Käsitteellinen klusterointi

Kim Viljanen
Kim.Viljanen@cs.helsinki.fi

Klusterointimenetelmät-seminaari
Helsingin yliopisto, Tietojenkäsittelytieteen laitos
Raportti C–2002–54, s. 86–95, marraskuu 2002

Tiivistelmä

Käsitteellinen klusterointi (conceptual clustering) on klusterointimenetelmä, jossa klusteroinnin lisäksi tuotetaan kullekin klusterille kuvaus siitä, mitkä ovat siihen kuuluvien alkioiden tyypilliset ominaisuudet. Kuvausten ansiosta klustereiden tulkitseminen on helpompaa kuin esimerkiksi klusteriin kuuluvia alkioita tarkastelemalla. Klustereiden kuvaamiseen voidaan käyttää sekä todennäköisyyskuvauksia että konjuktiivisia lausekkeita. Lisäksi tuotetut klusterit voidaan esittää hierarkisesti. Käsitteellisen klusteroinnin sovelluskohteina ovat aineiston analysoinnin lisäksi esimerkiksi mahdollisuudet käyttää opittua luokittelua päättelyyn sekä ontologioiden ja käsitehierarkioiden luomiseen.

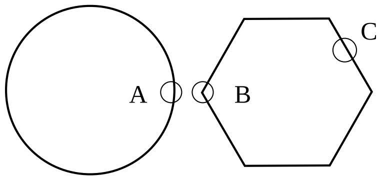
1 Johdanto

Klusterointi (clustering) tarkoittaa tietämyksen muodostamisen menetelmää, jossa syöteaineistona saadut alkiot ryhmitellään samankaltaisuuden perusteella ryppäisiin eli klustereihin (cluster). Alkioidenvälinen samankaltaisuus perustuu usein siihen, että alkioiden ominaisuuksien ajatellaan muodostavan moniulotteisen avaruuden, jossa kahden alkion etäisyys toisistaan lasketaan vertailemalla alkioiden ominaisuusvektoreita toisiinsa jotakin etäisyysmittaria käyttäen. Tuloksena syntyy tieto siitä, mihin klusteriin kukin alkio kuuluu.

Edellä kuvattuun tavanomaiseen klusterointiin liittyy seuraavia ongelmia:

Klustereiden tulkinta Mikäli käyttäjä haluaa tietää tuloksena syntyneiden klustereiden tyypillisiä piirteitä voidaan tämä tehdä tarkastelemalla klustereihin kuuluvia alkioita, kuten esimerkiksi klusterin keskellä olevaa alkiota (centroid) tai useampia alkioita tarkastelemalla [AKJ99], tai tarkastelemalla klustereiden suhdetta toisiinsa hierarkisena puuna [AKJ99, Leh02]. Vaikka klusterit käytetyn klusterointimenetelmän puitteissa olisikin matemaattisesti katsottuna yksiselitteisesti määritelty, voi ihmisen kuitenkin olla vaikea klusterin yksittäisiä alkioita tai hierarkista puuta tarkastelemalla ymmärtää klusterin yleisiä piirteitä ja klustereiden välisiä eroja [RSM83]. Vaikeaselkoisuus johtuu siitä, että alkioilla voi olla hyvin paljon ominaisuuksia, jolloin alkioiden vertailu toisiinsa voi ihmisille olla vaikeaa.

Osa alkioiden ominaisuuksista voivat myös olla merkityksettömiä klusteroinnin kannalta, jolloin nämä ominaisuudet voivat sotkea sekä käytettävää klusterointimenetelmää että tuloksia tulkitsevaa käyttäjää [RSM83]. Aineiston esikäsittelyn [Ukk02] avulla tarpeettomia ominaisuuksia voidaan karsia, mutta siitä huolimatta voi syntyvien klustereiden ymmärtäminen olla vaikeaa, mikäli ominaisuuksia on paljon.



Kuva 1: Intuitiivinen kuva käsitteellisestä klusteroinnista: kuvassa olevien pisteiden A ja B *käsitteellinen* etäisyys toisistaan voi olla suurempi kuin B- ja C-pisteiden etäisyys toisistaan mikäli A ja B kuuluvat eri käsitteisiin. Tässä A kuuluu ympyräkäsitteeseen ja B kuusikulmaisen käsitteeseen. (Sovellettu lähteestä [RSM83].)

Alkioiden konteksti Pelkästään alkioiden ominaisuuksiin perustuva klusterointi ei välttämättä ota huomioon alkioiden kontekstia (lähellä olevia muita alkioita) tai sovellusalueen käsitteitä, joihin alkiot liittyvät [RSM83]. Erityisesti jälkimmäistä puutetta havainnollistaa kuva 1, jossa on intuitiivinen esimerkki tilanteesta, jossa kahden pisteen A ja B etäisyys toisistaan on käsitteellisesti suurempi (pisteet A ja B kuuluvat eri kuvioihin eli eri käsitteisiin) kuin pisteiden Euklidinen etäisyys toisistaan. Käsitteellisestä eroavaisuudesta johtuen olisi mielekästä ryhmitellä pisteet A ja B eri klustereihin, vaikka niiden etäisyys toisistaan jollain toisella mittarilla mitattuna olisi pieni.

