AI Clean Code Analysis Tool - Predlog projekta

1. Naziv teme

Automatska analiza Java klasa i metoda prema principima Clean Code-a, poput onih iz knjige “Clean Code” autora Robert C. Martina, pomoću hibridnog pristupa (Heuristike + KNN + CNN).

2. Definicija problema

Problem koji se rešava je automatska procena kvaliteta, održivosti i čitljivosti Java koda. Cilj projekta je da se napravi sistem koji analizira Java klase i metode i klasifikuje ih u jednu od tri klase na osnovu čitljivosti i održivosti. Te tri klase su (nezvanično) dobar, moguće izmene i neophodne izmene (zeleno, žuto, crveno). Ovime će biti istaknute klase i posebno njihove metode koje krše Clean Code principe ili mogu biti malo unapređene. Model će koristiti kombinaciju heuristika, klasifikacije pomoću KNN algoritma i dubokog učenja.

3. Motivacija

Razvoj softvera danas zahteva održiv i čitljiv kod kako bi se smanjio tehnički dug, olakšalo timsko programiranje i ubrzalo otkrivanje grešaka. Početnici u svetu softvera često dobijaju komentare na PR-ovima u vezi sa slabo održivim i nečitljivim kodom od strane svojih kolega i nadređenih. Lično sam doživeo iskustvo gde mi na PR-u mentor i kolege ostavljaju komentare za sekcije koda koje nisu dovoljno čiste i treba ih unaprediti kako bi održavanje i čitljivost bili na većem nivou i kako bi uštedeo vreme i budućem sebi i kolegama. Ovi komentari su me puno unapredili i naterali da čitam knjige o čistom kod i tako unapredim sebe. Sada dosta ređe dobijam komentare u vezi sa čistim kodom, a da sam imao alat poput ovog moje unapređivanje bi bilo još brže. Početnicima u svetu softvera, poput mene, ovakav alat može pomoći da nauče principe pisanja čistog koda tako što će brzo uočiti koje sekcije su kritične, dok profesionalnim timovima može dati brzu ocenu kritičnih sekcija koda po pitanju čitljivosti i održivosti.

4. Skup podataka

Inicijalna ideja mi je bila da podatke sakupim na sledeća 2 načina:

1) Video sam da već postoje slični projekti odrađeni i proverio sam neke njihove skupove podataka, ali za sada nisam pronašao idealan koji se poklapa sa onim što sam zamislio. Video sam par sličnih koje bih mogao. iskoristiti i ako ne pronađem idealne, te najsličnije ću malo izmeniti kako bi odgovarali mom projektu i iskoristiti. Evo 2 zanimljiva javna skupa podataka:
https://huggingface.co/datasets/se2p/code-readability-krod
https://huggingface.co/datasets/se2p/code-readability-merged

2) Preuzeo bih javno dostupne Java projekte sa GitHub-a i ocenio ih lično i pomoću komentara na PR-ovima. Za ovo bih pokušao da koristim još par iskusnijih developera i sopstveni kod koji su ocenjivali ili neki drugi. Takođe, iskoristio bih postojeće AI modele, poput Chat GPT, da ubrzam čitav proces, ali bih pregledao njihove ocene sekcija.

5. Način pretprocesiranja podataka

Podaci će se pretprocesirati na sledeći način:

1) Podela podataka na train/val/test skupove.
2) Parsiranje Java koda na klase i metode pomoću biblioteka.
3) Izračunavanje određenih heurističkih metrika iz njih, poput broja metoda za klasu ili broj linija koda za metodu.
4) Parsiranje Java klasa i metoda na kod tokene, koji će biti pretvoreni u sekvence za CNN.

6. Metodologija

Metodologija projekta uključuje hibridni pristup sa tri komponente. Svaka komponenta činila bi deo ocene. Te komponente, koje nisu nužno ravnopravne u finalnoj oceni, su sledeće:
1) Heuristički sloj: određena heuristička pravila (npr dužina funkcije) iz knjige Clean Code.
2) KNN algoritam - Na osnovu određenih osobina metode/klase (broj parametara, dužina, ugnježdavanja…), KNN bi svrstavao kod u neku klasu.
3) CNN: koristi tokenizovane sekvence koda kako bi prepoznao obrasce lošeg stila.

Sve ovo na kraju bi se kombinovalo u finalnu ocenu. Neke od ovih stvari značile bi više od drugih u toj finalnoj oceni.

7. Način evaluacije

Evaluacija će se vršiti poređenjem predikcija sa unapred određenim i labeliranim test skupom podataka, izdvojenim tokom train/test/val. Poredićemo klase koje je dao model sa labelama i videćemo koji procenat je tačan. Posebno analizirati mesta gde se desi ekstremno loše poređenje, tj. stvarna klasa je “dobar” ili “neophodne izmene”, a procena je totalno suprotno, tj. ovo drugo navedeno.

8. Tehnologije

Za implementaciju će se koristiti:
1) Python biblioteke za AI, poput scikit-learn, PyTorch, TensorFlow…
2) Python bilbioteke za parsiranje Jave

9. Relevantna literatura

- Clean Code: A Handbook of Agile Software Craftsmanship.
  Author: Robert C. Martin
- The Pragmatic Programmer: Your Journey to Mastery.
  Author: Andrew Hunt and David Thomas
- Definitivno da će biti još i AI literature, ali trenutno nisam siguran šta sve, tek sam početnik. Većina sličnih alata danas koristi LLM-ove, te nisam uspeo pronaći rad koji bi totalno odgovarao onome što sam ovde opisao, ali ću definitivno dodatno istraživati slične stvari i pokušati realizovati nešto slično. Za sada, mogu navesti jedan rad koji obuhvata deo moje ideje: https://github.com/ehtishamDev/Automated-Code-Quality-Classification-with-Convolutional-Neural-Networks-and-NLP?utm_source=chatgpt.com