Université De Montpellier
Faculté Des Sciences

Niveau : Master 2

Module : IA pour le Génie Logiciel

HAI916I

TP : Code2Vec

Supervisé par :
M. Marianne Huchard
M. Chouki Tibermacine

Réalisé par :

Allouch Yanis
Kaci Ahmed

2021/2022

Table des matières

1 Réalisation de la *Step 4* du dépôt git

Après avoir cloné le projet Git énoncé dans le sujet de TP et téléchargé le modèle (java14m_data.tar.gz) servant à l'utilisation de code2vec. Nous avons créé trois fichiers Input.java (voir Listing 1, Listing 3 et Listing 5) basé sur des projets personnels antérieurs.

Ensuite, nous avons suivis le workflow avec le modèle entrainé sans paramètre (Step 4) et nous avons obtenu les résultats suivant que nous allons présenter, dans le Listing 1, Listing 3 et Listing 5. (ces derniers ont été tronqués de leurs en-têtes pour améliorer leur lisibilité).

```java
private boolean isJavaFile(File file) {

  final String extentionWanted = ".java";
  int extentionIndex = file.getName().length() - 5;
  int endFileIndex = file.getName().length();
  final String fileExtention = file.getName().substring(
   extentionIndex, endFileIndex);

  return fileExtention.equals(extentionWanted);
}
```

Listing 1 – Première Input.java

```
Starting interactive prediction...
Modify the file: "Input.java" and press any key when ready, or "q"
    / "quit" / "exit" to exit
Original name:  is|java|file
  (0.335654) predicted: ['save', 'to', 'file']
  (0.144616) predicted: ['compare']
  (0.123919) predicted: ['request', 'overwrite', 'file']
  (0.095342) predicted: ['on', 'file', 'deletion']
  (0.064893) predicted: ['is', 'selected']
  (0.064431) predicted: ['check', 'file']
  (0.052602) predicted: ['get', 'local', 'path']
  (0.052427) predicted: ['get', 'project', 'path']
  (0.033171) predicted: ['is', 'mine']
  (0.032946) predicted: ['verify']
Attention:
0.075346  context: file,(VariableDeclaratorId0)^(Parameter)^(
    MethodDeclaration)_(BlockStmt)_(ReturnStmt)_(MethodCallExpr0)_(
    NameExpr0),fileextention
0.065303  context: file,(ClassOrInterfaceType1)^(Parameter)^(
    MethodDeclaration)_(BlockStmt)_(ReturnStmt)_(MethodCallExpr0)_(
    NameExpr3),equals
0.057388  context: file,(ClassOrInterfaceType1)^(Parameter)^(
    MethodDeclaration)_(BlockStmt)_(ReturnStmt)_(MethodCallExpr0)_(
    NameExpr0),fileextention
0.050228  context: file,(VariableDeclaratorId0)^(Parameter)^(
    MethodDeclaration)_(BlockStmt)_(ReturnStmt)_(MethodCallExpr0)_(
    NameExpr3),equals
```

```
0.046385  context: file,(VariableDeclaratorId0)^(Parameter)^(
    MethodDeclaration)_(BlockStmt)_(ExpressionStmt)_(
    VariableDeclarationExpr)_(VariableDeclarator)_(
    StringLiteralExpr1),java
0.044308  context: file,(ClassOrInterfaceType1)^(Parameter)^(
    MethodDeclaration)_(BlockStmt)_(ExpressionStmt)_(
    VariableDeclarationExpr)_(VariableDeclarator)_(
    StringLiteralExpr1),java
0.042243  context: METHOD_NAME,(NameExpr1)^(MethodDeclaration)_(
    BlockStmt)_(ReturnStmt)_(MethodCallExpr0)_(NameExpr3),equals
0.040910  context: boolean,(PrimitiveType0)^(MethodDeclaration)_(
    Parameter)_(ClassOrInterfaceType1),file
0.035646  context: substring,(NameExpr4)^(MethodCallExpr)^(
    VariableDeclarator)^(VariableDeclarationExpr)^(ExpressionStmt)^(
    BlockStmt)_(ReturnStmt)_(MethodCallExpr0)_(NameExpr0),
    fileextention
0.030011  context: METHOD_NAME,(NameExpr1)^(MethodDeclaration)_(
    BlockStmt)_(ReturnStmt)_(MethodCallExpr0)_(NameExpr0),
    fileextention
Modify the file: "Input.java" and press any key when ready, or "q"
    / "quit" / "exit" to exit
```

Listing 2 – Résultat de l'input n°1

```java
public List<String> getJavaFiles() {
        File directory = new File(path);
        return parser.getFilesPaths(directory);
}
```

