Deeltoets 1 DAR

28 mei 2013

9:00 - 10:45, BBL-001

- Vermeld op elk vel je naam en studentnummer.
- Toon bij het inleveren je collegekaart.
- Schrijf en formuleer duidelijk.
- Je mag een A4 met aantekeningen raadplegen.
- Het tentamen duurt 1:45 uur

1 top-k algoritmen

We hebben een functie $f = P_1 + P_2$. Voor deze functie willen we een top k van maximale waarden bepalen. De aflopend gesorteerde lijsten voor P_1 en P_2 zijn hieronder gegeven.

OID	P_1
4	100
6	80
1	70
5	60
3	50
2	40

OID	P_2
6	90
1	80
5	70
4	60
2	50
3	50

(i) Beschrijf hoe via het Threshold Algoritme een top 3 berekend wordt.

(ii) Beschrijf hoe via het No Random Access Algoritme een top 3 berekend wordt.

2 Sparse matrix techniques en Google Pagerank

(i) Wat is de algoritmische complexiteit van de standaardalgoritme voor vermenigvuldiging van een matrix M met een vector v in een n-dimensionale ruimte? Ga er van uit dat we voor zowel M als v arraystructuren gebruiken.

(ii) Een matrix is sparse (spaars) als het aantal niet-nul-elementen aanzienlijk kleiner is dan n^2. Beschrijf een alternatieve datastructuur van een matrix om de complexiteit te verbeteren. Schets een algoritme voor Mv.

(iii) Stel vast dat de Google matrix G niet sparse is. Hoe kun je toch sparse matrix technieken gebruiken?

Ter herinnering:

$G = \alpha S + (1 - \alpha)T$, met

$S = H + \frac{1}{n}ea^\top$ en $T = \frac{1}{n}ee^\top$

3 Map-Reduce

Beschrijf hoe je via Map-Reduce woordfrequenties kunt bepalen. Input is een gesplitste tekstfile die woorden bevat zonder interpunctie. Output is een opsomming van woorden met de bijbehorende frequentie, in willekeurige volgorde.
Voorbeeld:
input = to be or not to be
output = (be,2), (or,1), (to,2), (not,1)

4 alignment

Het Needleman-Wunsch-algoritme (NW) is in feite een aanpassing van een standaardalgoritme voor approximate string matching op gewone tekst, in het bijzonder op woorden. In het laatste geval gaat men uit van de Levenshtein distance oftewel edit distance tussen strings. Deze is gebaseerd op drie soorten edit-acties: insert, delete en update. Voorbeelden:

- insert: van `keur` naar `kleur`
- delete: van `niets` naar `iets`
- update: van `flets` naar `fiets`

Het gaat daarbij steeds om één symbool per keer. De kosten van elke edit-actie bedragen 1. Dit correspondeert met een score van -1. De kosten van een match van twee identieke letters zijn 0. De afstand tussen twee strings is het minimale aantal edit-acties dat nodig is om de ene string om te vormen naar de andere. Merk op dat dit een symmetrische maat is.
Voorbeeld: de edit-distance van `GEHEUGEN` en `BEHAGEN` is drie. De G wordt een B, de tweede E een A en de U wordt verwijderd.

(i) Merk op dat er een andere manier is om op afstand drie uit te komen. Welke?

In plaats van de protein similarity (zoals bij NW) gaan we nu de edit distance tussen twee strings berekenen. Dit kunnen we wederom bewerkstelligen via dynamic programming. We hoeven slechts een klein aantal wijzigingen op NW aan te brengen om in plaats van de similarity de edit distance te berekenen.

(ii) Druk de scorefunctie $F(i,j)$ uit in $F(i-1,j)$, $F(i,j-1)$ en $F(i-1,j-1)$. De functie geeft de negatieve edit-distance weer.

(iii) Stel voor `GEHEUGEN` en `BEHAGEN` een matrix op die de berekening van de edit-distance via dynamic programming weergeeft. Duidt de twee oplossingen aan die de minimale afstand representeren.

(iv) Waarom hanteren we deze afstandsmaat niet in het domein van proteïnestrings? Is deze afstandsmaat in de context van genoomstrings toch bruikbaar?