Käsitteellinen klusterointi (conceptual clustering) on menetelmä, jolla pyritään ratkaisemaan edellä kuvattuja ongelmia siten, että klusterointia tuotettaessa kullekin klusterille tuotetaan myös mielekäs kuvaus klusterin alkioiden tyypillisistä piirteistä huomioiden samalla klustereiden kuvausten yksinkertaisuus ja selkeys sekä klusteroinnin istuvuus aineistoon [RSM83, LP88]. Verrattuna muihin samankaltaisuusmittareihin, on käsitteellinen samankaltaisuusmittari yleisin mahdollinen samankaltaisuusmittari [AKJ99].

Kuvausten yksinkertaisuus pohjautuu oletukseen siitä, että yksinkertainen selitys on todennäköisemmin oikeassa kuin monimutkainen vastineensa[1], kunhan samalla vältetään myös triviaalit klusteroinnit, kuten yksi klusterointi jokaista alkiota kohden.

Esimerkiksi kuvassa 1 olevat pisteet A ja B voitaisiin Euklidisen etäisyytensä puolesta liittää samaan klusteriin (ehto 1), mutta koska syntyvän klusterin kuvauksesta ("kuusikulmio+ympyrä") voisi tulla tarpeettoman monimutkainen (ehto 2), ei pisteitä A ja B jälkimmäisestä ehdosta johtuen todennäköisesti liitettäisi samaan klusteriin.

Käsitteellisen klusteroinnin tuloksina syntyviä klustereita voidaan luokkina, jossa klusterin (luokan) alkiot ovat kyseisen luokan instansseja ja luokilla on ominaisuuksia (klustereiden kuvaukset), jotka kuvaavat luokkaan kuuluvia alkioita. Mikäli tuotetut klusterit ryhmitellään hierarkisesti, voidaan tällaista hierarkiaa ajatella luokkien periytyvyytynä (a on b:n aliluokka) tai sisältyvyyssuhteena (a sisältyy b:hen). Luokkanäkökulmasta katsottuna käsitteellisen klusteroinnin tavoitteena on tuottaa syöteaineiston perusteella luokittelu, joka kuvaa annetun aineiston siten, että luokittelua voidaan käyttää myös tulevan aineiston luokitteluun. Esimerkki tästä näkökulmasta on COBWEB-järjestelmä [Fis87].

Käsitteellisen klusteroinnin tärkeimpiä sovelluskohteita ovat aineiston analysointi (kuten muillakin klusterointimenetelmillä) sekä luokittelun oppiminen, jota voidaan hyödyntää esimerkiksi eksplisiittisten käsitteistöjen, kuten taksonomioiden (käsitehierarkia) ja ontologioiden (käsitehierarkia, jossa käsitteillä on ominaisuuksia ja suhteita toistensa välillä) automaattiseen tuottamiseen [GWM95, CCH01]. Taksonomioita ja ontologioita tarvitaan

[1]Sama idea on myös Occamin partaveitsi -periaatteessa, jonka mukaan yksinkertainen selitys on aina paras.

tietämyskantojen (knowledge base) ja Semantic Web[2] -sovellusten toteuttamiseen, jossa ajatuksena on se, että tietoa esitetään koneen ymmärtämässä muodossa.

Vakiintunut tapa tuottaa taksonomioita ja ontologioita on käyttää siihen asiantuntijoita, joka on sekä hidasta että kallista. Hitaudesta ja kalliudesta johtuen esimerkiksi semantic web -pohjaiset palvelut eivät toistaiseksi ole yleistyneet. Käsitteellinen klusterointi voisi olla yksi keino nopeuttaa ns. älykkäiden sovellusten yleistymistä.

Muutamia tunnettuja käsitteellisen klusteroinnin algoritmeja ovat esimerkiksi CLUSTER/2, COBWEB, AutoClass II, CLUSTER/S, RESEARCHER, CLASSIT ja LABYRINTH. Tässä paperissa esitellään tarkemmin CLUSTER/2-, COBWEB- sekä AutoClass II -järjestelmien toimintaperiaatteet.

Seuraavassa luvussa määritellään käsitteellinen klusterointi hieman määrämuotoisemmin. Tätä määritystä hyödyntäen esitellään luvuissa 3, 4 ja 5 menetelmät CLUSTER/2, COBWEB ja AutoClass II, kukin omassa luvussaan. Lopuksi esitetään yhteenveto esitetyistä menetelmistä ja käsitteellisestä klusteroinnista.

2 Käsitteellinen klusterointi

Yleinen käsitteellinen klusterointiongelma voidaan esittää kuusikkona $(\mathcal{X}, \mathcal{L}, \mathcal{I}, \mathcal{T}, \mathcal{D}, \mathcal{G})$ ja sen yksi ilmentymä (toteutus) voidaan esittää seitsikkona $(X_n, L_n, I_n, T_n, D_n, G_n, S)$, jossa $X_n \in \mathcal{X}$, $L_n \in \mathcal{L}$, $I_n \in \mathcal{I}$, $T_n \in \mathcal{T}$, $D_n \in \mathcal{D}$, $G_n \in \mathcal{G}$ ja joukko S sisältää klusteroitavat alkiot (alkioiden ominaisuudet), jotka ovat joukon X_n äärellinen alijoukko [LP88]. Yksittäinen alkio kuvataan merkinnällä e (event) ja alkiot muodostavat yhdessä joukon S eli syötteen. Esimerkiksi eri käsitteellisen klusteroinnin algoritmit muodostavat kukin yhden ilmentymän yleisestä käsitteellisestä klusterointiongelmasta. Seuraavaksi käydään läpi edellä kuvattujen symbolien merkitys ilmentymän osalta. (Tarkka kuvaus yleisestä tapauksesta lähteestä [LP88].)