Listing 3 – Second Input.java

```
Starting interactive prediction...
Modify the file: "Input.java" and press any key when ready, or "q"
    / "quit" / "exit" to exit
Original name:  get|java|files
  (0.516548) predicted: ['get', 'directory']
  (0.167017) predicted: ['get', 'index', 'path']
  (0.128806) predicted: ['get', 'path']
  (0.034596) predicted: ['read', 'all']
  (0.031786) predicted: ['get', 'source', 'path']
  (0.029946) predicted: ['get', 'sub', 'directory']
  (0.025149) predicted: ['is', 'directory']
  (0.023115) predicted: ['list', 'content']
  (0.021662) predicted: ['get', 'affected', 'paths']
  (0.021377) predicted: ['create', 'path']
Attention:
0.147383  context: string,(ClassOrInterfaceType0)^(
    ClassOrInterfaceType)^(MethodDeclaration)_(BlockStmt)_(
    ExpressionStmt)_(VariableDeclarationExpr)_(VariableDeclarator)_(
    ObjectCreationExpr)_(NameExpr1),path
0.124728  context: path,(NameExpr1)^(ObjectCreationExpr)^(
    VariableDeclarator)^(VariableDeclarationExpr)^(ExpressionStmt)^(
    BlockStmt)_(ReturnStmt)_(MethodCallExpr0)_(NameExpr0),parser
```

```
0.092909  context: path,(NameExpr1)^(ObjectCreationExpr)^(
    VariableDeclarator)^(VariableDeclarationExpr)^(ExpressionStmt)^(
    BlockStmt)_(ReturnStmt)_(MethodCallExpr0)_(NameExpr2),directory
0.090906  context: METHOD_NAME,(NameExpr1)^(MethodDeclaration)_(
    BlockStmt)_(ReturnStmt)_(MethodCallExpr0)_(NameExpr0),parser
0.071261  context: METHOD_NAME,(NameExpr1)^(MethodDeclaration)_(
    BlockStmt)_(ReturnStmt)_(MethodCallExpr0)_(NameExpr2),directory
0.067422  context: string,(ClassOrInterfaceType0)^(
    ClassOrInterfaceType)^(MethodDeclaration)_(BlockStmt)_(
    ReturnStmt)_(MethodCallExpr0)_(NameExpr3),getfilespaths
0.062629  context: string,(ClassOrInterfaceType0)^(
    ClassOrInterfaceType)^(MethodDeclaration)_(BlockStmt)_(
    ReturnStmt)_(MethodCallExpr0)_(NameExpr0),parser
0.055047  context: string,(ClassOrInterfaceType0)^(
    ClassOrInterfaceType)^(MethodDeclaration)_(BlockStmt)_(
    ReturnStmt)_(MethodCallExpr0)_(NameExpr2),directory
0.043602  context: string,(ClassOrInterfaceType0)^(
    ClassOrInterfaceType)^(MethodDeclaration)_(NameExpr1),
    METHOD_NAME
0.032642  context: string,(ClassOrInterfaceType0)^(
    ClassOrInterfaceType)^(MethodDeclaration)_(BlockStmt)_(
    ExpressionStmt)_(VariableDeclarationExpr)_(VariableDeclarator)_(
    VariableDeclaratorId0),directory
Modify the file: "Input.java" and press any key when ready, or "q"
    / "quit" / "exit" to exit
```

Listing 4 – Résultat du second input

```java
public class MyApplicationForAddition {

        public static void main(String[] args) {
                int a = getA();
                int b = getB();

                int result = getResultAddition(a,b);

                System.out.println("Le resultat est " + result);
        }

        int getA(){
                return 40;
        }

        int getB(){
                return 2;
        }

        int getResultAddition(int n, int m){
                return n + m;
        }
}
```

