TP1 : Grands domaines de l'IA et du GL et leurs interactions

Mahi, BEGOUG

mahi.begoug@etu.umontpellier.fr

Ibrahim, BERKANE

ibrahim-abouyatime.berkane@etu.umontpellier.fr

19 September 2021

1 Introduction

Nous avons choisi l'article [1] portant sur la suppression automatiques des commentaires TODO dans un code source.

Les commentaires TODO sont utilisés lors de la phase d'implémentation de logiciel pour décrire les tâches à développer. Néanmoins, après que les tâches ont été effectuées, les développeurs oublient quelquefois de supprimer ces commentaires ce qui produit des commentaires TODO obsolètes qui peuvent perturber les équipes de développement. Par exemple, des nouveaux développeurs intègrent l'équipe de développement alors qu'ils peuvent changer le code qu'était déjà développé, ce qui augmente la probabilité d'introduire des bogues à l'avenir et de réduire la qualité du logiciel.

2 Problématique

Les questions de recherche posées par Zhipeng Gao et al [1] sont :

1. Comment détecter les commentaires TODO obsolètes ?
2. Comment les supprimer automatiquement avant qu'ils ne mènent à des effets indésirables pour le fonctionnement du logiciel?

3 Objectifs

L'objectif visée par Zhipeng Gao et al [1] est d'appliquer les réseaux de neurones pour identifier les commentaires TODO obsolètes et de les supprimer automatiquement dans les logiciels.

4 Solution Proposée

Zhipeng Gao et al [1] proposent un modèle de classification binaire pour modéliser la problématique. Pour chaque validation du code (commit), le changement du code extrait du fichier diff est nommé C, le message de commit est nommé M, le commentaire TODO associé aux changements du code est T ,et S est le statut de TODO commentaire (si S positive ,TODO commentaire est résolu sinon il n'est pas résolu).

D'un point de vue statistique, l'objectif est d'entraîner le modèle θ en utilisant le triple {C,T,M} pour maximiser la probabilité conditionnel P(S|{C,T,M}), et d'un point de vue mathématique l'objectif est de déterminer la fonction y tel que y = argmaxP(S|{C,T,M}). La solution proposée est réalisé en deux étapes, illustré par la Figure 1 :

1. Identification des commentaires TODO résolus et non résolus a une difficulté est que le code change et les commentaires TODO sont de types différents (code source vs langage naturel) et ne peuvent pas être facilement mappés par simple correspondance de leurs tokens lexicaux. La solution est d'introduire trois encodeurs utilisant BERT encoder : l'encodeur de commentaire TODO, l'encodeur de changement de code et l'encodeur de message de commit.

2. Pour saisir les relations entre le vecteur de changement du code, le commentaire TODO et le message de commit, il est nécessaire de lier et de fusionner leurs informations. Pour ce faire, un Multi-Layer Perceptron (MLP) est ajouté pour répondre à ce besoin, qui est partagé avec les trois encodeurs. Le MLP prend les représentations contextualisées (c'est-à-dire hc, ht et hm) en entrée et génère la probabilité de l'état final S = 0, 1, et la fonction sigmoid :

$$\sigma(x) = 1/(1 + e^{-x}) \tag{1}$$

qui produira la probabilité finale du statut S entre 0 et 1.

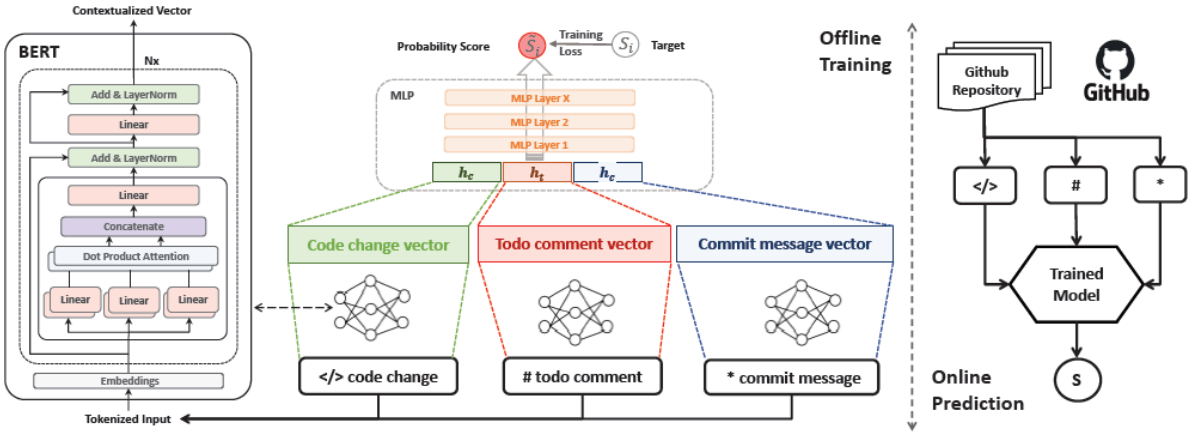

Figure 1: Processus de l'approche

5 Collection du jeu de données

D'abord, ils ont collecté les commentaires TODO obsolètes des 10 000 premiers dépôts Github, en particulier, dans les projets développés en Java et Python selon leurs nombres des étoiles [2]. Le processus de collection comprend les étapes suivantes :

1. Identification des commits liées à TODO.
2. Normalisation la différence (Diff) et le message de validation (Commit).
3. Extraction de commentaires TODO.
4. Étiquetage des données.
5. Vérification manuelle.
6. Fractionnement des données.

6 Conclusion (Évaluation de l'approche et futur work)

Ils ont testé leurs approches sur le jeu de données. Ils ont utilisé python pour l'implémenter [3]. Après, ils ont fait une synthèse que l'approche proposée a démontré son efficacité et ses performances prometteuses. A l'avenir, Ils prévoient d'étudier l'efficacité de leur approche par rapport à d'autres langages de programmation et de l'adapter à d'autres types de commentaires tels que FIXME, HACK et d'autres.

Bibliographie

[1] Zhipeng Gao et al. "Automating the Removal of Obsolete TODO Comments". In: *arXiv preprint arXiv:2108.05846* (2021).

Webographie

[2] Zhipeng Gao et al. *Data set*. URL: `https://github.com/beyondacm/TDClearner/tree/main/core_code/data`.

[3] Zhipeng. *Implémentation de l'outil*. URL: `https://github.com/beyondacm/TDClearner`.