Alkioiden ominaisuuksien joukko X

Klusteroitavien joukkojen ominaisuudet voidaan kuvata ominaisuuksien $x_1, x_2, ..., x_n \in X_n$ avulla, jossa nämä ominaisuudet muodostavat joukon X_n (n=ominaisuuksien lukumäärä).

1. esimerkki: Ominaisuuksien kuvaukseen voidaan valita käytettäväksi totuusarvoja, jolloin yhden alkion e_i ominaisuudet voivat olla (ominaisuuksien lukumäärä n=4): $e_i = x_1\bar{x}_2x_3\bar{x}_4$, jossa merkintä $\bar{x}_i$ tarkoittaa negaatiota. Mikäli kaikilla ominaisuuksilla $x_i = 1, i \in 1...4$, on $e_i = \langle 1, 0, 1, 0 \rangle$.

Alkion ominaisuudet voivat saada edellä kuvattujen totuusarvojen lisäksi arvokseen myös esimerkiksi reaalilukuja ja diskreettejä arvoja (esimerkiksi auton ovien lukumäärä) [PC88] sekä rakenteisia arvoja [RSM83].

Klustereiden kuvauskieli L

Käsiteellisessä klusteroinnissa klusterit kuvataan jollakin kielellä L_n, jossa jokainen klusteri $c \in L_n$. Tämä poikkeaa tavanomaisesta klusteroinnista, jossa klusterit kuvataan klusteriin kuuluvien alkioiden toimesta.

Klusterin kuvaukseen käytetty kieli L voi perustua moneen eri lähestymistapaan. Yksinkertaisin ja useimmille intuitiivisesti ymmärrettävä tapa kuvata klustereita on loogisten

[2]http://www.semanticweb.org/

konnektiivien (conjunctive logical expression) käyttö [RSM83]. Esimerkki loogisia konnektiiveja käyttävästä klusterin kuvauksesta on $[X_1 > 3][X_2 =$'kissa' $\vee$ 'koira'], joka tarkoittaa sitä, että X_1 on suurempi kuin 3 ja X_2 on pienempi kuin 2. Vastaavasti edellä kuvattu totuusarvoinen kuvaus on myös looginen konnektiivikuvaus, jossa klusterin kuvaus voi esimerkiksi olla $[X_1 = 1]$, $[X_2 = 0]$ ja $[X_3 = 3..6]$, jossa jälkimmäisellä tarkoitetaan arvoaluetta välillä 3..6. Yksinkertaisten loogisten konnektiivien käytön ongelma on se, että niiden avulla voidaan kuvata vain suorakaiteen muotoisia tai isotrooppisia klustereita [AKJ99], joka käytännössä tarkoittaa sitä, ettei kyseistä kuvaustapaa käyttäen voida esimerkiksi kuvata tilannetta, jossa klustereiden väliset rajat ovat osittain epäselvät.

Yksinkertaisia loogisia konnektiiveja edistyksellisempi tapa klustereiden kuvaamiseen on todennäköisyyskuvausten käyttö joko siten, että klusterin ominaisuudet esitetään todennäköisyyksinä [Fis87] tai siten, että alkion kuuluminen tiettyyn klusteriin esitetään todennäköisyyksinä [PC88]. Näitä kuvaustapoja käsitellään tarkemmin luvuissa 4 ja 5.

2. esimerkki: Klustereiden kuvauskieleksi L_n voidaan valita sama kieli, jolla alkiot kuvataan (ns single representation trick). Näin ollen, esimerkkiä 1. jatkaen, klusteri c_j voidaan kuvata totuusarvoisena vektorina m, jonka arvona voi esimerkiksi olla $m = x_1\bar{x}_2x_3\bar{x}_4 = \langle 1, 0, 1, 0\rangle$. Huomio, että klusterin kuvaus voi olla myös lyhyempi kuin arvo n, eli esimerkiksi klusterin c_k kuvaus voisi olla esimerkiksi $m = \bar{x}_1\bar{x}_3 = \langle 0, ?, 0, ?\rangle$.

Tulkinta $I_n : L_n \to 2^{X_n}$

Tulkinnalla I tarkoitetaan tulkintaa siitä, mitkä alkiot kuuluvat tiettyyn klusteriin. Alkion e_i kuuluminen klusteriin c kuvataan merkinnällä $e \in I_n(c)$, jossa I_n on käytettävä tulkinta.

Yksinkertainen tulkinta on seuraava: Jos kieli L_n perustuu samoihin ominaisuuksiin X_n joilla alkiotkin kuvataan, kuuluu klusteriin c kaikki ne alkiot joiden ominaisuudet täsmäävät klusterin määritelmään. Eli esimerkiksi, jos klusterin c kuvaus on vektori $m = x_1\bar{x}_2x_3\bar{x}_4 = \langle 1, 0, 1, 0\rangle$, kuuluuvat klusteriin kaikki alkiot, joilla on ominaisuutena $e = x_1\bar{x}_2x_3\bar{x}_4 = \langle 1, 0, 1, 0\rangle$ eli kaikki $e \in\subseteq I_n(c)$. Tämän lisäksi alkiolla e voi olla myös muita ominaisuuksia, jotka eivät vaikuta klusteriin kuulumiseen (esimerkiksi jokin ominaisuus x_5).