Listing 5 – Troisème Input.java

```
Starting interactive prediction...
Modify the file: "Input.java" and press any key when ready, or "q" / "quit" / "exit" to exit
Original name:  main
  (0.996219) predicted: ['main']
  (0.002518) predicted: ['foo']
  (0.000635) predicted: ['method']
  (0.000240) predicted: ['extracted']
  (0.000185) predicted: ['bar']
  (0.000113) predicted: ['f']
  (0.000038) predicted: ['function']
  (0.000023) predicted: ['r']
  (0.000015) predicted: ['m']
  (0.000015) predicted: ['a']
Attention:
0.132082  context: args,(VariableDeclaratorId0)^(Parameter)^(MethodDeclaration)_(BlockStmt)_(ExpressionStmt)_(VariableDeclarationExpr)_(VariableDeclarator)_(MethodCallExpr1)_(NameExpr1),getb
0.081608  context: args,(VariableDeclaratorId0)^(Parameter)^(MethodDeclaration)_(BlockStmt)_(ExpressionStmt)_(VariableDeclarationExpr)_(VariableDeclarator)_(MethodCallExpr1)_(NameExpr1),geta
0.066656  context: args,(VariableDeclaratorId0)^(Parameter)_(ArrayBracketPair2),[]
0.061893  context: [],(ArrayBracketPair2)^(Parameter)^(MethodDeclaration)_(BlockStmt)_(ExpressionStmt)_(VariableDeclarationExpr)_(VariableDeclarator)_(MethodCallExpr1)_(NameExpr1),getb
0.044478  context: args,(VariableDeclaratorId0)^(Parameter)^(MethodDeclaration)_(BlockStmt)_(ExpressionStmt)_(VariableDeclarationExpr)_(VariableDeclarator)_(MethodCallExpr1)_(NameExpr3),getresultaddition
0.036897  context: [],(ArrayBracketPair2)^(Parameter)^(MethodDeclaration)_(BlockStmt)_(ExpressionStmt)_(MethodCallExpr0)_(FieldAccessExpr0)_(NameExpr0),system
0.032848  context: [],(ArrayBracketPair2)^(Parameter)^(MethodDeclaration)_(BlockStmt)_(ExpressionStmt)_(MethodCallExpr0)_(BinaryExpr:plus)_(StringLiteralExpr0),lersultatest
0.031300  context: METHOD_NAME,(NameExpr1)^(MethodDeclaration)_(BlockStmt)_(ExpressionStmt)_(VariableDeclarationExpr)_(VariableDeclarator)_(MethodCallExpr1)_(NameExpr1),getb
0.031273  context: [],(ArrayBracketPair2)^(Parameter)^(MethodDeclaration)_(BlockStmt)_(ExpressionStmt)_(VariableDeclarationExpr)_(VariableDeclarator)_(MethodCallExpr1)_(NameExpr1),geta
0.026797  context: [],(ArrayBracketPair2)^(Parameter)^(MethodDeclaration)_(BlockStmt)_(ExpressionStmt)_(MethodCallExpr0)_(FieldAccessExpr0)_(NameExpr2),out
Original name:  get|a
  (0.791144) predicted: ['get', 'pulse']
  (0.108399) predicted: ['get', 'height']
  (0.030621) predicted: ['get', 'icon', 'height']
  (0.018270) predicted: ['get', 'digest', 'size']
  (0.013907) predicted: ['get', 'row', 'height']
  (0.011284) predicted: ['get', 'icon', 'width']
  (0.008696) predicted: ['get', 'priority']
  (0.007444) predicted: ['get', 'default', 'height']
  (0.005342) predicted: ['get', 'depth']
  (0.004893) predicted: ['get', 'initial', 'animation', 'time']
Attention:
0.359031  context: int,(PrimitiveType0)^(MethodDeclaration)_(
    BlockStmt)_(ReturnStmt)_(IntegerLiteralExpr0),40
0.321949  context: int,(PrimitiveType0)^(MethodDeclaration)_(
    NameExpr1),METHOD_NAME
0.319020  context: METHOD_NAME,(NameExpr1)^(MethodDeclaration)_(
    BlockStmt)_(ReturnStmt)_(IntegerLiteralExpr0),40
Original name:  get|b
  (0.245504) predicted: ['get', 'number', 'of', 'operands']
  (0.209002) predicted: ['get', 'column', 'count']
  (0.180355) predicted: ['get', 'dimension']
  (0.122531) predicted: ['arity']
  (0.087040) predicted: ['get', 'target', 'dimensions']
  (0.086154) predicted: ['get', 'source', 'dimensions']
  (0.021641) predicted: ['get', 'data', 'size']
  (0.019133) predicted: ['get', 'view', 'type', 'count']
  (0.016192) predicted: ['get', 'default', 'index']
  (0.012449) predicted: ['get', 'container', 'columns']
Attention:
0.406867  context: METHOD_NAME,(NameExpr1)^(MethodDeclaration)_(
    BlockStmt)_(ReturnStmt)_(IntegerLiteralExpr0),2
0.360656  context: int,(PrimitiveType0)^(MethodDeclaration)_(
    NameExpr1),METHOD_NAME
0.232476  context: int,(PrimitiveType0)^(MethodDeclaration)_(
    BlockStmt)_(ReturnStmt)_(IntegerLiteralExpr0),2
Original name:  get|result|addition
  (0.660863) predicted: ['m']
  (0.296608) predicted: ['add']
  (0.009221) predicted: ['fred']
  (0.008630) predicted: ['sum']
  (0.007434) predicted: ['method']
  (0.004702) predicted: ['inc']
  (0.004635) predicted: ['n']
  (0.003026) predicted: ['mod']
  (0.003025) predicted: ['step']
  (0.001857) predicted: ['duplicate']
Attention:
0.130732  context: int,(PrimitiveType0)^(MethodDeclaration)_(
    NameExpr1),METHOD_NAME
0.123291  context: m,(VariableDeclaratorId0)^(Parameter)^(
    MethodDeclaration)_(BlockStmt)_(ReturnStmt)_(BinaryExpr:plus)_(
    NameExpr0),n
0.084760  context: int,(PrimitiveType1)^(Parameter)^(
    MethodDeclaration)_(BlockStmt)_(ReturnStmt)_(BinaryExpr:plus)_(
    NameExpr1),m
0.084211  context: int,(PrimitiveType1)^(Parameter)^(
    MethodDeclaration)_(BlockStmt)_(ReturnStmt)_(BinaryExpr:plus)_(
    NameExpr0),n
0.069463  context: n,(VariableDeclaratorId0)^(Parameter)^(
    MethodDeclaration)_(Parameter)_(VariableDeclaratorId0),m
0.063667  context: m,(VariableDeclaratorId0)^(Parameter)^(
    MethodDeclaration)_(BlockStmt)_(ReturnStmt)_(BinaryExpr:plus)_(
    NameExpr1),m
0.054777  context: int,(PrimitiveType0)^(MethodDeclaration)_(
    Parameter)_(VariableDeclaratorId0),n
0.047494  context: n,(VariableDeclaratorId0)^(Parameter)^(
    MethodDeclaration)_(BlockStmt)_(ReturnStmt)_(BinaryExpr:plus)_(
    NameExpr0),n
0.045244  context: n,(VariableDeclaratorId0)^(Parameter)^(
    MethodDeclaration)_(BlockStmt)_(ReturnStmt)_(BinaryExpr:plus)_(
    NameExpr1),m
0.042350  context: int,(PrimitiveType0)^(MethodDeclaration)_(
    Parameter)_(PrimitiveType1),int
Modify the file: "Input.java" and press any key when ready, or "q"
    / "quit" / "exit" to exit
```
Listing 6 – Résultat du troisième input

