LOG3430 - Méthodes de test et de validation du logiciel

Laboratoire 5

Tests Methamorphiques

Département de génie informatique et de génie logiciel
École Polytechnique de Montréal

Automne 2020

1 Introduction

Dans ce travail pratique vous allez effectuer les jeux des tests métamorphiques pour tester le système RENEGE.

2 Objectifs

L'objectif principal de ce laboratoire est :

1. Pratiquer la conception des tests métamorphiques.

3 Mise en contexte théorique

Une des techniques utilisée pour tester des programmes sans oracle consiste à utiliser un «pseudo-oracle» (Davis et Weyuker, 1981) [1], dans lequel plusieurs implémentations d'un algorithme traitent la même entrée et les résultats sont comparés. Cependant, ce n'est pas toujours possible, car il n'y a pas toujours plusieurs implémentations.

Cependant, même sans l'implémentations multiples, les applications présentent souvent des propriétés telles que si l'entrée est modifiée d'une certaine manière, il est possible de prédire la nouvelle sortie, étant donné la sortie d'origine. Cette approche est connue comme les tests métamorphiques. Les tests métamorphiques étaient proposés par Chen et al (1998) [2].

L'idée est simple : même si nous ne connaissons pas la sortie correcte d'une seule entrée, nous pouvons connaître les relations entre les sorties correspondant aux différentes entrées. Nous pouvons vérifier le logiciel pour ces relations, appelées propriétés métamorphiques. S'ils ne tiennent pas, c'est un signe que le logiciel présente des défauts.

Tests métamorphiques

Dans ce travail pratique on va tester les propriétés métamorphiques pour le système RENEGE : la performance du système ne change pas

1. après changement de l'ordre des e-mails dans le "train dataset" ;
2. après changement de l'ordre des e-mails dans le "test dataset" ;
3. après changement de l'ordre des mots dans le "train dataset" ;
4. après changement de l'ordre des mots dans le "test dataset" ;
5. après l'ajout des mêmes e-mails dans le "train dataset" ;
6. après l'ajout des mêmes e-mails dans le "test dataset" ;
7. après l'ajout du "bruit" dans le "train dataset" ;
8. apres l'ajout du bruit dans "test dataset".

1. `https://dl.acm.org/doi/10.1145/800175.809889`
2. `https://arxiv.org/abs/2002.12543`

Remarques

1. Pour caractériser la performance de notre système, on va utiliser telle métrique comme "accuracy". Pour calculer l'accuracy il faut utiliser la configuration du système comme dans le TP1.
2. Pour changer l'ordre des e-mails vous pouvez utiliser la méthode random.shuffle() de Python. changer l'ordre des mots, il faut tout d'abord les "tokenizer" (transformer dans une liste). Vous pouvez utiliser la fonction split() du Python. Pour "detokenizer" vous pouvez utiliser la méthode .join() du Python.
3. "L'ajout des mêmes e-mails" cela veut dire que si vous avez 300 mails, il faut doubler ces 300 mails pour obtenir 600 mails. Donc dans ces 600 mails vous allez avoir 2 copies de chaque mail. Dans le cas de train dataset vous allez avoir un dataset de 1400 mails et pour test dataset - 600 mails.
4. "L'ajout du bruit" signifie ajout de nouveaux mots.On ajoute les nouveaux mots seulement si l'objet ou le corps du message contient plus des 10 mots. Il faut ajouter 10% de la quantité totale des mots dans le corps/sujet de message. Donc, si le corps contient 100 mots, il faut ajouter 10 mots nouveaux comme bruit. Vous allez choisir les mots d'une liste qu'on vous fournit (words.txt) et qui contient 100 mots. Vous pouvez utiliser la fonction random.choice() du Python pour choisir les mots.

Les tâches

La tâche principale est avec l'outil Unittest créé un jeu de tests métamorphiques pour vérifier les propriétés métamorphiques du votre système.

1. Créer 8 tests métamorphiques dans le fichier test_main.py. Chaque test doit correspondre à une chaque propriété c.a.d. il faut nommer les tests comme "test_property1", "test_property2", etc.
2. Il faut créer les fichiers json pour chaque transformation métamorphique réalisé. Plus précisément :
   - Pour changement d'ordre des e-mails : fichiers "train700_mails.json" et "test300_mails.json".
   - Pour changement d'ordre des mots : fichiers "train700_words.json" et "test300_words.json".
   - Pour le doublement des e-mails : fichiers "train700x2.json" et "test300x2.json".
   - Pour l'ajout du bruit : fichiers "train700_noise.json" et "test300_noise.json".
3. Fournir un tableau avec "accuracy" initiale et "accuracy" après chaque test.
4. Vous allez utiliser les fichiers json avec transformations métamorphiques pour effectuer les tests. Pour que le test passe il faut que la différence entre l'accuracy initiale et accuracy post-métamorphique ne surpasse 3%.
5. Si certains de vos tests ne passe pas, essayez de trouver les défauts dans votre système. Si vous ne réussissez pas, précisez quels peuvent être les sources potentielles des bogues.
6. Pour les tests qui passent, expliquez quelles différences vous remarquez dans la performance du votre système.
7. Finalement, selon vous, quelles autres transformations métamorphiques peuvent être utilisées pour tester le système.

4 Livrables attendus

Les livrables suivants sont attendus :

- Un rapport pour le laboratoire [4 points]. Dans le rapport il faut inclure le tableau des résultats et aussi votre analyse des résultats. Avez vous trouve les défauts ?
- Le dossier contenant tous les modules du RENEGE, test_main.py et fichier json avec transformations. [16 points].

Le tout à remettre dans une seule archive **zip** avec pour nom matricule1_matricule2_lab1.zip à téléverser sur Moodle.

Le rapport doit contenir le titre et numéro du laboratoire, les noms et matricules des coéquipiers ainsi que le numéro du groupe.

Consultez le site Moodle du cours pour la date et l'heure limites de remise des fichiers. Un retard de]0,24h] sera pénalisé de 10%, de]24h, 48h] de 20% et de plus de 48h de 50%.