Prepoznavanje uporabnika tipkovnice

Leon Modic Matej Miočič Andraž Rozman

7. junij 2020

1 Opis problema

Različni ljudje razvijemo različne stile tipkanja. Naloga je, napišete program, ki bo na podlagi količin, ki jih lahko izmerimo pri tipkanju (kot je časovni zamik med posameznimi pari znakov), prepoznal uporabnika.

2 Opis matematičnega modela

Izbrali smo 47 tipk na tipkovnici in za vsakega uporabnika smo generirali več učnih primerov matrik povprečnih časov velikosti 47×47

$$averageMatrix^{47\times47} = \begin{bmatrix} 0.1301 & 0.0 & \dots & 0.0 & 0.0 \\ 0.0 & 0.0 & \dots & 0.0 & 0.0 \\ \vdots & & & \vdots & \\ 0.0 & 0.122 & \dots & 0.0 & 0.1 \\ 0.0 & 0.0 & \dots & 0.735 & 0.0 \end{bmatrix}$$

Za vsakega izmed uporabnikov znotraj mape `data/` se generira A_i matrika (torej A_{aljaz}, A_{andraz}, A_{leon}, A_{Matej}), ki je sestavljena iz matrix povprečnih časov za vsa merjenja določenega uporabnika.

$$A_i = \begin{bmatrix} p_1 & p_2 & \dots & p_m \end{bmatrix}$$

Kjer so p_i 2209×1 vektorji sestavljeni iz stolpcev matrike povprečnih časov ($averageMatrix$) zloženi eden nad drugega. Nato naredimo SVD razcep:

$$A_i = U_i S_i V_i^T$$

b je 2209×1 vektor sestavljen iz stolpcev matrike povprečnih časov, ki se je generirala za trenutno osebo, kateri želimo določiti ime.

$$U_i S_i y_i = b \rightarrow y_i = (U_i S_i)^+ b$$

$$min(||b - A_i x||) = min(||U_i^T b - S_i y_i||)$$

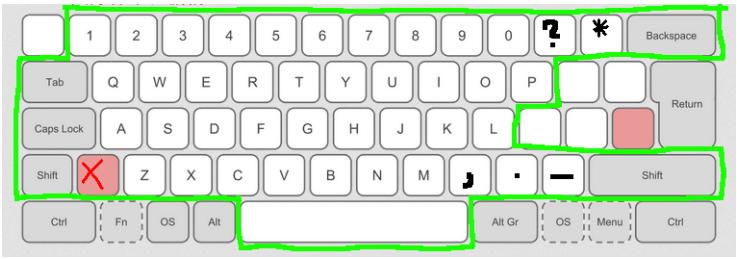
$$names = \begin{bmatrix} "aljaz" \\ "andraz" \\ "leon" \\ "Matej" \end{bmatrix}, norms = \begin{bmatrix} 1.92 \\ 1.2 \\ 1.44 \\ 1.35 \end{bmatrix}$$

3 Opis programske kode

V našem projektu smo naredili dve funkciji za uporabnike: ***main*** in *saveMeasure*. S klicem funkcije *main* uporabnik prične postopek ustvarjanja matrike, ki ga konča s pritiskom tipke *escape*. Nato funkcija na podlagi ustvarjene matrike in drugih predhodno ustvarjenih matrik s *saveMeasure*, prepozna uporabnika ali pa izpiše napako, če to ni mogoče. Ta proces deluje le, če je bilo ustvarjenih dovolj matrik s funkcijo *saveMeasure*.

Funkcija ***saveMeasure*** sprejme tri argumente. Prvi (*user*) je tipa niz in označuje ime uporabnika, drugi (*text*) je tipa niz in označuje naslov besedila, tretji (i) pa je celoštevilskega tipa in predstavlja zaporedno številko matrike za dan naslov. Ob klicu funkcije z veljavnimi argumenti, se znotraj mape *data* ustvari mapa z imenom uporabnika *user* in znotraj te mape še mapa z imenom besedila *text*, če katera od teh dveh map ne obstaja. Nato se kliče funkcija *startMeasure* in prične se proces stvaritve matrike i. Ko *startMeasure* vrne matriko, jo funkcija *saveMeasure* shrani v prej omenjeno mapo.

