# TP 3 : ÉVALUATION DE CONSTRUCTEUR DE PHRASE

## INTRODUCTION

Pour le troisième TP, nous allons construire une autre version de calcul de similarité (TP 1). Cette fois-ci, vous allez construire un logiciel qui compare deux phrases (en français) afin de décider si elles ont le même contenu (si elles sont similaires). Ce genre de logiciel est utilisé en intelligence artificielle (IA) pour analyser des phrases construites par un ordinateur. Nous voulons comparer une phrase générée avec une phrase cible. La prochaine section va décrire le logiciel. La section suivante donne les directives de construction de votre logiciel et les éléments d'évaluation. La dernière section donne les directives de remise.

## DESCRIPTION

Cette section décrit le logiciel que vous devez construire. Vous y trouverez une description de l'interaction que votre logiciel aura avec l'utilisateur et des tâches que le logiciel doit accomplir.

### ENTRÉES

Votre logiciel doit lire deux phrases ($p_2$ et $p_1$) au clavier. Utilisez la classe `Scanner` de la façon suivante :

```
Scanner sc = new Scanner( System.in );

String phrase1 = sc.nextLine();
String phrase2 = sc.nextLine();

sc.close();
```

Cela va vous donner deux `String` corrects pour le logiciel.

### N-GRAMME

Votre logiciel doit trouver tous les mots (nous allons utiliser le terme 1-gramme pour désigner un mot) de chaque phrase. Un mot est une suite continue de lettre minuscule et/ou majuscule. Tout autre caractère sera considéré comme un séparateur de mots. Votre logiciel va ensuite afficher les mots de la phrase. Par exemple, si la première phrase est "*Lors de la construction d'un tableau*", alors le logiciel **affiche** :

1-grammes de la phrase 1 = ["*Lors*", "*de*", "*la*", "*construction*", "*d*", "*un*", "*tableau*"]

Votre logiciel doit aussi calculer les 2-grammes, les 3-grammes et les 4-grammes de la phrase.

Un $n$-gramme est une séquence de $n$ mots qui se suivent dans la phrase. Donc, pour la phrase précédente, nous avons les $n$-grammes suivant :

2-grammes de la phrase 1 = [ (“*Lors*”, “*de*”), (“*de*”, “*la*”), (“*la*”, “*construction*”), (“*construction*”, “*d*”), (“*d*”, “*un*”), (“*un*”, “*tableau*”) ]

3-grammes de la phrase 1 = [ (“*Lors*”, “*de*”, “*la*”), (“*de*”, “*la*”, “*construction*”), (“*la*”, “*construction*”, “*d*”), (“*construction*”, “*d*”, “*un*”), (“*d*”, “*un*”, “*tableau*”) ]

4-grammes de la phrase 1 = [ (“*Lors*”, “*de*”, “*la*”, “*construction*”), (“*de*”, “*la*”, “*construction*”, “*d*”), (“*la*”, “*construction*”, “*d*”, “*un*”), (“*construction*”, “*d*”, “*un*”, “*tableau*”) ]

Votre logiciel doit **afficher** les 2-, 3- et 4-grammes pour les deux phrases en entrées.

Dans le reste de la description, nous allons utiliser la notation suivante pour désigner l’information :

- $p_1 \Longrightarrow$ La première phrase entrée par l’utilisateur.
- $p_2 \Longrightarrow$ La deuxième phrase entrée par l’utilisateur.
- $p_1.g_1 \Longrightarrow$ Les 1-grammes (mots) de la première phrase.
- $p_2.g_1 \Longrightarrow$ Les 1-grammes (mots) de la deuxième phrase.
- $p_1.g_2 \Longrightarrow$ Les 2-grammes de la première phrase.
- $p_2.g_2 \Longrightarrow$ Les 2-grammes de la deuxième phrase.
- $p_1.g_3 \Longrightarrow$ Les 3-grammes de la première phrase.
- $p_2.g_3 \Longrightarrow$ Les 3-grammes de la deuxième phrase.
- $p_1.g_4 \Longrightarrow$ Les 4-grammes de la première phrase.
- $p_2.g_4 \Longrightarrow$ Les 4-grammes de la deuxième phrase.