Tiiviysmitta T, etäisyysmitta D, klusteroinnin hyvyysmitta G

Tiiviysmitalla mitataan klusteroinnin yleistä tiiviyttä (ytimekkyys) eli sitä, missä määrin klusteroinnin klusterit kuvaavat samankaltaisia alkioita (similarity). Etäisyysmittarilla puolestaan mitataan kuinka paljon eroavaisuutta klusteroinnin sisällä on vertailemalla klustereiden kuvauksia (dissimilarity).

Klusteroinnin hyvyysmitta G perustuu sekä tiiviysmittaan T että etäisyysmittaan D siten, että $G_n(C) = G_n(T_n(C), D_n(C))$, jossa C on arvioitava klusterointi. Hyvyysmitan G avulla pyritään löytämään aineistoon S parhaiten sopiva klusterointi C, joka on myös samalla klustereiden kuvausten osalta yksinkertainen.

3. esimerkki: Mikäli klustereiden kuvaukseen käytetään esimerkissä 1. ja 2. käytettyjä totuusvektoreita, voidaan tiiviysmitaksi määritellä klusterointiin $c \in C$ kuuluvan tiivimmän kuvauksen pituus min $t_n(c)$. Eli esimerkiksi, jos klusterin c_i kuvaus on $m_i = x_1\bar{x}_2x_3\bar{x}_4$ ja klusterin c_j kuvaus on $m_j = \bar{x}_1\bar{x}_3$, on kyseisen klusterin tiiviys $\min t_n(m_i), t_n(m_j) = 4, 2 = 2$. Vastaavasti etäisyysmittarin määritelmäksi voidaan sopia, että sillä tarkoitetaan pienintä eroavaisuutta kahden klusterin c_i ja c_j välillä, eli min $d_n(c)$. Etäisyys d edellä kuvattujen m_i:n ja m_j:n välillä on 2. Mikäli sovitaan, että G_n:n arvo on $G_n = \min T_n(C), D_n(C)$ on G:n arvo edellä kuvatussa tilanteessa $G_n = 2$.

Edellä kuvatun klusterin hyvyyden mittaustavan on tarkoitus olla laajennettavissa mihin tahansa käsitteelliseen klusterointimenetelmään sopivaksi [LP88]. Tästä johtuen se on ilmaistu varsin yleisellä tasolla ja sen oikeellisuuden arviointi vaikeaa: onko varmasti niin, ettei klusteroinnissa tarvitse huomioida alkioiden sopivuutta klustereihin?

Klusteroinnin tulos

Klusteroinnin tavoitteena on löytää jokin klusterointi C, joka kattaa syötteenä saatujen alkioiden joukon S ja jossa klusterointi C on hyvyysmitalla G mitattuna paras mahdollinen klusterointi verrattuna muihin klusterointivaihtoehtoihin.

3 CLUSTER/2

CLUSTER/2 [RSM83] on Michalskin ja Steppin vuonna 1983 esittelemä ensimmäinen käsitteellinen klusterointijärjestelmä, joka tuottaa syötteenä saadulle aineistolle konjunktiivisen hierarkisen käsitteellisen klusteroinnin.

Järjestelmä saa syötteenään joukon alkioita E, klustereiden lukumäärän k sekä klusteroinnin hyvyyden määritelmä LEF:in (Lexicographical Evaluation Functional with tolerances) avulla, jossa käyttäjä voi valita seuraavista eri tavoista aineistoon parhaiten sopivat mittarit: klusteroinnin ja alkioiden yhteensopivuus (fit), klusterikuvausten yksinkertaisuus (simplicity), klustereiden välinen eroavaisuus (inter-cluster difference), poikkevien arvojen lukumäärä[3] (discrimination index) sekä ulottuvuuden vähennys[4] (dimensionality reduction).

Klusteroinnin tuloksena syntyy pistevieras klusterointi eli kukin alkio kuuluu vain yhteen klusteriin.

Algoritmi koostuu seuraavista vaiheista:

1. Aloituspisteiden valinta (lukumäärä k), jotka valitaan syötteenä saaduista alkioista. Pisteet voidaan valita joko satunnaisesti tai jollain muulla tavalla.

2. Jokaiselle edellä valitulle pisteelle luodaan mahdollisimman yleinen käsite eli ns tähti. Luotava kuvaus ei saa peittää muita pisteitä. Luotava käsite kuvataan luvussa 2 esitetyillä konjuktiivisilla loogisilla lauseilla.

3. Klusteroinnin optimointi siten, että edellisessä kohdassa luotuja mahdollisesti päällekkäin meneviä käsitteitä muutetaan siten, etteivät ne jatkossa mene päällekkäin.