Funkcija ***startMeasure*** ne sprejme nobenega argumenta. Najprej ustvari tabelo veljavnih tipk, *keys*. To so črke angleške abecede, številke, ... Vse veljavne tipke so vidne na spodnji sliki (slika 1). Ker je vseh veljavnih tipk 47, funkcija inicializira dve 47x47 matriki. Prva (*times*) beleži vsoto časovnih razlik med pritiski dveh tipk, druga (n) pa število teh pritiskov. Glavna while zanka funkcije se izvaja dokler ne pritisnemo tipke *escape*. Na začetku vsake iteracije kličemo funkcijo *KbQueueCheck*, ki vrne informacijo če je bil zaznan pritisk tipke, ter tabelo tipk, ki so bile pritisnjene. Funkcija *KbQueueCheck* je del vmesnika *PsychToolbox*, ki je potreben za izvajanje našega programa. Za delovanje funkcije *KbQueueCheck* pa je sta potrebna klica funkcij *KbQueueCreate* in *KbQueueStart* še pred zanko. Nato s pomočjo funkcije *find* preslikamo kodo tipke v indeks tipke znotraj tabele *keys*. Če je preslikava trenutno pritisnjene tipke ter zadnje tipke veljavna, zabeležimo pritisk kombinacije teh dveh tipk in izračunamo ter prištejemo časovno razliko med tem in zadnjim pritiskom. Ko zaključimo while zanko izračunamo matriko povprečnih časov med dvema tipkama, timesAverage, ki jo funkcija tudi vrne, z naslednjim ukazom (ki tudi prepreči deljenje z 0): timesAverage = times./(n+(n==0))



Slika 1: Veljavne tipke

Struktura je razvidna v funkciji mainCalculate. Na začetku se premaknemo v mapo data, ki vsebuje mape z imeni oseb, ki so sodelovali pri projektu. To pomeni, da so pretipkali določen tekst in shranili matriko s funkcijo saveMeasure.

Nato se z zankami premikamo po mapah. S prvo zanko shranimo ime lastnika testov v spremenljivko userDirTemp z ukazom ls. Spremenljivko še transponiramo, da dobimo imena v vrsticah. Nato odstranimo presledke in imena dodamo v matriko names za kasnejšo uporabo.

Potem generiramo matriko A. Matrika je 2209 x m (kjer m predstavlja število testnih primerov), ki vsebuje meritve s povprečnimi časi med tipkami.

Kako jo naredimo?

Podobno kot prej se z zankami najprej premaknemo v mapo z imeni. Nato se premaknemo v mapo, ki vsebuje teste, glede na besedilo. Še z zadnjo zanko pa naložimo test (matriko 47x47) v spremenljivko X. X nato pretvorimo v vektor in dodamo k matriki A.

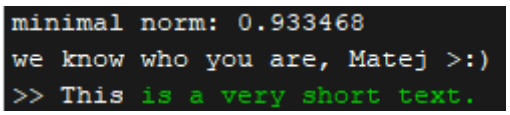
Ko dobimo matirko A za enega izmed tesnih oseb, naredimo SVD razcep in vstavljamo stvari v enačbo.

4 Rezultati in komentarji rezultatov

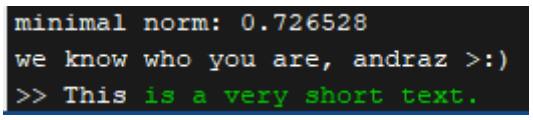
Uspešne rezultate smo dobivali že v začetku izdelave projekta. Najprej smo si izbrali besedilo dolgo približno 1000 znakov. Vsak izmed udeležencev ga je prepisal petkrat. Tako smo imeli shranjenih 15 testnih matrik. Ko smo testirali program na istem besedilu (pretipkali smo isto besedilo), nas je program prepoznal. Probali smo tudi na drugih besedilih, vendar smo dobili malo manj zanesljive odgovore.

Sklenili smo, da bomo dodali več testnih matrik, tokrat z različnimi besedili. Dodali smo še 8 različnih besedil, s povprečno 500 znaki vsak. Tokrat smo prepisali vsako besedilo samo enkrat. Dobili smo zelo pozitivne rezultate, saj nas je program zanesljivo prepoznal po treh stavkih.

Razumljivo pa je, da bo program ne bo vedel, kdo je uporabnik tipkovnice, če prepišemo prekratko besedilo, zato bo vračal različne odgovore, tudi če ga prepiše ista oseba:



Slika 2: Test1 programa z zelo kratkim besedilom.



Slika 3: Test2 programa z zelo kratkim besedilom.

5 Razdelitev dela v skupini

Večinoma smo vso kodo pisali tako, da je tisti, ki je delil zalson dejansko tipkal, skupaj pa smo razmišljali kaj na se napiše.

6 Reference in dejanksa koda

Github