## N-GRAMMES COMMUN

Lorsque que vous avez trouvé les 1-, 2-, 3- et 4-grammes pour les deux phrases, il suffit de calculer les valeurs suivantes :

- $|p_1.g_1| \Longrightarrow$ nombre de 1-grammes dans la première phrase.
- $|p_2.g_1| \Longrightarrow$ nombre de 1-grammes dans la deuxième phrase.
- $|p_1.g_2| \Longrightarrow$ nombre de 2-grammes dans la première phrase.
- $|p_2.g_2| \Longrightarrow$ nombre de 2-grammes dans la deuxième phrase.
- $|p_1.g_3| \Longrightarrow$ nombre de 3-grammes dans la première phrase.
- $|p_2.g_3| \Longrightarrow$ nombre de 3-grammes dans la deuxième phrase.
- $|p_1.g_4| \Longrightarrow$ nombre de 4-grammes dans la première phrase.
- $|p_2.g_4| \Longrightarrow$ nombre de 4-grammes dans la deuxième phrase.

Ensuite, il faut calculer le nombre de 1-gramme commun à la première et la deuxième phrase. Donc l’intersection entre les 1-grammes de la première phrase et les 1-gramme de la deuxième phrase.

- $p_1.g_1 \cap p_2.g_1 = \{e | e \in p_1.g_1 \wedge e \in p_2.g_1\}$

La case des lettres n’est pas importante. Donc Lors = LORS = lors. Ensuite, il faut calculer la cardinalité (taille, nombre d’éléments) de cet ensemble :

- $c_1 = |p_1.g_1 \cap p_2.g_1|$

Faire ce calcul pour les 2-, 3- et 4- grammes :

- $c_2 = |p_1.g_2 \cap p_2.g_2|$
- $c_3 = |p_1.g_3 \cap p_2.g_3|$
- $c_4 = |p_1.g_4 \cap p_2.g_4|$

Le logiciel doit **afficher** les valeurs de $c_1$, $c_2$, $c_3$ et $c_4$. Il doit ensuite calculer une valeur que nous appelons le rappel. Le rappel est le pourcentage de n grammes de la deuxième phrase(cible) qui ont été retrouvés dans la première phrase (générée). Nous calculons les rappels suivants (résultat avec des doubles) :

- $r_1 = \frac{c_1}{|p_2.g_1|}$
- $r_2 = \frac{c_2}{|p_2.g_2|}$
- $r_3 = \frac{c_3}{|p_2.g_3|}$
- $r_4 = \frac{c_4}{|p_2.g_4|}$

Le logiciel doit **afficher** les valeurs de $r_1$, $r_2$, $r_3$ et $r_4$. Il doit ensuite calculer une valeur que nous appelons la précision. La précision est le pourcentage de n grammes de la première phrase(généré) qui ont été retrouvés dans la deuxième phrase (cible). Nous calculons les précisions suivantes :

- $q_1 = \frac{c_1}{|p_1.g_1|}$
- $q_2 = \frac{c_2}{|p_1.g_2|}$
- $q_3 = \frac{c_3}{|p_1.g_3|}$
- $q_4 = \frac{c_4}{|p_1.g_4|}$

Le logiciel doit **afficher** les valeurs de $q_1$, $q_2$, $q_3$ et $q_4$. Il doit finalement calculer une valeur que nous appelons le f-mesure qui est une combinaison du rappel et de la précision. Nous calculons les f-mesures suivants :

- $F_1 = 2 \cdot \frac{r_1 \cdot q_1}{r_1 + q_1}$
- $F_2 = 2 \cdot \frac{r_2 \cdot q_2}{r_2 + q_2}$
- $F_3 = 2 \cdot \frac{r_3 \cdot q_3}{r_3 + q_3}$
- $F_4 = 2 \cdot \frac{r_4 \cdot q_4}{r_4 + q_4}$

Le logiciel doit finalement **afficher** les valeurs de $F_1$, $F_2$, $F_3$ et $F_4$. Nous utilisons ces valeurs dans plusieurs domaines pour mesurer la réussite d'un logiciel. Votre programme termine ici.

## RÉSUMÉ DES ENTRÉES SORTIES

En noir, ce qui est affiché par l'ordinateur, en vert, ce qui est écrit par l'utilisateur.

Entrez la phrase générée : Lors de la construction d'un tableau.

Entrez la phrase cible : Lors de la construction d'un logiciel contenant un tableau.