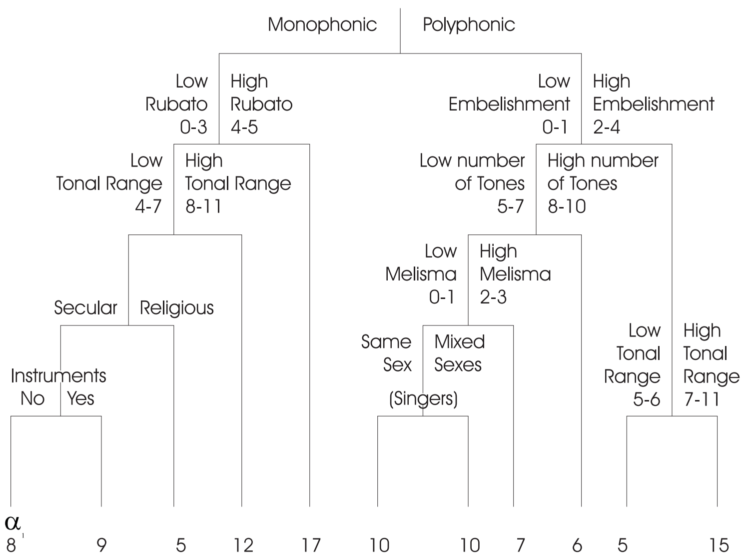
4. Päättymiskriteerin tarkistaminen. Mikäli tämä oli ensimmäinen kierros, tallennetaan luotu klusterointi. Muussa tapauksessa verrataan nyt luotua klusterointia aiemman kierroksen klusterointiin LEF-kriteerin avulla. Algoritmi päättyy, mikäli tiettynä määränä aiempia kierroksia ei ole syntynyt aiempaa parempaa klusterointia.

5. Uusien alkupisteiden valinta syötteenä saaduista alkioista. Pisteet valitaan siten, että ne sisältyvät luotuihin käsitteisiin. Vaihtoehtoisia tapoja valita pisteet ovat: 1) keskipisteen valinta (mahdollisimman keskellä käsitettä) ja 2) reunalla olevan pisteen valinta.

   Siirrytään vaiheeseen 2.

Klustereiden hierarkia tuotetaan varsinaisen klusteroinnin päätyttyä.

[3] Ominaisuus, jonka arvo on eri jokaisen klusterin kuvauksessa.
[4] Ominaisuuksien minimimäärä, joka tarvitaan erottelemaan klusterit toisistaan.



Kuva 2: CLUSTER/2-järjestelmän tuottama tulos kun syötteenä oli espanjalaisia kansanlaulujen ominaisuuksia kuvanneita alkioita [RSM83].

Edellä kuvattu algoritmi on käytännössä varsin tehoton, koska tähtien luominen on raskasta mikäli alkioiden kuvaukseen käytettyjen ominaisuuksien $|X|$ lukumäärä on suuri ja/tai jos luotavien klustereiden lukumäärä k on suuri. Tästä johtuen [RSM83] esittää useita parannusehdotuksia algoritmiin, joita tässä yhteydessä ei ole tilasta johtuen mahdollista käsitellä.

Esimerkkinä CLUSTER/2:sen toiminnasta on 100 espanjalaista kansanlaulua kuvanneiden tietoalkioiden joukko, jossa kutakin laulua on kuvattu 22 musiikkiaiheisella ominaisuudella [RSM83]. Esimerkkejä ominaisuuksista ja niiden arvoalueista ovat "Tonal Range=1..11", "Harmonic Structure=Monophonic, Polyphonic" sekä "A Serenade=yes, no".

Klusterointia tehtiin kukin taso kerrallaan siten, että klustereiden määräksi valittiin jokaisella kerralla $k = 2$. Esimerkki klusterista on α_1 = [harmonic structure=monophonic] [rubato=low] [tonal range=low] [type=secular] [instruments used=no] (nämä ovat klusteriin johtavilla poluilla hierarkiassa) sekä ominaisuudet [no. of distinct tones=5..8] [dance=no] [panegyric=no] [no. of phrases=1..2] [melisma=0..2] [tension=1..3]. Kuvassa 2 on graafinen esitys muodostuneista klustereista. Toisenlaisia tuloksia syntyisi, mikäli valittuja LEF-kriteerejä muutettaisiin. Tuotettu klusterointi vaikuttaa selkeältä ja helposti ymmärrettävältä.

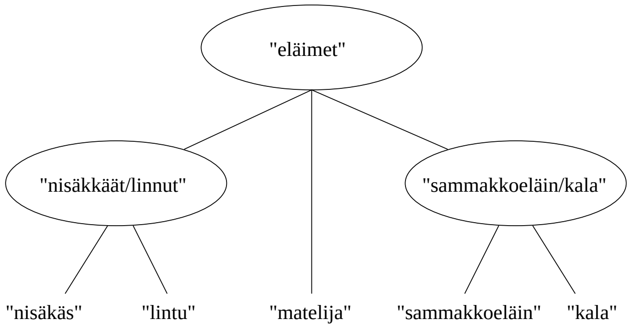
4 COBWEB-järjestelmä

COBWEB-järjestelmän [Fis87] tavoitteena on tuottaa syöteaineiston perusteella hierarkinen luokittelumalli käyttäen käsitteellistä klusterointia. Järjestelmä saa syötteenään tietoalkioita, joihin on liitettynä ominaisuuksia avain-arvo-parien muodossa, jossa arvot voivat olla ainoastaan nominaaleja. Esimerkki mahdollisesta järjestelmälle annetusta syötteestä esitetään taulukossa 1, jossa kukin rivi edustaa yhtä alkiota. Syötetietojen perusteella COBWEB tuottaa aineistoon sopivan luokitteluhierarkian sekä kullekin luokalle *todennäköisen kuvauksen* (probabilistic concept) luokasta. Lopputulos esimerkkiaineiston perusteella luodusta luokittelusta esitetään kuvassa 3.

