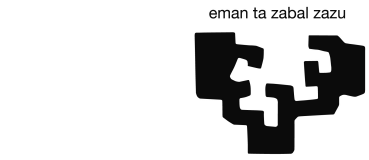


Euskal Herriko Unibertsitatea UPV/EHU

Informatika Ingeniaritzako Gradua

Konputazioa

Gradu Amaierako Proiektua

Web meatzaritzako sistema baten doitzea eta haren aplikazioa testuinguru berrietara

---

Egilea

Lierni Sestorain Saralegui

Zuzendaria

Olatz Arbelaitz Gallego

2015/06/23

Hitzaurrea

Dokumentu hau 2015. urtean Lierni Sestorain Saraleguik, UPV/EHU unibertsitateko Donostiako Informatika Fakultatean Informatika Ingeniaritzako Graduan ikasle naizenak, egindako Gradu Amaierako Proiektuari dagokio. Proiektuaren zuzendaria Olatz Arbelaitz da, Donostiako Informatika Fakultateko irakaslea.

Proiektua *ALDAPA* taldean egin dugu. *ALDAPA* Euskal Herriko Unibertsitateko Donostiako Informatika Fakultateko ikerketa-taldea da. Algoritmoak, datu meatzaritza eta paralelismoa aztertzen ditu batez ere (*ALgorithms, DAta mining and PArallelism*).

Taldeak web meatzaritzako sistema bat garatu du erabiltzaile berrientzat estekak aurreikusteko helburuarekin, beste batzuen artean. Hori dela eta, proiektu honetan, sistemaren atal hori birfindu dugu, inplementatuta dauzkan algoritmo batzuk aukera berriekin ordezkatuz eta beste batzuen ezaugarriak sistemara egokituz, eta ingurune berri baten (*Discapnet* ezintasunak dituzten pertsonentzako webgunearen) datuak aplikatu dizkiogu; izan ere, Bidasoa Turismo webgunearekin erabili ohi da sistema. Bukatzeko, lortutako emaitzetatik ondorioak atera eta etorkizunerako lana finkatu dugu.

Hitz gakoak: Web meatzaritza, esteka, aurreikuspena, eredu, bilaketa.

Eskerrak

Hasteko, eskerrak eman nahi dizkiot *ALDAPA* taldeari, haiekin lan egiteko aukera emateagatik eta proiektuan zehar eskainitako laguntza guztiagatik, baina batez ere proiektuaren zuzendari den Olatz Arbelaitzi, proiektu osoan zehar emandako aholkuengatik eta orientazioagatik.

Bestetik, fakultateko nire ikaskideei edo graduan zehar harremana izan dudan guztiei, azken lau urteetan pasatako momentu guztiengatik.

Amaitzeko, familiari eta lagun minei, beti ondoan egoteagatik.

Aurkibidea

Irudien Zerrenda

Taulen Zerrenda

1. kapitulua

Proiektua

1.1 Sarrera

Gaur egun, interneten erabilera zabala dela eta, zenbait zeregin berri sortu dira. Horien artean, esteken aurreikuspena dago, alor bat baino gehiagotan erabiltzen den zeregina. Adibidez, segurtasun agentziek, webgune maltzurren arteko erlazioak aztertuz, ikuskatu ez dituzten orriak bila ditzakete; sare sozialetan, lagunak, gustoko orriak edota ekitaldiak aurkitzeko erraztasunak eskeintzen dira; medizina eta biologian, esteken aurreikuspenari esker, besterik ezean, ikerketa luze eta garestien ondorioz aurkituko liratekeen erlazioak bilatzen dira...

Finean, sare baten bitartez adieraz daitekeen edozein ingurune naturalek aurreikuspen metodo baliagarri bat dauka eta metodo hori ingurune horretako galdera garantzitsuren bati erantzuteko gai da.

Hori dela eta, *ALDAPA* ikerketa-taldeak erabiltzaile-profilak sortu eta estekak aurreikusteko sistema sortu zuen. Horren bitartez, *Bidasoa Turismo* webguneko (hemendik aurrera, BTw) datuekin, erabiltzaileak sistemak sortutako profil desberdinetan sailkatzen dira eta gero, profil horrek erabili ohi dituen URL-ak proposatzen zaizkio erabiltzaile berriari.