1-grammes de la phrase 1 = ["Lors", "de", "la", "construction", "d", "un", "tableau"]

1-grammes de la phrase 1 = [“Lors”, “de”, “la”, “construction”, “d”, “un”, “tableau”]

2-grammes de la phrase 1 = [(“Lors”, “de”), (“de”, “la”), (“la”, “construction”), (“construction”, “d”), (“d”, “un”), (“un”, “tableau”)]

3-grammes de la phrase 1 = [(“Lors”, “de”, “la”), (“de”, “la”, “construction”), (“la”, “construction”, “d”), (“construction”, “d”, “un”), (“d”, “un”, “tableau”)]

4-grammes de la phrase 1 = [(“Lors”, “de”, “la”, “construction”), (“de”, “la”, “construction”, “d”), (“la”, “construction”, “d”, “un”), (“construction”, “d”, “un”, “tableau”)]

1-grammes de la phrase 2 = [“Lors”, “de”, “la”, “construction”, “d”, “un”, “logiciel”, “contenant”, “un”, “tableau”]

2-grammes de la phrase 2 = [(“Lors”, “de”), (“de”, “la”), (“la”, “construction”), (“construction”, “d”), (“d”, “un”), (“un”, “logiciel”), (“logiciel”, “contenant”), (“contenant”, “un”), (“un”, “tableau”)]

3-grammes de la phrase 2 = [(“Lors”, “de”, “la”), (“de”, “la”, “construction”), (“la”, “construction”, “d”), (“construction”, “d”, “un”), (“d”, “un”, “logiciel”), (“un”, “logiciel”, “contenant”), (“logiciel”, “contenant”, “un”), (“contenant”, “un”, “tableau”)]

4-grammes de la phrase 2 = [(“Lors”, “de”, “la”, “construction”), (“de”, “la”, “construction”, “d”), (“la”, “construction”, “d”, “un”), (“construction”, “d”, “un”, “logiciel”), (“d”, “un”, “logiciel”, “contenant”), (“un”, “logiciel”, “contenant”, “un”), (“logiciel”, “contenant”, “un”, “tableau”)]

c1 = 7, c2 = 6, c3 = 4, c4 = 3.

r1 = 0.7, r2 = 0.6666666666666666, r3 = 0.5, r4 = 0.42857142857142855.

q1 = 1.0, q2 = 1.0, q3 = 0.8, q4 = 0.75.

F1 = 0.8235294117647058, F2 = 0.8, F3 = 0.6153846153846154, F4 = 0.5454545454545454.

## CONSTRUCTION, DIRECTIVE ET ÉVALUATION

### CONSTRUCTION

Vous devez construire une classe pour les n-grammes. Utilisez le moins de méthode ‘static’ possible (2 maximum, en comptant le ‘main’).

### DIRECTIVES

1. Le TP est à faire seul ou en équipe de deux.
2. Code :
    a. Pas de `goto`, `continue`.
    b. Les `break` ne peuvent apparaitre que dans les `switch`.
    c. Pour une méthode :

        i. Un seul `return`.
        ii. Additionnez le nombre de `if`, `for`, `while`, `switch` et `try`. Ce nombre ne doit pas dépasser 5.
3. Indentez votre code. Assurez-vous que l'indentation est faite avec des espaces.
4. Commentaires
    - Commentez l'entête de chaque classe et méthode.
    - Une ligne contient soit un commentaire, soit du code, pas les deux.
    - Utilisez des noms d'identificateur significatif.
    - Une ligne de commentaire ne devrait pas dépasser 80 caractères. Continuez sur la ligne suivante au besoin.
    - Nous utilisons Javadoc :
        - La première ligne d'un commentaire doit contenir une description courte (1 phrase) de la méthode ou la classe.
            - Courte.
            - Complète.
            - Commencez la description avec un verbe.
            - Assurez-vous de ne pas simplement répéter le nom de la méthode, donnez plus d'information.
        - Ensuite, au besoin, une description détaillée de la méthode ou classe va suivre.
            - Indépendant du code. Les commentaires d'entêtes décrivent ce que la méthode fait, ils ne décrivent pas comment c'est fait.
            - Si vous avez besoin de mentionner l'objet courant, utilisez le mot 'this'.
        - Ensuite, avant de placer les **tags,** placez une ligne vide.
        - Placez les **tag** @param, @return et @throws au besoin.
            - @param : décris les valeurs acceptées pour la méthode.
        - Dans les commentaires, placer les noms de variable et autre ligne de code entre les tags <code> … </code>.
        - Écrivez les commentaires à la troisième personne, EN FRANÇAIS.

## ÉVALUATION

- Fonctionnalité (13 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 (2 pts) : commentaire, indentation et noms d'identificateur significatif.

## REMISE

Remettre le TP par l'entremise de Moodle. Placez vos fichiers `'*.java'` dans un dossier compressé, vous devez remettre l'archive. Le TP est à remettre avant le 26 avril 23 :59.