Järjestelmä toimii inkrementaalisesti siten, että luokittelumallia voi aina täydentää uusilla alkioilla, jolloin jo tuotettua mallia muutetaan tarpeellisilta osin. Inkrementaalisuus erottaa COBWEB:in useimmista muista käsitteellisen klusteroinnin toteutuksista, joissa toteu-

alkion nimi	päällyste	sydänkammiot	ruumiinlämpö	hedelmöitys
'nisäkäs'	karva	neljä	tasalämpöinen	sisäinen
'lintu'	höyhen	neljä	tasalämpöinen	sisäinen
'matelija'	suomu	epätäydellinen-neljä	vaihtolämpöinen	sisäinen
'sammakkoeläin'	kostea iho	kolme	vaihtolämpöinen	ulkoinen
'kala'	suomut	kaksi	vaihtolämpöinen	ulkoinen

Taulukko 1: Esimerkki syötteenä käytettävästä alkiodatasta. (Suomennettu alkuperäisestä lähteestä [Fis87].)



Kuva 3: COBWEB-järjestelmän tuottama puuesitys eläinaiheisessa esimerkissä [Fis87].

tus on usein ollut epäinkrementaalinen. Epäinkrementaalisuus tarkoittaa sitä, että aina kun alkioiden määrä muuttuu, on luokittelumalli rakennettava alusta alkaen uudestaan, joka on sekä aikaa että resursseja vaativaa eikä sovellu hyvin tilanteisiin, joissa klusteroitavaa aineistoa tulee jatkuvasti lisää.

COBWEB sisältää hierarkien päivittämiseen toiminnot, joilla luokkia (klustereita) voi lisätä, poistaa, yhdistää ja jakaa. Nämä toiminnot ovat tärkeitä, jotta järjestelmä pystyy peruuttamaan aiempia huonoja oppimistuloksia esimerkiksi jakamalla liian monimutkaisiksi paisuneita luokkia tai yhdistämällä samankaltaisia luokkia. Ilman käänteistoimintoja hierarkiasta tulisi vääristynyt ja se heijastelisi hyvin vahvasti alkioiden sisäänlukujärjestystä. Muokkaustoimintojen ansiosta COBWEB-järjestelmälle ei tarvitse kertoa luokkien määrää etukäteen, vaan niiden määrä muuttuu dynaamisesti syötteen mukaisesti.

Järjestelmä ylläpitää kuhunkin luokkaan liittyen tietoja siitä, kuinka todennäköisiä syötteenä saatujen alkioiden ominaisuudet ovat luokkaan kuuluvilla alkioilla. Esimerkiksi "Kala"-luokassa on todennäköistä, että alkioilla on ominaisuus "päällyste=suomut", kun taas epätodennäköistä olisi, että "Kala"-luokan alkioilla olisi ominaisuus "päällyste=höyhen". COBWEB:ssä oletuksena on ominaisuuksien riippumattomuus toisistaan (mikä ei välttämättä ole realistinen oletus, kuten myös Fisher itse toteaa[Fis87]).

COBWEB:ssa luokitteluhierarkian luominen perustuu siihen, että jokaiselle alkiolle pyritään löytämään paras mahdollinen sijoittelu olemassa olevassa hierarkiassa. Alkion lisäys tapahtuu lähtemällä juuriluokasta liikkeelle, seuraten aina parasta vaihtoehtoa kunnes alkiolle löytyy sopiva lehtiluokka, johon alkio lisätään. Vaihtoehtoisia toimintatapoja kullakin tasolla ovat: 1) mikäli luokka on lehtiluokka, voidaan alkio lisätään luokan jäseneksi 2) alkiolle voidaan luoda uusi luokka, johon se lisätään ensimmäiseksi alkioksi, 3) olemassa olevia luokkia voidaan yhdistää (merge), 4) olemassa olevia luokkia voidaan jakaa (split) tai 5) valitaan paras luokka, jonka aliluokkiin sijoitusta seuraavaksi yritetään. Kohdassa 3 suoritus jatkuu yhdistämisen jälkeen uudessa, yhdistetyssä luokassa. Kohdan 4 jälkeen siirrytään koko hierarkian juureen, josta alkion sijoitus aloitetaan uudestaan.

Kullakin tasolla menetelmän valinta tapahtuu heurestisesti kokeilemalla kaikkia vaihtoehtoja erikseen (ilman että muutokset jäisivät voimaan) ja laskemalla kunkin vaihtoehdon kohteena olevan luokan saama ns. category utility -arvo (CU). Poikkeuksena on kohta 3, jossa määräävänä tekijänä on tason parhaimpien kahden luokan saama CU-arvo. CU-arvo määräytyy sen mukaan mikä on luokkaan kuuluvien alkioiden samankaltaisuus muiden luokkien alkioiden kanssa (inter-class similarity) ja vastaavasti kuinka paljon mitattavan luokan alkiot poikkeavat muista alkioista (intra-class similarity).

