Politechnika Warszawska

Wydział Matematyki i Nauk Informacyjnych

Streszczenie

Rozprawa Doktorska

Tytuł: System do oceny podobieństwa kodów źródłowych w językach funkcyjnych oparty na metodach uczenia maszynowego i agregacji danych

Autor: Maciej Bartoszuk

Promotor: dr hab. inż. Marek Gągolewski, prof. PW

Rozprawa poświęcona jest wykrywaniu podobnych kodów źródłowych (zwanych także klonami) w językach funkcyjnych na przykładzie języka R. Detekcja klonów ma dwa główne zastosowania: po pierwsze, służy nauczycielom i trenerom, którzy prowadzą zajęcia z programowania, wskazując prace, których samodzielne wykonanie być może należy zakwestionować. Po drugie, pomaga zapewniać wysoką jakość kodu w projektach informatycznych, poprzez unikanie powtórzeń.

Po pierwsze, w pracy zostaje zaproponowana operacyjna definicja podobieństwa pary kodów źródłowych, która służy jako formalne sformułowanie problemu. Polega ona na wymienieniu możliwych modyfikacji kodu, a następnie stwierdzeniu, że kody A i B są do siebie podobne, jeśli A może powstać przy użyciu dobrze określonych przekształceń B. Co więcej, jako że obecnie spotykane w literaturze sposoby badania skuteczności proponowanych metod nie są zadowalające, zaproponowano nowe podejście, pozwalające na rzetelną ocenę ich jakości. Stworzono zbiory benchmarkowe, poprzez wylosowanie zgromadzonych uprzednio funkcji z popularnych pakietów, a następnie przekształcono losowo wybrane z nich przy użyciu wymienionych wcześniej modyfikacji. Pozwala to na kontrolę zarówno rozmiaru badanej grupy funkcji, jak i jej cech: frakcji klonów czy liczby dokonywanych przekształceń. Na potrzeby pracy ustalono 108 zestawów wartości poszczególnych parametrów sterujących cechami zbioru, a następnie dla każdego z nich, aby wyeliminować wpływ czynnika losowego, wygenerowano po 10 instancji.

Po drugie, proponowane do tej pory algorytmy działają na pewnym z góry określonym sposobie reprezentacji kodu, np. na tokenach. Nie dokonywano porównań, jak dobrze sprawiłby się taki sam algorytm dla innego ciągu, np. surowych znaków. W tej pracy oddzielono algorytmy, jako sposoby porównywania pewnych ciągów lub grafów, od samej reprezentacji, a następnie zbadano skuteczność wszystkich tak uzyskanych kombinacji.

W szczególności zaproponowano nowy sposób reprezentacji, skupiający się na użytych wywołaniach funkcji. Takie podejście pozwala na precyzyjne określenie, w jakim stopniu na skuteczność danej metody składa się sam algorytm, który jest być może już obecny w literaturze, a w jakim – samo przekształcenie kodu do konkretnej postaci. W pracy przetestowano cztery reprezentacje (trzy sekwencyjne i jedną grafową) oraz siedem algorytmów porównujących (cztery dla postaci sekwencyjnych oraz trzy dla grafowych).

Po trzecie, dotychczasowe podejścia do porównywania kodów źródłowych oparte na grafie zależności programu (ang. *Program Dependence Graph*, PDG) nie są efektywne i opierają się na algorytmach, których złożoność obliczeniowa jest wykładnicza. W tej pracy zbadano po raz pierwszy w kontekście detekcji klonów skuteczność algorytmu o złożoności wielomianowej opartego na metodzie Weisfeilera–Lehmana służącego do znajdowania podobnych grafów, a także zaproponowano własną metodę o nazwie SimilaR, która jest lepiej dostosowana pod specyfikę problemu znajdowania podobnych kodów źródłowych. Wyniki eksperymentalne potwierdzają jej lepszą skuteczność. Dodatkowo przedstawiono algorytm tworzenia grafu zależności programu dostosowany do specyfiki języków funkcyjnych, ilustrując jego użycie na przykładzie języka R. Zaproponowana metoda pozwala w łatwiejszy sposób wykrywać modyfikacje stosowane przez osoby dopuszczające się plagiatu.

Ponieważ różne algorytmy porównywania kodów źródłowych koncentrują się na różnych jego cechach, dodatkowo praca skupia się na agregacji wyników kilku takich podejść w kontekście rozwiązywania problemu klasyfikacji binarnej oraz regresji. Znane z literatury metody, takie jak lasy losowe czy maszyna wektorów podpierających, mają jednak zasadniczą wadę: nie pozwalają na interpretację wpływu poszczególnych metod na ostateczny wynik, a także nie gwarantują takich własności, jak monotoniczność ze względu na poszczególne składowe czy idempotentność. Co więcej, nie pozwalają na bezpośrednie porównanie generowanych przez nie wyników. Innymi słowy, np. wartość 0,76 zwrócona przez jedną metodę nie musi reprezentować takiego samego stopnia podobieństwa jak wartość 0,76 uzyskana przy użyciu innego algorytmu. Zaproponowano więc własną metodę agregacji wyników zwracanych przez algorytmy, aby podjąć pojedynczą decyzję o podobieństwie dwóch kodów źródłowych. Jest ona oparta o krzywe i płatki B-sklejane. Podejście to pozwala zachować dobrze interpretowalną postać modelu, poprzez sprowadzenie wyników poszczególnych składowych do jednej skali. Eksperymentalnie wykazano, że lepsze wyniki są osiągane dla zestawu metod niż każdej z nich użytej pojedynczo, a także, że zaproponowana metoda agregacji daje lepsze wyniki niż inne modele uczenia maszynowego, w tym m.in. algorytm lasów losowych.

Warto ponadto zauważyć, że podejścia prezentowane w literaturze posługują się symetryczną postacią podobieństwa: zwracana jest jedna wartość, która opisuje, jak podobne

są obie funkcje do siebie. W pracy tej uznano taki sposób za niewystarczający i zbadano podejście, w którym metoda porównująca dwie funkcje zwraca parę wartości: jak bardzo pierwsza zawiera się w drugiej ($A \subset B$) oraz jak bardzo druga zawiera się w pierwszej ($A \supset B$). Następnie zaproponowano sposoby agregacji tych dwóch wartości przy użyciu t-norm (minimum, produktowej i Łukasiewicza), średniej arytmetycznej lub t-konorm (dualnych do wymienionych t-norm). Dzięki odpowiedniej agregacji wyników kilku metod osiągnięto lepsze wyniki, niż w przypadku podejścia symetrycznego.

Stworzone zostało także oprogramowanie, które pozwala na skorzystanie z dokonań niniejszej pracy w praktyce: publicznie dostępny pakiet w języku R umieszczony w repozytorium CRAN, a także serwis internetowy, dzięki któremu można porównać kody źródłowe.