Abiapuntu moduan sistema hori izanda, proiektu honen helburua estekak aurrikusteko sistemaren zatia hobetzea edo birfintzea da. Horretarako, proiektuan zehar, gainbegiratutako eta gainbegiratu gabeko sailkatzaile desberdinak aztertzen ditugu, sistemarentzako egokiak zein izan daitezkeen hausnartuz. Ondoren, probak erabaki, aldagaiak finkatu eta

exekuzioak martxan jartzen ditugu, emaitzak ikuskatu ahal izateko. Bukatzeko, lortutako emaitza guztiak kontuan hartuz, ondorioak atera eta *ALDAPA* taldeari sistema hobetua, ahalik eta emaitza hoberenak ematen dituena, eman diogu.

Dokumentu hau Euskal Herriko Unibertsitateko Informatika Fakultateko Konputazio alorreko Gradu Amaierako Proiektuaren memoria da, Lierni Sestorainek idatzia. Dokumentu honetan zehar, proiektuaren inguruko azalpen guztiak ematen dira. Aurrerago ikus daiteke memoriaren egitura (ikus 1.4).

1.2 Motibazioa

Proiektu hau aurrera eramateko interes nagusiak bi izan dira: ikerketa-taldearena eta egilearena.

Alde batetik, *ALDAPA* ikerketa-taldeak esku artean daukan sistema indartu nahi du, lortzen dituen emaitzak hobeagotuz, baita bere aplikazioa *Bidasoa Turismo* webgunearen datuetatik haratago eraman ere.

Bestalde, Gradu Amaierako Proiektuan, algoritmo desberdinekin eta datu-kantitate handien prozesamenduan egin nahi dugu lana. Izan ere, konputazioa alorretik, bi eremu gustokoenetakoak izan dira. Hainbat aukera aztertu ondoren, proiektu hau izan da gustokoena.

1.3 Proposatutako soluzioa

Sistemak duen momentuko konfigurazioa jakinik (ikus 3.2 atala), PAM *clustering* algoritmoaz gain, hurrengo orrietan *clustering* algoritmo hierarkikoen funtzionamendua aztertzen dugu, horretarako SHAN (*Sequential Hierarchical Agglomerative Non-overlapped*) algoritmoa erabiliz, *single*, *ward*, *complete* eta *average linkage* aukerekin. Horrela, gainbegiratu gabeko algoritmo desberdinen funtzionamendua aztertzen dugu, baita horrek *clustering*-ean duen eragina ere.

Bestalde, sistemaren bigarren atalean, erabiltzaile berriei profilak esleitu eta horren arabera URL-ak proposatzeko KNN algoritmoa erabiltzen da, ondorioz, zati horretara heltzean, gainbegiratutako algoritmo honen funtzionamendua ere ikuskatzen dugu. Gainera, azterketaren arabera, sistema osoaren eraginkortasuna hobe dezaketen aldaketak identifikatu, inplementatu eta probatzen ditugu, bukaeran, sistema hoberenarekin gelditzeko asmoz.

1.4 Memoriaren egitura

Proiektua Lehenengo kapituluan, proiektuan daukagun arazoa azaltzen dugu. Baita proiektu hau aukeratzera bultzatu gaituzten arrazoiak ere. Gainera, bilatu nahi dugun soluzioa aipatzen dugu.

Proiektuaren Kudeaketa Plana Proiektuaren kudeaketaren nondik norakoak azaltzen dira: proiektua nola egituratzen den eta nola inplementatzen den. Helburuak, irismena eta entregatu beharreko dokumentuak deskribatzen dira, interesatuen paperekin eta betebeharrekin batera.