Muodollisemmin esitettynä CU-arvo lasketaan seuraavan kaavan 1 mukaisesti:

$$CU = \frac{\sum_{k=1}^{n} P(C_k)[\sum_i \sum_j P(A_i = V_{ij}|C_k)^2 - \sum_i \sum_j P(A_i = V_{ij})^2]}{n} \tag{1}$$

Arvo C_k viittaa johonkin luokkaan käsittelyssä olevalla tasolla luokitteluhierarkiassa. Käsittelyssä olevalla tasolla olevien lukumäärä ilmaistaan arvolla n ja tasolla olevat luokat muodostavat joukon $C_1, C_2, ..., C_n$. A_i on jokin ominaisuus (attribute), joka luokkaan kuuluvilla alkioilla voi olla (esimerkiksi "päällyste") ja V_{ij} ominaisuuden A_i arvo (esimerkiksi "suomut").

Todennäköisyys $P(C_k)$ kertoo, kuinka todennäköisesti luokitukseen kuuluva alkio kuuluu luokkaan C_k. Todennäköisyys $P(A_i = V_{ij}|C_k)$ vuorostaan viittaa siihen, kuinka todennäköisesti luokkaan C_k kuuluvilla alkioilla on ominaisuus $A_i = V_{ij}$ (intra-class similarity, ennakoitavuus). Todennäköisyys $P(A_i = V_{ij})$ kertoo kuinka todennäköinen ominaisuus $A_i = V_{ij}$ on kaikille alkioille (inter-class similarity, ennustavuus).

COBWEB:in avulla voidaan tuottaa luokittelumalli, jossa luokat määritellään todennäköisinä käsitteinä (probabilistic concept). Esimerkkinä käsitteen "Eläimet" todennäköisyyskuvaus:

ominaisuudet	arvot ja todennäköisyydet
päällyste	karva 0,20; höyhen 0,20; suomu 0,20; kostea iho 0,20; suomut 0,20
sydänkammiot	neljä 0,40; epätäydellinen-neljä 0,20; kolme 0,20; kaksi 0,20
ruumiinlämpö	tasalämpöinen 0,40; vaihtolämpöinen 0,60
hedelmöitys	sisäinen 0,60; ulkoinen 0,40

Kuten esimerkistä näkee, voi yksittäisen luokan tulkitseminen olla varsin hankalaa silloin, kun luokka kuvaa hyvin laajaa joukkoa alkioita (esimerkiksi koko eläinkuntaa). Mikäli tämäntyyppinen todennäköisyyskuvaus halutaan muuttaa totuusarvoiseksi kuvaukseksi, on sovittava jokin raja-arvo, jonka ylittävä todennäköisyys pidetään loogisena totena ja muussa tapauksessa epätotena. Yksinkertaisimillaan voidaan asettaa jokin kiinteä raja-arvo, kuten esimerkiksi 0.67, jota Fisher ehdottaa käytettäväksi [Fis87]. Toinen vaihtoehto on se, että käytettävä raja-arvo riippuisi siitä, mikä on tietyn luokan tarkasteltavana olevan ominaisuuden ennakoitavuus (predictability) ja ennustavuus (predictive), jolloin ominaisuuksille saadaan painotuksia.

COBWEB:in suorituksena lopputuloksena syntyy hierarkinen puu, johon syöteaineiston alkioiden perusteella luodut luokat ja niihin liitetyt alkiot on sijoitettu. Esimerkki tällaisesta puusta näkyy kuvassa 3.

COBWEB-järjestelmä vaikuttaa toimivalta käsitteelliseltä klusterointimenetelmältä ainakin sillä perustein, mitä Fisher itse kirjoittaa [Fis87] sekä mitä esimerkiksi [JH01] asiasta toteaa. COBWEB:in suurimpia puutteita on rajoittunut syöteaineisto, joksi kelpaa ainoastaan nominaaliset avain-arvo-parit, johon ratkaisuksi on kehitetty useita erilaisia versioita COBWEB-järjestelmästä. COBWEB-menetelmän on myös todettu olevan riippuvainen syötteen järjestyksestä (eri järjestys tuottaa erilaisen klusteroinnin) [YPY96].

5 AutoClass II

AutoClass II-järjestelmä [PC88] on tilastolliseen Bayes-tekniikkaan perustuva hierarkinen klusterointijärjestelmä, joka tuottaa syötteenä annetuille alkioille luokituksen, luokille todennäköisyyskuvauksen sekä todennäköisyydet alkioiden kuulumisesta kuhunkin luokkaan.

AutoClassin suurin ero CLUSTER/2- ja COBWEB-järjestelmiin on se, että kukin alkio kuuluu jokaiseen luokkaan jollakin todennäköisyydellä $\mathrm{p}(x_i|x_i \in C_j)$, jossa x_i on jokin alkio ja C_j on jokin luokka. Alkioiden kuuluminen jollakin todennäköisyydellä jokaiseen luokkaan vastaa paremmin todellisuutta, jossa luokkien väliset rajat ovat usein epätarkat ja yksi alkio voi luontevasti olla kahden tai usemman luokan rajamailla. AutoClassissa alkioiden ominaisuudet voivat olla sekä reaalilukuja että diskreettejä arvoja.

Algoritmi jakautuu kahteen osaan, josta toisessa päätellään tarvittavien luokkien lukumäärä ja toisessa luokkia kuvaavat parametrit. Luokkien lukumäärän yläraja pyydetään käyttäjältä syötteenä, mutta AutoClass osaa vähentää luokkien määrää syöteaineistoon sopivaksi. Luokkia kuvaavat parametrit ovat $\vec{\theta}_j$ ja π_j, jossa $\vec{\theta}_j$ sisältää luokan ominaisuudet ja π_j on satunnaisen alkion todennäköisyys sille, että se kuuluisi luokkaan j.

