Reprezentacja kodu programów dla identyfikacji wadliwych fragmentów kodu bazująca na uczeniu maszynowym

Oprogramowanie stanowi jeden z istotnych czynników stymulujących rozwój nowoczesnego społeczeństwa i przemysłu. Każdego dnia, obserwujemy jak kolejne aspekty naszego życie stają się coraz silniej uzależnione od produktów informatycznych, tym samym czyniąc nas bardziej narażonymi na ewentualne skutki ich błędnego działania. Niestety bolesna prawda jest taka, że tworzone oprogramowanie bardzo często zawiera błędy a ich naprawa jest niezwykle kosztowna. Dla przykładu National Institute of Standards and Technology w Stanach Zjednoczonych oszacował, że błędy w oprogramowaniu kosztują gospodarkę tego kraju 59,5 miliardów dolarów rocznie. Chociaż są to niewątpliwe poważne straty materialne należy pamiętać, że błędy w oprogramowaniu mogą kosztować znacznie więcej – ludzkie życie. Niestety znamy wiele przypadków, w których awaria oprogramowania doprowadziła do śmierci osób. Jak wykazały analizy wiele z tych wypadków można byłoby uniknąć, gdyby zastosowano odpowiednie środki mające na celu zapewnienie jakości oprogramowania. Wiemy także, że opłaca się identyfikować błędy w oprogramowaniu możliwie jak najszybciej, ponieważ ich koszt usunięcia rośnie bardzo dynamicznie z upływem czasu.

Jedną z powszechnie stosowanych technik wykrywania błędów w oprogramowaniu jest prowadzenie przeglądów kodu oprogramowania przez programistów. Stosowanie tej praktyki nie tylko pozwala znajdować defekty, które mogą doprowadzić do błędnego działania oprogramowania, ale także pomagają poprawić jakość samego kodu zmniejszając koszty jego utrzymania. Niestety, przeglądy kodu są praktyką pracochłonną, mocno obciążającą programistów. Zapewne, gdyby udało się odciążyć ich od tego typu zadań mogliby oni poświęcić więcej czasu na implementacje nowych funkcjonalności aplikacji.

Zatem może istniałaby możliwość, aby komputer przejął część z zadań związanych z przeglądem kodu źródłowego oprogramowania? Każdego dnia słyszymy przecież o tym jak sztuczna inteligencja i algorytmy uczenia maszynowego wyręczają człowieka w kolejnych obszarach jego działalności. Czynności, które jeszcze kilka lat temu wykowane były ręcznie dzisiaj obsługiwane są przez maszyny. Podobnie mogłoby być w omawianym przypadku. Wraz z rozwojem ogólnodostępnych narzędzi do zarządzania kodem oprogramowania takich jak GitHub czy Gerrit społeczność naukowa uzyskała dostęp do ogromnego zbioru danych, który może posłużyć do uczenia algorytmów sztucznej inteligencji, tak aby mogły one wspomagać programistów w trakcie przeglądów kodu (np. tylko na platformie GitHub w 2020 roku było udostępnionych 190 milionów repozytoriów kodu).

Niestety komputery nie są w stanie zrozumieć kodu programu w taki sposób jak czynią to ludzie. Musimy zadbać o to, że kod programu zostanie przetransformowany do postaci zrozumiałej dla maszyny, z której może się ona czegoś nauczyć. W tej chwili czynimy to poprzez pozyskanie numerycznych cech opisujących dany fragment kodu. Takie cechy możemy pozyskiwać traktując kod programu jak każdy inny tekst i używając podejść jakie stosowane są do przetwarzania języka naturalnego. Drugim podejściem jest pozyskiwanie informacji charakterystycznych dla danego języka programowania. Niestety każde z tych podejść ma swoje ograniczenia. Ale może udałoby się nam połączyć ich silne strony?

Celem badań zaproponowanych w ramach projektu jest zbadanie możliwości połączenia różnych podejść do reprezentacji kodu źródłowego programów, tak aby wypracować jego uniwersalną reprezentację. Taka reprezentacja mogłaby zostać użyta przez komputery i algorytmy uczenia maszynowego do wspierania przeglądów kodu oprogramowania znajdując defekty w oprogramowaniu.

Aby osiągnąć nasz cel planujemy pozyskanie znacznej ilości danych o projektach informatycznych (projekty typu open source oraz projekty komercyjne), na bazie których wykonamy szereg eksperymentów obliczeniowych badając silne i słabe strony dostępnych podejść do reprezentacji kodu dla algorytmów uczenia maszynowego wykrywając różnych rodzaju defekty i potencjalne problemy w utrzymaniu kodu. Bazując na wynikach naszych analiz, chcielibyśmy zaproponować uniwersalne podejście do reprezentacji kodu, które będzie mogło zostać użyte do prowadzenia analiz kodu programów.

Zamierzamy upublicznić zarówno oprogramowanie powstałe w trakcie projektu jak i zbiory danych, aby umożliwić społeczności akademickiej prowadzenie dalszych prac badawczych w kierunku rozwoju narzędzi analizy kodu bazujących na algorytmach uczenia maszynowego.