NSU: DN3

Uvod

Ukvarjali se bomo z grafom referenc, ki smo ga pripravili iz Agdine standardne knjižnice. Programski jezik Agda se namreč večinoma uporablja za pomoč pri dokazovanju in formalizacijo matematičnih izrekov. V tokratni domači nalogi bi se radi naučili modela, ki bi matematikom predlagal, katere do sedaj dokazane trditve bi se jim splačalo uporabiti v dokazu, ki ga trenutno pišejo, a ga ne znajo dokončati.

Z drugimi besedami ... V danem grafu (glej 1.1) smo nekaj povezav pobrisali. Naša naloga bo naučiti se model, ki bo sprejel par vozlišč (u, v) tega grafa, vrnil pa bo 1, če misli, da bi morala biti v grafu usmerjena povezava $u \to v$, in 0 sicer.

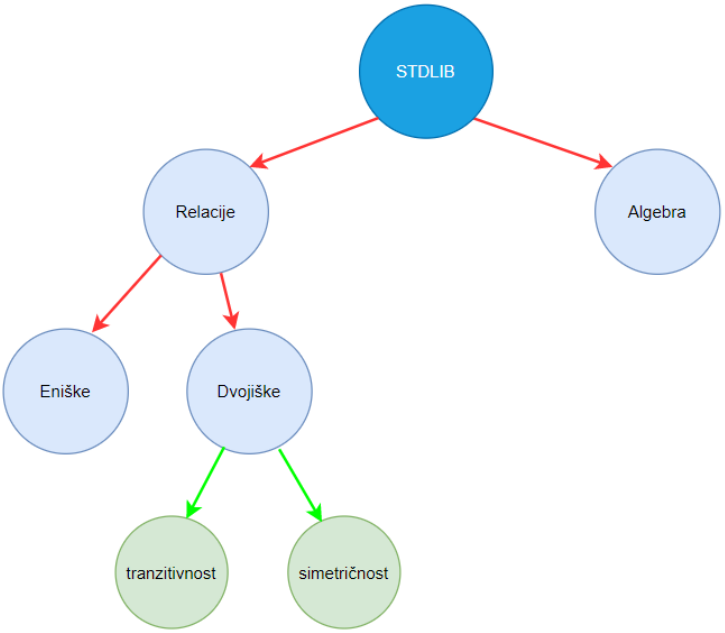
1 Podatki

Podatki so iz treh delov: (usmerjen multi)graf, definicije in testne povezave. V nadaljevanju opišemo vsakega od njih. Pri natančnejšem ogledu lastnosti podatkov si lahko pomagate z datoteko `skripta_za_pomoc.py`.

Originalni graf, ki predstavlja celotno knjižnico `stdlib`, smo nekoliko priredili. Ker napovedujemo povezave, smo nekatere pobrisali. Med vozlišči grafa, ki predstavljajo definicije, smo naključno izbrali petino vozlišč (imenujmo jih testna vozlišča). Nato smo jim pobrisali 75% delež izhodnih povezav tipa `REFERENCE_BODY` (upoštevajoč uteži). Definicijam, ki pripadajo testnim vozliščem (glej 1.2), smo pobrisali 75% vozlišč v telesu.

1.1 Graf

Podatki predstavljajo (usmerjen multi)graf referenc med raznimi definicijami v Agdini standardni knjižnici `stdlib`. Knjižnica ni samo ena ogromna datoteka (kjer je definirano vse od seštevanja naravnih števil do homotopij), ampak je organizirana kot hierarhija modulov, ki so razbiti na podmodule, (ki so razbiti na ...), ki vsebujejo definicije. Del hierarhije modulov v `stdlib` prikazuje spodnji diagram, kjer so (pod)moduli prikazani s svetlomodro, definicije pa z zeleno barvo:



Na njem manjka veliko rdečih povezav (tipa `VSEBUJE`) do preostalih (pod)modulov in zelenih povezav (tipa `DEFINIRA`) do preostalih definicij.

