersection entre les sacs.
4. Calculer la métrique de similarité.

Extraction

Vous devez premièrement trouver les mots formant les phrases. Un mot est une suite continue de lettre minuscule et/ou majuscule. Tout autre caractère sera considéré comme un séparateur de mots. Par exemple, la phrase p_1 = "*Lors de la construction d'un tableau*" nous donnera les mots suivants :

["*Lors*", "*de*", "*la*", "*construction*", "*d*", "*un*", "*tableau*"]

Bigramme espacé

Vous devez ensuite construire les bigrammes espacés qui se retrouvent dans la phrase. Un bigramme est simplement une paire de mots. Les mots qui composent la paire sont pris dans la phrase et doivent conserver l'ordre qu'ils avaient dans la phrase. Par exemple ("*de*", "*la*") est un bigramme de la phrase. Par contre ("*la*","*de*") n'est pas valide, car les mots ne sont pas dans le même ordre qu'ils ont dans la phrase.

Une autre condition importante pour la formation des bigrammes est la distance entre les mots. Deux mots qui sont séparés par x mots ont une distance de (x+1) entre eux. Par exemple, les mots "*Lors*" et "*construction*" ont 2 mots entre eux ("*de*", "*la*"), donc ils sont à une distance de 2+1=3. Pour la construction des bigrammes espacés vous ne prenez que les mots qui ont une distance inférieure ou égale à δ. Si l'utilisateur avait donné un δ de -1, alors vous prenez tous les bigrammes possibles, peu importe la distance entre les deux mots. Pour notre exemple, si le $\delta = 4$ alors nous obtenons les bigrammes suivants :

s_1 = [("*Lors*", "*de*"), ("*Lors*", "*la*"), ("*Lors*", "*construction*"), ("*de*", "*la*"), ("*de*", "*construction*"), ("*de*", "*d*"), ("*la*", "*construction*"), ("*la*", "*d*"), ("*la*", "*un*"), ("*construction*", "*d*"), ("*construction*", "*un*"), ("*construction*", "*tableau*"), ("*d*", "*un*"), ("*d*", "*tableau*"), ("*un*", "*tableau*")].

Remarquez que si le même mot apparaît plus d'une fois dans la phrase, alors il peut y avoir des bigrammes identiques, il est important de conserver les bigrammes identiques.

Intersection

Ensuite, vous devez trouver l'intersection entre les deux sacs de bigrammes espacés. Utilisons une deuxième phrase p_2 = "*Cette classe va permettre la construction d'un tableau*". Ce qui nous donne les bigrammes suivants (avec un espace maximal $\delta = 4$)

$s_2 =$ [("*Cette*", "*classe*"), ("*Cette*", "*va*"), ("*Cette*", "*permettre*"), ("*classe*", "*va*"), ("*classe*", "*permettre*"), ("*classe*", "*la*"), ("*va*", "*permettre*"), ("*va*", "*la*"), ("*va*", "*construction*"), ("*permettre*", "*la*"), ("*permettre*", "*construction*"), ("*permettre*", "*d*"), ("*la*", "*construction*"), ("*la*", "*d*"), ("*la*", "*un*"), ("*construction*", "*d*"), ("*construction*", "*un*"), ("*construction*", "*tableau*"), ("*d*", "*un*"), ("*d*", "*tableau*"), ("*un*", "*tableau*")]

L'intersection entre les deux sacs de bigrammes est un sac contenants les bigrammes qui se retrouvent dans les deux sacs. Par exemple :

$i = s_1 \cap s_2$= [("*la*", "*construction*"), ("*la*", "*d*"), ("*la*", "*un*"), ("*construction*", "*d*"), ("*construction*", "*un*"), ("*construction*", "*tableau*"), ("*d*", "*un*"), ("*d*", "*tableau*"), ("*un*", "*tableau*")]

Si un bigramme apparaît plus d'une fois alors vous prenez le minimum de fois qu'il apparaît dans une des phrases. Par exemple, si le bigramme ("*d*", "*un*") apparaît 4 fois dans la première phrase et 2 fois dans la deuxième phrase, alors il sera 2 fois (minimum(4, 2) = 2) dans l'intersection. La case des lettres n'est pas importante, c.-à-d. le mot "*tableau*" est équivalent au mot "*Tableau*".

Métrique de similarité

Il ne reste plus qu'à calculer la métrique de similarité avec l'information que nous avons. Vous allez utiliser des **`double`** pour faire ce calcul.

$$u = \frac{|i|}{|s_1|}$$

$$v = \frac{|i|}{|s_2|}$$

$$f = \frac{2uv}{u+v}$$

Sorties

Finalement, affichez le résultat à l'écran, seul, rien d'autre.

Directives

1. Le tp est à faire seul.
2. Vous devez construire des classes appropriées.
3. Commentaire :

   a. Commentez l'entête de chaque classe et méthode. Ces commentaires doivent contenir la description de la méthode et le rôle de ces paramètres.
   b. Une ligne contient soit un commentaire, soit du code, pas les deux.
   c. Utilisez des noms d'identificateur significatif.
   d. Utilisez le français.
4. Code :
   a. Pas de `goto`, `continue`.
   b. Les `break` ne peuvent apparaître que dans les `switch`.
   c. Un seul `return` par méthode.
5. Indentez votre code. Assurez-vous que l'indentation est faite avec des espaces.

Remise

Remettre le tp par l'entremise de Moodle. Placez vos fichiers `'*.java'` dans un dossier compressé de Window, vous devez remettre l'archive. Le tp est à remettre avant le 24 novembre 23 :59.

Évaluation

- Fonctionnalité (8 pts) : des tests partiels vous seront remis. Un test plus complet sera appliqué à votre tp.
- Structure (2 pt) : veillez à utiliser correctement le mécanisme d'héritage et de méthode.
- Lisibilité (3 pts) : commentaire, indentation et noms d'identificateur significatif.