2 Récupération des *Attention vector*

Pour récupérer de façon programmatique les AST paths et leurs poids nous avons réalisé le script bash suivant (voir Listing 7) :

```bash
1  export CODE2VEC_PATH=code2vec.py
2  export MODELS_PATH=models/java14_model/saved_model_iter8.release
3  export RAW_OUTPUT_PATH=output.out
4  export ERR_LOG_PATH=err.log
5  export OUTPUT_PATH=ast_path.out
6  export TIME2SLEEP_IN_SECONDS=60
7
8  echo "Working"
9  echo -ne '\n' | python3 $CODE2VEC_PATH --load $MODELS_PATH --predict > $RAW_OUTPUT_PATH 2> $ERR_LOG_PATH &
10 echo "Waiting for code2vec..." && sleep $TIME2SLEEP_IN_SECONDS
11 echo "Code2Vec has Finished!"
12 echo "Processing " $RAW_OUTPUT_PATH
13 grep -E "context:|Attention:|Original name:" $RAW_OUTPUT_PATH > $OUTPUT_PATH
14 echo "Saved result in " $OUTPUT_PATH
15 echo "Done."
```

Listing 7 – Script de récupération des AST-Paths

Pour utiliser ce script, il suffit de paramétrer les variables de la ligne 1 à 6. Puis, de l'exécuter depuis un terminal Bash. Nous avons essayé notre script dans la version suivante de Bash (voir Listing 8) :

```
GNU bash, version 5.0.17(1)-release (x86_64-pc-linux-gnu)
```