Naš graf torej vsebuje usmerjene povezave različnih tipov (npr. vsebuje, definira, referencira) in vozlišča različnih tipov (npr. knjižnica, modul, definicija). Tu omenimo zgolj še to, da Agda loči več vrst definicij (kar si lahko na zgornjem diagramu predstavljamo kot različne odtenke zelene). Uradno (angleško) ime vrste in njegov pomen podaja spodnja tabela:

oznaka	opis vrste definicije
`:data`	Označuje podatkovni tip, ki ga definiramo (npr. naravna števila).
`:constructor`	Nanaša se na konstruktor(je) za dani podatkovni tip (npr. `naslednik(x)` ali `nič()`).
`:function`	Nanaša se na funkcije in konstante (saj so te funkcije 0 spremenljivk).
`:record`	*Zapis* je ustreznica Pythonovega razreda, saj drži skupaj več različnih vrednosti (`self.ime`, `self.starost`) in lahko vsebuje metode, ki operirajo z zapisi.
`:axiom`	Trditve, ki jih Agda privzame kot resnične in jih lahko uporabi pri dokazovanju.
`:primitive`	Definicije, ki imajo tudi svojo primitivno različico (predvsem zaradi hitrosti izvajanja programa).
`:sort`	Tip, katerega elementi so tipi. Prisoten zaradi tehničnih stvari, a ne pomaga pri razumevanju podatkov.

Za nas je najbolj pomembno, da so matematične trdtive, ki jih dokazujemo z Agdino pomočjo, funkcije (bodisi konstante bodisi "prave"). Ker se kompleksnejše definicije opirajo na bolj preproste, jih lahko predstavimo kot vozlišča v grafu, kot je opisan zgoraj in mu dodamo še usmerjene povezave, ki pripadajo referencam. Če npr. definiramo `plus` na tipu `Nat` (naravnih številih), potem obstajajo v grafu (zeleno) vozlišče `plus`, (zeleno) vozlišče `Nat` in usmerjena povezava `plus`→`Nat`.

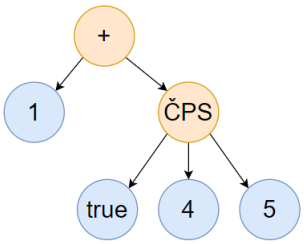
Ker so definicije funkcij dvodelne (v Agdi je treba povedati tip (domeno in kodomeno) ter definirati še samo telo funkcije), ločimo reference v grobem na tiste, ki izvirajo iz tipa funkcije, in tiste, ki izvirajo iz telesa: več o tem v `skripta_za_pomoc.py`.

1.2 Definicije

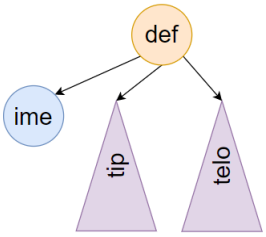
Definicij pri osnovni nalogi ne potrebujete, saj služijo kot vir dodatnih informacij za težji del. S tem ko smo dano definicijo predstavili zgolj z enim samim vozliščem, smo namreč morda kaj koristnega vrgli stran. Defincije predstavimo kot abstraktna sintaktična drevesa, ki smo jih dobili iz Agdine kode. Če bi v Agdi napisali

```
1 + (če true potem 4 sicer 5)
```

je pripadajoče abstraktno drevo



(kjer smo funkcijo če-potem-sicer okrajšali kot ČPS). Drevesa predstavimo s Pythonovim razredom `Definicija` in pomožnin razredom `Vozlisce` (glej `def_drevesa.py`). V resnici so Agdine definicije trodelne, saj ima koren drevesa tri otroke:



vozlišče z imenom definicije, poddrevo, ki opiše tip definicije, in poddrevo, ki opiše telo.

1.3 Testne povezave

To je `.csv` datoteka s štirimi stolpci: `od`, `do`, `tip povezave` in `prisotnost`. Tip povezave je vedno `REFERENCE_BODY`, preostala trojica pa določa imeni definicij, med katerima je (ali pa ni) usmerjena povezava. Na teh povezavah preizkusite svoj model.

