Agenzia nazionale per le nuove tecnologie,
l'energia e lo sviluppo economico sostenibile

Metodi di Machine Learning per il Software Engineering

Progetto Big Code

Autore: Serena D'Onofrio

Referente: Simonetta Pagnutti

Unità: ICT-HPC

7/10/2021

Big Code, progetto di error detection

- **Progetto Big Code**: metodi di Machine Learning e Deep Learning applicati a database di codice sorgente. L'idea è quella di estendere le tecniche di apprendimento automatico create nell'ambito dell'NLP, a dati di tipo codice.
- Collaborazione tra ENEA ed il Dipartimento di Informatica, Scienza ed Ingegneria (DISI) di UniBo per valorizzare la presenza del mirror di **Software Heritage**, il più grande archivio europeo di codice sorgente.

Task
Detection di errori all'interno di uno snippet di codice sorgente utilizzando il Deep Learning.

Progettazione ed implementazione di un modello predittivo capace di verificare correttamente se, all'interno di uno snippet di codice sorgente di un programma, è presente una certa tipologia di errore.

Big Code, progetto di error detection

Motivazione

Automatizzazione del processo di quality checking del codice.

- Trovare errori logici e non solo sintattici, che non vengono identificati dai compilatori tradizionali, istruendo la rete neurale con del codice sorgente corretto.
- Processo automatico che può essere utilizzato all'interno di qualsiasi pipeline di Machine Learning.

Dataset

"Code and Static Analysis Dataset" codici C/C++ sono accoppiati all'output dell'analizzatore statico Infer. Il dataset è composto da 3170 progetti di GitHub in C/C++.
Ogni progetto ha la sua directory, nella sottocartella source si trova il codice originale, in derivatives si trova l'output dell'analizzatore statico, che contiene i file bugs.txt e report.json.
Pulizia del dataset e individuazione dell'errore più comune.

Big Code, progetto di error detection

Metodo

- Rappresentazione del codice
- Fine Tuning

Rappresentazione del codice

Discretizzazione del codice in modo da renderlo leggibile da una rete neurale, mantenendo integre le informazioni sulla sua struttura e sul suo significato.

La rappresentazione, che non è unica, è composta da due parti:

1) Lo snippet di codice viene associato ad una rappresentazione intermedia, che nel nostro caso è il suo albero di sintassi astratta (AST).
2) L'AST viene mappato in un vettore *v*, che è pronto ad essere dato in input a nuovi layer della rete neurale.

Big Code, progetto di error detection

L'**AST** è una tupla $\langle N, T, X, s, \delta, \varphi \rangle$ in cui

N nodi non terminali, T nodi terminali,

X insieme di valori, $s \in N$ radici,
$\delta : N \to (N \cup T)$ mappa N nei figli,
$\varphi : T \to X$.

Il **cammino dell'AST** di lunghezza k è una sequenza

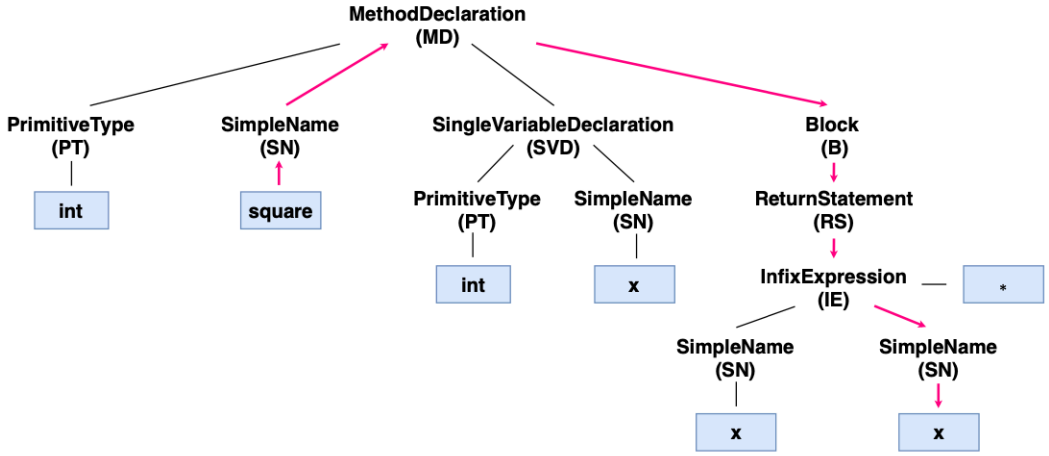
$$n_1 d_1 \ldots n_k d_k n_{k+1}$$

$n_1, n_{k+1} \in T$, $n_i \in N$ per $i = 2, \cdots, k$,
$d_i \in \{ \uparrow, \downarrow \}$ per $i = 1, \cdots, k$ sono le direzioni dei movimenti nell'albero.