Listing 8 – Version de bash

Résultats

Le listing suivant (Listing 9) illustre le contenu du fichier résultant de la récupération des attentions vector, émanant de l'exécution de l'input.java présenté dans le Listing 3 :

```
Original name:  get|java|files
Attention:
0.147383  context: string,(ClassOrInterfaceType0)^(ClassOrInterfaceType)^(MethodDeclaration)_(BlockStmt)_(ExpressionStmt)_(VariableDeclarationExpr)_(VariableDeclarator)_(ObjectCreationExpr)_(NameExpr1),path
0.124728  context: path,(NameExpr1)^(ObjectCreationExpr)^(VariableDeclarator)^(VariableDeclarationExpr)^(ExpressionStmt)^(BlockStmt)_(ReturnStmt)_(MethodCallExpr0)_(NameExpr0),parser
0.092909  context: path,(NameExpr1)^(ObjectCreationExpr)^(VariableDeclarator)^(VariableDeclarationExpr)^(ExpressionStmt)^(BlockStmt)_(ReturnStmt)_(MethodCallExpr0)_(NameExpr2),directory
```

```
0.090906  context: METHOD_NAME,(NameExpr1)^(MethodDeclaration)_(
    BlockStmt)_(ReturnStmt)_(MethodCallExpr0)_(NameExpr0),parser
0.071261  context: METHOD_NAME,(NameExpr1)^(MethodDeclaration)_(
    BlockStmt)_(ReturnStmt)_(MethodCallExpr0)_(NameExpr2),directory
0.067422  context: string,(ClassOrInterfaceType0)^(
    ClassOrInterfaceType)^(MethodDeclaration)_(BlockStmt)_(
    ReturnStmt)_(MethodCallExpr0)_(NameExpr3),getfilespaths
0.062629  context: string,(ClassOrInterfaceType0)^(
    ClassOrInterfaceType)^(MethodDeclaration)_(BlockStmt)_(
    ReturnStmt)_(MethodCallExpr0)_(NameExpr0),parser
0.055047  context: string,(ClassOrInterfaceType0)^(
    ClassOrInterfaceType)^(MethodDeclaration)_(BlockStmt)_(
    ReturnStmt)_(MethodCallExpr0)_(NameExpr2),directory
0.043602  context: string,(ClassOrInterfaceType0)^(
    ClassOrInterfaceType)^(MethodDeclaration)_(NameExpr1),
    METHOD_NAME
0.032642  context: string,(ClassOrInterfaceType0)^(
    ClassOrInterfaceType)^(MethodDeclaration)_(BlockStmt)_(
    ExpressionStmt)_(VariableDeclarationExpr)_(VariableDeclarator)_(
    VariableDeclaratorId0),directory
```

Listing 9 – Résultat de récupération des AST-Paths et leur poids

3 Question ouverte

Question :

— Est-ce pertinent d'utiliser les AST paths pour identifier des bad-smells si par exemple il n'y a pas de poids dominant (méthode avec plusieurs fonctionnalités) ?

Après avoir testées sur plusieurs exemples, nous avons constamment obtenu un poids dominant. Étant dans l'incertitude sur le fait d'avoir essayé tous les cas possibles, nous nous sommes documenté grâce aux articles présentés dans la bibliographie de ce rapport et nous avons abouti à la conclusion suivante.
En se basant sur les résultats des articles «Recommendation of Move Method Refactoring Using Path-Based Representation of Code» [4] et «Automatic Detection of Architectural Bad Smells through Semantic Representation of Code» [5], nous pouvons avancer que l'utilisation des AST-paths est pertinentes dans l'identification des bads smells [3]. En effet, l'utilisation du deep learning dans ce domaine offre de meilleurs résultats que les méthodes, traditionnelles, basées sur l'utilisation de métrique. Cependant, l'article «Evaluating the Effectiveness of Code2Vec for Bug Prediction When Considering That Not All Bugs Are the Same» [2] indique que leur utilisation ne fournit pas tout le temps des résultats parfait et qu'il existe des domaines d'amélioration qui pourrait être appliqué aux modèles d'apprentissage sémantique à l'avenir.

Références bibliographiques

[1] Uri Alon et al. « Code2Vec : Learning Distributed Representations of Code ». In : *Proc. ACM Program. Lang.* 3.POPL (jan. 2019), 40 :1-40 :29. issn : 2475-1421. doi : 10.1145/3290353. url : http://doi.acm.org/10.1145/3290353.

[2] Kilby Baron. « Evaluating the Effectiveness of Code2Vec for Bug Prediction When Considering That Not All Bugs Are the Same ». Mém. de mast. University of Waterloo, 2020.

[3] Martin Fowler. *Refactoring : improving the design of existing code*. Addison-Wesley Professional, 2018.

[4] Zarina Kurbatova et al. « Recommendation of Move Method Refactoring Using Path-Based Representation of Code ». In : *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*. 2020, p. 315-322.

[5] Ilaria Pigazzini. « Automatic detection of architectural bad smells through semantic representation of code ». In : *Proceedings of the 13th European Conference on Software Architecture-Volume 2*. 2019, p. 59-62.