2 Naloga

2.1 Osnovna naloga

Uporabite metodo `node2vec`, s katero dobite upodobitev $\varphi(v)$ za vsako vozlišče v grafa iz domače naloge. Ker končni model sprejme par vozlišč, moramo učne podatke za končni klasifikacijski model M (npr. naključni gozd ali logistično regresijo) pripraviti malo drugače kot na vajah, kjer smo napoved za vozlišče v dobili kot $M(\varphi(v))$. Sedaj je vhod v M par vektorjev $(\varphi(u), \varphi(v))$, zato postopamo tako:

1. Iz grafa izluščimo pozitivne in negativne primere (več o tem spodaj). To so pari vozlišč (u_i, v_i), ki pripadajo definicijam, kjer u_i referencira (pozitivni primeri) v_i oz. je ne referencira (negativni primeri).

2. Združimo pare vektorjev $\varphi(u_i)$ in $\varphi(v_i)$ v en sam vektor. S tem dobimo upodobitev povezave $\tilde{\varphi}(u_i \to v_i)$. Lahko ju enostavno staknemo, lahko ju seštejemo, izračunamo povprečje ... Izbira je prepuščena vam.

3. Naredimo učno tabelo iz vektorjev $\tilde{\varphi}(u_i \to v_i)$, ki ji dodamo še stolpec `y`, ki ga napovedujemo in ima v i-ti vrsti vrednost 1, če u_i referencira v_i, in 0 sicer.

4. Naučimo se model M in njegovo točnost preizkusimo na testni množici povezav.

2.1.1 Spolzki deli

Upodobitve. Pri uporabi `node2vec` pazite, saj je dani graf **usmerjen** in ima **različne tipe** povezav in vozlišč. Nekatere povezave so **utežene**, saj ni isto, če se desetkrat skličemo na isto lemo ali pa samo enkrat. Ustrezno

ga predelajte, tako da se boste lahko dobro naučili upodobitev. Konstruktorju za `node2vec` lahko podate parameter `weight_key`. Ne pozabite, da ima `node2vec` veliko parametrov, katerih smiselne vrednosti je morda treba najti s poskušanjem.

Pozitivni in negativni primeri. Poln usmerjen graf z n vozlišči vsebuje $n(n-1)$ povezav. Naš jih ima precej manj, saj se ne more vsaka trditev sklicevati na vse ostale. Zato lahko za pozitivne primere vzamete kar vse povezave tipa `REFERENCE_BODY` (to so tiste, ki nas zanimajo, kot je razvidno iz testne množice). Negativne primere je treba nekako izžrebati (da jih bo približno enako kot pozitivnih). Tukaj je treba premisliti (ali pa empirično ugotoviti), kaj so dobri negativni primeri. Morda bo enakomerno naključno žrebanje nepovezanih vozlišč v redu, morda pa ne.

Končni učni model. Vrečenje dreves (ang. *bagging*) ali naključni gozd sta zelo trpežna pristopa, ki (skorajda) nimata parametrov, ki bi jih bilo treba dobro nastaviti. Če boste uporabljali kaj drugega (npr. podporne vektorje), ne pozabite na parametre.

2.2 Napredna naloga

V učenje vključite še informacije, ki jih izluščite iz dreves definicij. Pri tem si lahko pomagate z

- datoteko z vektorji za večino besed, ki se pojavijo v definicijah - uporabili smo javno dostopne upodobitve angleških besed, ki so bile dobljene z metodo `word2vec`. Za definicijo *besede* si oglej funkcijo `razbij_niz`, ki npr. niz `levaAsociativnost2` razbije na `leva`, `asociativnost` in 2.
- s čimerkoli drugim, kar najdete ali vam pade na pamet (morda je npr. koristen pristop code2vec), ki drevesno strukturo definicije pretvori v vektor.

Predstavitev vozlišča je torej dvodelna: dimenzijam $\varphi(v)$, ki so naračunane iz položaja v v grafu, boste dodali še dimenzije, naračunane iz same definicije, ki ji pripada vozlišče v. Sicer je postopek enak kot pri osnovni nalogi.