Jatorrizko Sistema Kapitulu honetan, jatorrizko sistemaren inguruko azalpenak ematen dira: funtzionamendua, erabiltzen dituen algoritmoen deskribapena, ebaluazioa nola egiten duen eta, bukatzeko, lortzen dituen emaitzak. Kapitulua ixteko, jatorrizko sistemaren ahuleziak adierazten dira, proiektuarekin aurrera jarraitu ahal izateko.

***Clustering* atalean aldaketak** PAM *clustering* algoritmoa aldatzeko jarraitutako prozesua azaltzen da; ordezko algoritmoen ikerketa, inplementazioa, probak eta ondorioak, besteak beste.

Profil antzekoena bilatzeko atalean aldaketak Sistemaren atal honen ahuleziak adierazi ondoren, bakoitzaren arrazoiak, inplementazioa, probak eta ondorioak azaltzen dira.

Sistema beste testuinguru batean aplikatzea Lortutako sistemaren bertsio hoberena beste ingurune bateko datuekin probatzen da.

Jarraipena eta kontrola Atal honetan, proiektuan zehar egindako lana berrikusten da, hasierako plangintza eta helburuak lortutako emaitzarekin konparatuz.

Ondorioak Proiektuaren osotasunetik ateratako ondorioak azaltzen dira.

2. kapitulua

Proiektuaren kudeaketa plana

Kapitulu honetan proiektuaren helburuak lortzeko egindako plangintza azaltzen da, baita jarraipena eta kontrola nola egin den ere. Plangintzaren barnean, proiektuak dituen ataza desberdinak identifikatzen dira eta egutegi eta ordutegi bat zehazten da. Gainera, baliabideen, kalitatearen, komunikazioen eta arriskuen kudeaketa-planak ere azaltzen dira.

Hauek dira kapitulu honetako atalak:

- Helburuak
- Irismena
- Entregatu beharrekoak
- Antolakuntza-egitura
- Egutegia eta ordutegia
- Arriskuak
- Interesatutako parteak

2.1 Helburuak

Proiektuaren helburuak bi multzotan banatzen dira: alde batetik, proiektuaren egilearen helburuak eta, bestetik, *ALDAPA* taldearen helburuak. Ikus ditzagun jarraian:

Ikaslearen helburuak

Ikaslearen helburuak hiru motakoak izan daitezke. Helburu nagusiak proiektuko produktua, hots, sistema hobetzeko bete beharreko helburuak dira; helburu gehigarriak, hel-

buru nagusien egikaritzapena errazteko edo Gradu Amaierako Proiektua osatzeko egin beharrekoak, eta, amaitzeko, hautazko helburuak, proiektua hobetzeko bete beharreko helburuak.

- Helburu nagusiak:
  - Orokorrak:
    * Graduari lotutako konpetentziak aplikatzea, bilaketa egiteko gaitzea eta datu garrantzitsuak kudeatu, antolatu eta interpretatzea.
    * Konputazio espezialitatean ikasitakoa praktikan jartzen ikastea.
  - Espezifikoak:
    * Web meatzaritzako sistema baten funtzionamendua ulertzea eta ikastea; kasu honetan, esteken aurreikuspena egiten duen sistemarena, *ALDAPA* taldeak diseinatua.
    * Gainbegiratutako eta gainbegiratu gabeko algoritmoen funtzionamendua ulertzea.
    * Aukeratutako algoritmoekin probak egin eta emaitza erabakigarriak lortzea.
    * Eraginkortasun hoberena lortzen duen algoritmoarekin sistema hobeago bat prestatzea.
    * Emaitza moduan lortutako sistema *Discapnet* inguruneko (behar bereziak dituzten pertsonentzako ingurunea) datuei aplikatzea.
- Helburu gehigarriak:
  - *Aldapa* taldearen sistemaren inplementazioa editatu ahal izateko, programazio-ingurunea eta lengoaia sakonago lantzea.
  - LaTeX editorearen erabileran sakontzea.
  - Bakarrik lan egiteko gaitasuna indartzea.
  - Egindako lan guztia azaltzen duen dokumentu mardul bat idaztea.
- Hautazko helburuak:
  - Sistemaren orokortasuna hobeto neurtzeko, datuak *Discapnet* inguruneaz gain, *gipuzkoa.eus* inguruneko (erakunde-ingurunea) datuei ere aplikatzea.

