Primjena optimizacijskih algoritama i strojnog učenja u kontekstu funkcijskog programskog jezika Haskell

Luka Hadžiegrić

Zagreb, Veljača 2018

“Pod punom odgovornošću pisano potvrđujem da je ovo moj autorski rad čiji niti jedan dio nije nastao kopiranjem ili plagiranjem tuđeg sadržaja. Prilikom izrade rada koristio sam tuđe materijale navedene u popisu literature, ali nisam kopirao niti jedan njihov dio, osim citata za koje sam naveo autora i izvor, te ih jasno označio znakovima navodnika. U slučaju da se u bilo kojem trenutku dokaže suprotno, spreman sam snositi sve posljedice uključivo i poništenje javne isprave stečene dijelom i na temelju ovoga rada.”

U Zagrebu, 20.2.2018

Predgovor

Ovim putem želim se zahvaliti svome mentoru dr.sc. Goranu Klepcu koji me je uveo u svijet podatkovne znanosti i pružio mi priliku da radim na ovom projektu, svojoj obitelji na velikoj potpori tijekom studiranja, te svim svojim učiteljima i profesorima koji su me vodili kroz moje dosadašnje obrazovanje, pogotovo dr.sc Janu Šnajderu.

Sažetak

Porastom kompleksnosti programskih rješenja funkcijska paradigma postaje sve značajnija u razvoju softvera jer obećava veću modularnost, stabilnost i razumljivost programskog koda.

Ovaj rad istražuje te prednosti kroz izradu Internet aplikacije i primjenu algoritama strojnog učenja i optimizacije.

Ključne riječi: haskell, optimizacija, strojno učenje, postgresql, internet aplikacija, obrada podataka

Sadržaj

Poglavlje 1

Uvod

Cilj ovog rada je istražiti primjenu čistog funkcijskog programskog jezika Haskell na praktičnom primjeru izrade pametne kuharice koja koristi metode strojnog učenja i optimizacije kako bi korisniku preporučila potencijalno zanimljiv recept ili stvorila tjedni jelovnik koji maksimalno iskorištava sastojke.

Ideja je proizašla iz nezadovoljstva nakon višegodišnjeg korištenja objektno orijentiranih programskih jezika te pojavom elemenata funkcijskog programiranja u popularnim programskim jezicima poput JavaScript-a, C#-a i Jave.

Današnje interakcije ljudi i različitih područja ljudskog djelovanja postaju sve složenije te važnost računarske znanosti i računarstva kao glavnog vezivnog tkiva i kanala za razmjenu i obradu informacija postaje sve veća.

Iz tog razloga pojavljuju se sve složeniji sustavi i primjene programskih rješenja nad kojima se mora brzo iterirati uz maksimalnu sigurnost i kvalitetu. Takva dinamika zahtjeva efektivne alate za upravljanje kompleksnošću i tu se objektno orijentirani pristup pokazuje sve manje poželjnim izborom.

Ovaj trend korištenja funkcijskih programskih jezika prati sve veći broj manjih firmi koje nemaju teret zastarjelog (eng. *legacy*) koda, ali ne zaostaju niti veće firme poput FaceBook-a koji je razvio svoj sustav Sigma[**sigma**] za borbu protiv neželjenih poruka (eng. *spam*) u Haskell-u.

Poglavlje 2

Haskell

Slika 2.1: Haskell logo

Haskell je standardiziran, generalni, lijen i čist funkcijski programski jezik sa ne striktnom semantikom i jakim statičkim tipskim sustavom[**haskell_history**].

Nazvan je po logičaru Haskellu Curryu i nastao je 1990. godine kao standard kojeg je definirao odbor sa idejom da se ujedine tadašnji postojeći funkcijski jezici u zajedničku cjelinu koja bi služila kao baza za daljnja istraživanja.

2.1 Glavne značajke

2.1.1 Lijenost

Lijenost je svojstvo jezika koje znači da se izraz ne evaluira sve dok se ne zatraži njegova vrijednost iz nekog drugog izraza koji se trenutno izvršava. Ovakva odlika nam omogućava da radimo sa beskonačno velikim listama podataka, rekurzijama bez kraja i sl.[**lazy_vs_nonstrict**]

2.1.2 Ne striktna semantika

Ovo svojstvo dobro komplementira lijenost te govori da se izrazi reduciraju izvana prema unutra. Kod strogih jezika redukcija se obično vrši iznutra prema van tako da ukoliko imamo, recimo par vrijednosti (eng. *touple*) koji želimo proslijediti nekoj funkciji i jedna vrijednost je npr. beskonačna lista, kod strogih jezika takav izraz će upasti u beskonačnu petlju jer će pokušati izvršiti beskonačnu listu. Haskell zbog ovog svojstva ima puno veće mogućnosti pri baratanju sa takvim apstraktnim strukturama.[**lazy_vs_nonstrict**]

2.1.3 Čistoća

Većina jezika omogućava da funkcije osim što vraćaju neku vrijednost izvrše i neke akcije poput pisanja na disk ili u bazu. U čistim programskim jezicima takve akcije zovemo sporedni efekti (eng. *side effects*) kako bi se naglasilo da je povratna vrijednost najvažniji dio ishoda funkcije.[**fp_purity**]

Referencijalna transparentnost

Čisti izračuni