# 1. projekt
# Iskanje po zbirki dokumentov

Domen Gašperlin
Rok Grmek
Jakob Gaberc Artenjak
Anže Gregorc

Matemetično modeliranje, Fakulteta za računalništvo in informatiko

April, 2017

# 1 Opis problema

Namen projekta je izdelati iskalnik relevantnih dokumentov po ključnih besedah z metodo *latentnega semantičnega indeksiranja* (LSI), saj so metode, ki izberejo le dokumente, ki vsebujejo natanko iskane besede, precej nenatačne. Ljudje namreč uporabljamo veliko sopomenk, ki jih preproste metode ne povežejo. Metoda LSI zgradi model, ki združuje več besed v pojme in zato najde tudi dokumente, ki so relevantni, pa ne vsebujejo iskalne besede.

Izdelati je potrebno program, ki bo v dani zbirki za dane ključne besede poiskal najbolj relevantne dokumente.

# 2 Naloge

## 2.1 Iz zbirke dokumentov zgradite matriko A povezav med besedami in dokumenti.

Matrika A:

$$\mathbf{A} = \begin{array}{c} \begin{array}{cccc} \text{doc1} & \text{doc2} & & \text{docD} \end{array} \\ \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1D} \\ a_{11} & a_{12} & & \\ \vdots & & \ddots & \\ a_{B1} & & & a_{BD} \end{bmatrix} \end{array} \begin{array}{l} \text{beseda1} \\ \text{beseda2} \\ \vdots \\ \text{besedaB} \end{array}$$

Vsak dokument ima v matriki svoj stolpec, vsaka beseda pa svojo vrstico. Element $a_{ij}$ pa je frekvenca i-te besede v j-tem dokumentu.

Postopek gradnje:

```
number_of_docs = length(file_names); # shranimo stevilo vseh dokumentov
all_words = []; # inicializacija polja, ki bo vsebovala vse besede
num_of_words_in_docs = zeros(1, number_of_docs); # vektor, ki za vsak
    dokument hrani stevilo vseh besed
for i = 1:number_of_docs # sprehodimo se po vseh dokumentih
  # preberemo i-ti dokument
  doc = textread([path_to_docs, filesep, file_names{i}], '%s');
  # vse besede spremenimo na samo alfa numericne znake in v male crke
  for j = 1:length(doc)
    doc{j} = lower(doc{j}(isalnum(doc{j})));
  end
  # dodamo besede i-tega dokumenta v polje vseh besed
  all_words = [all_words; doc];
  # dodamo stevilo vseh besed v i-tem dokumentu
  num_of_words_in_docs(i) = length(doc);
end
```

Ko imamo zgrajeno polje vseh besed, moramo odstraniti podvojene besede in s tem ustvarimo polje, ki bo služilo kot stolpec v matriki A.

```
[unique_words, ~, numbers] = unique(all_words);
```

In sedaj imamo vse pripravljeno za gradnjo matrike A:

```
  all_possible_numbers = (1:length(unique_words))'; # vektor od 1 do st
      vseh unikatnih besed

# matrika A dimenzije (st. vseh unikatnih besed) x (st. vseh dokumentov)
  A = zeros(length(unique_words), number_of_docs);
  doc_end = 0;

  # sprehodimo se po vseh dokumentih
  for i = 1:number_of_docs
    # hranita stevilo, pri kateri se zacnejo in koncajo besede i-tega
        dokumenta v polju vseh besed
    doc_start = doc_end + 1;
    doc_end = doc_start + num_of_words_in_docs(i) - 1;

    # dodamo frekvence i-tega dokumenta v matriko A
    A(:, i) = histc(numbers(doc_start:doc_end, 1), all_possible_numbers);
  end
```

## 2.2 Matriko A razcepite z odrezanim SVD razcepom $A = U_k S_k V_k^T$, ki obdrži le k največjih singularnih vrednosti.

V Octave okolju lahko odrezan SVD razcep dobimo z ukazom:

```
[U, S, V] = svds(A, k);
```

Odrezan SVD zmanjša t. i. "overfitting" (preveliko prilagoditev modela podatkom, kar povzroči povečan vpliv šuma).