Quindi, se $d_i = \uparrow$ allora $n_i \in \delta(n_{i+1})$,
se $d_i = \downarrow$ allora $n_{i+1} \in \delta(n_i)$.

```
int square(int x) {
    return x * x;
}
```

(a) An example code snippet



(b) The snippet's syntax tree. An example of a path is highlighted in pink.

Big Code, progetto di error detection

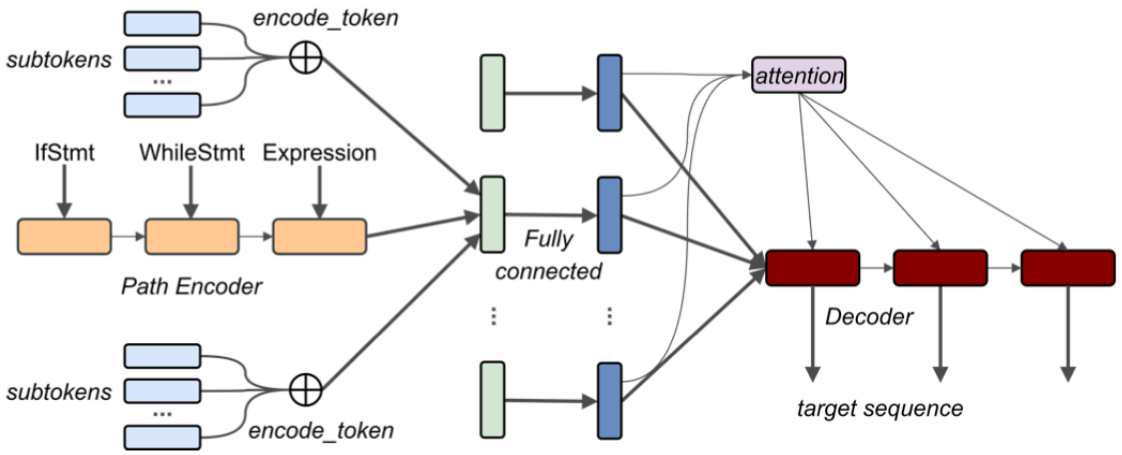
Tool utilizzati

Primo tentativo: ASTminer, rappresenta uno snippet di codice come un vettore di lunghezza fissata.

Secondo tentativo: code2seq, tratta il codice come una sequenza di token.

L'encoder crea una rappresentazione vettoriale per ogni cammino dell'AST separatamente utilizzando le bi-LSTM. Inoltre crea la rappresentazione dei token, ovvero i nodi terminali dell'AST, dividendoli in subtokens.

Il decoder analizza i cammini di AST codificati, per generare l'output.



Big Code, progetto di error detection

Infrastruttura: CRESCO, virtual environment con tutte le librerie utili.

TO DO

- Adattare code2sec al dataset in C/C++ con cppminer.
- **Fine tuning**: addestrare la rete al nostro task. Per farlo, creiamo una Feed Forward Neural Network per la classificazione, da aggiungere alla precedente architettura per addestrare l'intero sistema in modalità End-to-End.
- Generalizzare il problema: classificatore che permette di associare correttamente uno snippet di codice sorgente agli errori presenti all'interno del codice fornito.

Oltre a Big Code

- Crescoware, prime prove su creazione ed utilizzo dei container.
- Dinamiche RPS di interazione tra agenti, modelli di Lotka Volterra e May Leonard.

TO DO

- Sfruttare al meglio Crescoware ed utilizzarlo per tutti i progetti.
- Migliorare l'utilizzo di CRESCO.
- Trasversalità all'interno dei progetti di Machine Learning ed Artificial Intelligence in ENEA.

Grazie!

1101 0110 1100
0101 0010 1101
0001 0110 1110
1101 0010 1101
1111 1010 0000