Ikerketa taldearen helburuak

Ikerketa-taldeak bi helburu nagusi dauzka:

- *ALDAPA* taldeak diseinatutako web meatzaritzako sistema hobetzea, esteken aurreikuspen hobea egiteko.
- Sistema hobetua beste ingurune batzuetan probatzea eta ondorioak ateratzea.

2.2 Irismena

Proiektu honen helburu nagusia, aurretik esan bezala, *ALDAPA* ikerketa-taldeak daukan web meatzaritzako sistema hobetzea da, horretarako egokiena den algoritmoa aurkituz, eta, ondoren, sistema hori beste ingurune batean aplikatzea. Hortaz, irismena helburu horien arabera definitu dugu. Jarraian, betekizunak, mugak eta lan-metodologia azaltzen ditugu.

2.2.1 Betekizunak

Proiektuak Donostiako Informatika Fakultateak zehaztutako arauak bete behar ditu. Hurrengo zerrendan garrantzitsuenak zerrendatu ditugu:

- Proiektua defendatzeko, graduko gainontzeko kreditu guztiak gainditua izan behar dira.
- Proiektuak 12 kreditu ditu; beraz, gutxienez, 300 orduko dedikazioa eskatzen du.
- Proiektuko zuzendariak Informatika Fakultateko irakaslea izan behar du.
- Proiektuak Informatika Fakultatean eskaintzen diren espezialitateetako baten ingurukoa izan behar du; kasu honetan, konputazio espezialitatekoa.

2.2.2 Mugak

Kudeaketak ez dio proiektuari denbora gehiegi kendu behar; beraz, txosten honetan, kudeaketarako datu garrantzitsuenak bakarrik jasotzen dira.

Proiektuak 300 orduko dedikazioa eskatzen duenez, algoritmo bakoitzarekin egindako proba kopurua ere mugatu behar da. Kopuru hori, ordea, ezin da orain mugatu, ez baitakigu proba bakoitzak zenbat denbora behar duen. Hori dela eta, lehenengo probekin hastean, balioespen bat egin eta kopuru bat mugatu behar da.

Hala ere, nahiz eta Gradu Amaierako Proiektua den, kasu honetan, 450 ordu inguru lan egitea aurreikusten da, Espainiar Ministeritzako lankidetza-beka jaso duen proiektuetako bat baita.

2.2.3 Lan-metodologia

Zer eta nola egin?

Proiektua osatu ahal izateko, zenbait fase desberdinetatik pasa behar dugu:

Gaian kokatzea. Lehenengo zeregina bibliografia aztertu eta antzeko proiektuetako artikuluak irakurtzea da.

Sistema aztertzea. Lehenengo eta behin *ALDAPA* taldeko kideen azalpenak jasotzea lagungarria bada ere, sistema ulertzekotan, beharrezkoa da norberak sistema aztertzea.

Lehenengo frogak egitea. Jatorrizko sistemarekin probak egitean, sistemaren funtzionamenduarekin ohitzea lortzen dugu, baina ez hori bakarrik, baita aurrerago konpondu beharreko arazoak hautematea ere.

Konponbideak proposatzea. Sistemaren arazo nagusiak topatu ondoren, bakoitzarentzako konponbideren bat bilatu behar dugu; horretarako, berriro, ikerketa artikuluak irakurri behar ditugu, arazoaren inguruan gehiago informatzeko.

Inplementazioa. Pixkanaka, arazoak konpontzeko sistemaren bertsio berriak inplementatu eta probatu behar dira, emaitzetatik ondorioak ateratzeko.