Algoritmia suoritetaan käyttäjän valitseman iteraatiokierroksen verran, jossa kierrosten määrä tyypillisesti on noin 20 [PC88]. Yhden iteraatiokierroksen laskennallinen vaativuus on suuruusluokkaa $I * J * K$, jossa I on alkioiden määrä, J luokkien määrä ja K alkioiden ominaisuuksien määrä.

Järjestelmän suurimpia etuja on se, että ratkaisu perustuu vakaaseen matemaattiseen malliin eikä ad hoc -mittareihin. Tästä johtuen järjestelmä tarjoaa hyvän pohjan järjestelmän käyttämiseen myös muissa yhteyksissä [PC88].

6 Yhteenveto

CLUSTER/2-, COBWEB- ja AutoClass II -järjestelmiä vertaamalla voidaan todeta, että on useita tapoja tehdä käsitteellistä klusterointia. Riippuen siitä, onko painopisteenä ihmiselle tarkoitettujen selkeiden kuvausten vai hyvän luokittelumallin tuottaminen voidaan klustereiden kuvaukseen käyttää hyvin erilaisia menetelmiä. Tässä raportissa esiteltiin kaksi eri menetelmää kuvata klustereita (loogiset konnektiivit ja todennäköisyyskuvaukset). Lisäksi sivuttiin AutoClassissa käytettyä todennäköisyyskuvausta, jossa alkio kuuluu luokkakohtaisella todennäköisyydellä kuhunkin luokkaan. Mikäli tavoitteena on tuottaa ihmiselle ymmärrettävä kuvaus, ei todennäköisyyskuvauksiin perustuvat kuvaukset eivät vaikuta kovin helppotajuisilta, kuten COBWEB-esimerkin yhteydessä todettiin.

Käsitteellisessä klusteroinnissa klusteroinnin laadun mittari perustuu lähtökohtaisesti luokkien ominaisuuksiin (kuinka yksinkertaisia luokan kuvaukset ovat, kuinka hyvin ne kuvaavat luokkaan kuuluvia alkioita ja kuinka paljon luokan kuvaus eroaa muista luokkien kuvauksista). COBWEB:in ja AutoClassin laadun mittarit vaikuttivat selkeämmiltä kuin CLUSTER/2:ssa käytetty viiden eri vaihtoehdon mittari, vaikkakin vaihtoehdot olivat sinänsä selkeän kuuloisia.

Kaikki esitetyt menetelmät löytävät vähintäänkin lokaalin optimin, COBWEB ilmeisesti löytää globaalin optimin, koska algoritmi sijoittaa aina alkiot CU-arvon mukaan laskettuna niille parhaiten sopiviin luokkiin. CLUSTER/2- ja AutoClass-menetelmien osalta globaalin optimin löytäminen on epävarmaa johtuen siitä, että lähtöparametrit voivat olla väärin asetettuja.

Viitteet

[AKJ99] P. J. Flynn A. K. Jain, M. N. Murty. Data clustering: A review. *ACM Computing Surveys*, 31(3):265 – 323, September 1999.

[CCH01] Patrick Clerkin, Pádraig Cunningham, and Conor Hayes. Ontology discovery for the semantic web using hierarchical clustering. In Gerd Stumme, Andreas Hotho, and Bettina Berendt, editors, *Semantic Web Mining workshop, 12th European Conference on Machine Learning (ECML'01), 5th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'01)*, 2001.

[Fis87] Douglas H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2:139 – 172, 1987.

[GWM95] Rober Godin Guy William Mineua. Automatic structuring of knowledge bases by conceptual clustering. *IEEE Transactions on Knowledge and Data Engineering*, 7(5):824 – 828, 1995.

[JH01] Micheline Kamber Jiawei Han. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, 340 Pine Street, Sixth Floor, San Francisco, CA, USA, 2001.

[Leh02] Kari Lehmussaari. Hierarkkinen klusterointi. In *Klusterointimenetelmät-seminaari*, Raportti C–2002–54, Helsingin yliopisto, Tietojenkäsittelytieteen laitos, marraskuu 2002.

[LP88] Robert E. Reinke Leonard Pitt. Criteria for polynomial time (conceptual) clustering. *Machine Learning*, 2:371 – 396, 1988.

[PC88] M. Self et al. P. Cheeseman, J. Kelly. Autoclass: A bayesian classification system. In *Proceedings of the Fifth International Conference on Machine Learning*, pages 54 – 64. Morgan Kaufmann Publishers, 1988.

[RSM83] Robert E. Stepp Ryszard S. Michalski. Learning from observation: Conceptual clustering. In Tom M. Mitchell Ryszard S. Michalski, Jaime G. Carbonell, editor, *Machine Learning*, pages 331–363. Tioga Publishing Company, Palo Alto, CA, USA, 1983.

[Ukk02] Johannes Ukkonen. Aineiston esikäsittely klusterointia varten. In *Klusterointimenetelmät-seminaari*, Raportti C–2002–54, Helsingin yliopisto, Tietojenkäsittelytieteen laitos, marraskuu 2002.

[YPY96] Jungsoon Park Yoo, Chrisila C. Pettey, and Sung Yoo. A hybrid conceptual clustering system. In *Proceedings of the 1996 ACM 24th annual conference on Computer science*, pages 105 – 114. ACM Press, 1996.