**Razmislite kaj predstavljajo stolpci matrike $U_k$ in matrike $V_k$.** Stolpci matrike $U_k$ predstavljajo "skrite značilnosti" (hidden feature) besed. Ker je matrika ortonormirana, lahko govorimo kot o neki klasifikaciji besed. Enako velja za matriko $V_k$, le da njene vrstice predstavljajo "skrite značilnosti" dokumentov.

**Kakšen k uporabiti?** Odločili smo se, da bomo s trisekcijo poiskusili najti najprimernejši k, ki bo ohranil razcep dovolj natančen in obenem zmanjšal vpliv šuma.

Potek trisekcije: najprej za vsak i od 1 do števila pivotov matrike A izračunamo SVD razcep. V vektor shranimo logaritemske napake (napaka i-tega razcepa glede na matriko A):

```
svd_errors = zeros(rank(a)-1, 1);
for i=1:length(svd_errors)
   [U, S, V] = svds(a, i);
   svd_errors(i) = log(norm(U*S*V' - a, 'inf'));
end
```

Nato s pomočjo trisekcije najdemo dve točki, ki razdelijo podatke (vektor napak) na tri dele. Vsi deli imajo enako število podatkov. Vsako točko posebej pa uporabimo za razdelitev vseh podatkov, po katerih z linearno ($y = x \cdot b + e$) metodo najmanjših kvadratov najdemo dve najprimernejši premici. Postopek trisekcije nadaljujemo tako, da prestavimo skrajno desno oz. levo točko (odvisno, kateri dve premici najbolje opisujeta krivuljo napake).

Implementacija trisekcije:

```
left_limit = 1;
right_limit = length(svd_errors);
while(right_limit - left_limit > 2)
   left = round(left_limit + (right_limit - left_limit) / 3);
   right = round(right_limit - (right_limit - left_limit) / 3);
   [~, ~, r1_left] = ols(svd_errors(1:left), (1:left)');
   [~, ~, r2_left] = ols(svd_errors(left:length(svd_errors)),
       (left:length(svd_errors))');
   [~, ~, r1_right] = ols(svd_errors(1:right), (1:right)');
   [~, ~, r2_right] = ols(svd_errors(right:length(svd_errors)),
       (right:length(svd_errors))');
   if(mean([r1_left; r2_left].^2) < mean([r1_right; r2_right].^2))
      right_limit = right;
   else
      left_limit = left;
   end
end
k = round((right_limit + left_limit) / 2);
```

### 2.3 Iskani niz besed (poizvedbo) zapišite z vektorjem q. Iz poizvedbe q generirajte vektor v prostoru dokumentov s formulo $\hat{q} = q^T \cdot U_k \cdot S_k^{-1}$

Vektor q ima dimenzije enake vektorju, ki vsebuje vse unikatne besede. Zgrajen pa je tako, da je vrstica besede, ki ni v nizu besed (poizvedbi), enaka 0. Vrstice besed, ki pa so vsebovane v poizvedbi pa nosijo vrednost frekvenc določene besede v nizu. Postopek je podoben generiranju frekvenc (stolpcev) za vsak dokument pri nalogi 2.1.

```
q = zeros(length(unique_words), 1);
for i = 2:length(args)
   q = q + ismember(unique_words, lower(args{i}(isalnum(args{i}))));
end
```

Iskanim dokumentom ustrezajo vrstice Vk, ki so dovolj blizu vektorju $\hat{q}$. Kot je v navodilih pisalo, smo za razdaljo uporabili kosinus kota med dvema vektorjema in ne Evklidske razdalje med njima.

```
q2 = q' * U * inv(S);
cos = (V * q2') ./ (sqrt(sum(q2.^2)) * sqrt(sum(V.^2, 2)));
relevant_docs=sortrows([(1:number_of_docs)', cos](cos > min_cos, :), -2);
doc_names = file_names(relevant_docs(:, 1));
```