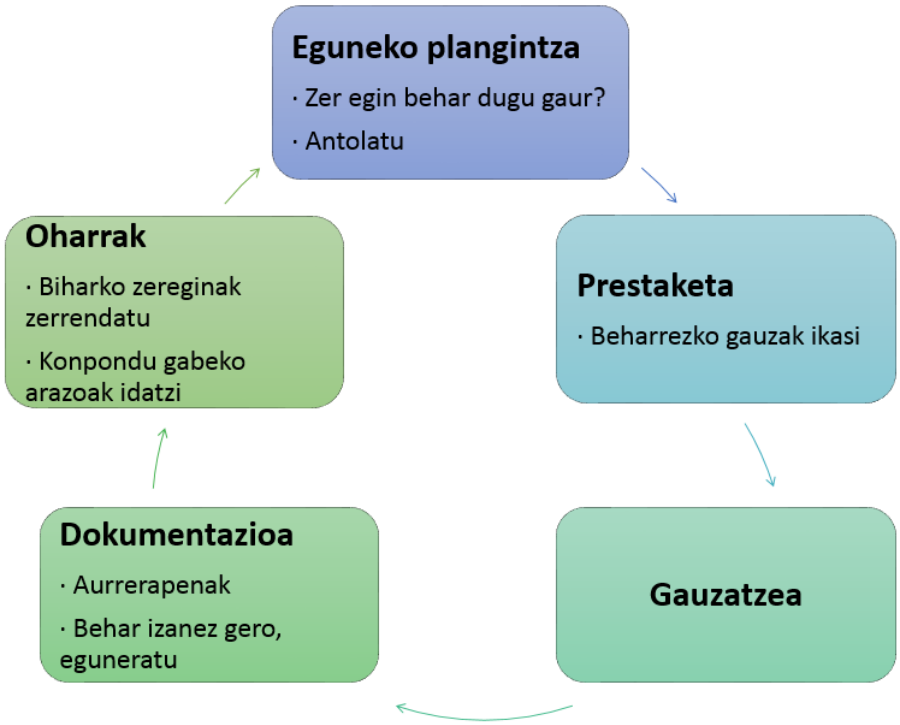
Horretaz gainera, proiektuaren bizi-ziklo osoan zehar, zuzendariarekin bilerak egin behar dira, gutxienez, astean bat, baina, beharren arabera, gehiago adosteko aukera dago. Oro har, bilera hauek ikaslearen eta zuzendariaren artekoak badira ere, hirugarren pertsonekin ere elkar gaitezke behar izanez gero, esaterako, *ALDAPA* taldeko kideekin, sistemarekin arazorik balego.

Eguneroko lan-metodologia

Egunero ordu dezente eskainiko dizkiogunez proiektuari, motibazioa mantentzeko eguneroko iterazio-plangintza jarraitzea erabaki dugu. Horretarako, metodo batzuk begiratu ditugu interneten eta 2.1 irudian ikus daitekeen emaitzara iritsi gara. Modu horretan, lanaldiaren hasieran, egunean zehar egin beharreko zereginak definitzen ditugu. Ondorioz, egunero ideia desberdin bat daukagu buruan eta lana ez da hain errepikakorra egiten. Hala ere, egunen batean zereginen batekin blokeatuta geratzen bagara, nahikoa da beste zerbaitetan zentratzea, momentukoa hurrengo zereginetan idatziz, aurrerago beste ikuspuntu batetik begiratzeko.

Iterazio-zikloa azaltzen dugu jarraian. Lehenengo minutuetan, egun osorako zereginak definitzen ditugu, eta, hortik, agenda antolatzen dugu. Beti beharrezkoa ez bada

ere, batzuetan zereginak egiteko prestakuntzarentzat tartetxo bat behar dugu: artikuluak irakurri, azken berrien gainean egon, kontzeptu jakin batzuk landu... Ondoren, mamiari ekin behar diogu. Zati horretan sartzen dira sistemaren aldaketak, probak edo proiektuaren emaitzarekin zerikusia duen beste edozer; bilerak eta jarraipena, besteak beste. Gero, egindako aurrerapenak dokumentatu behar ditugu, baita aurreko egunetako dokumentazioa eguneratu ere, behar izanez gero. Bukatzeko, egin gabe gelditu diren lanak eta hurrengo egunean egin ditzakegun zereginak zerrendatu behar ditugu.



Irudia 2.1: Eguneroko iterazio zikloa.

Hardware azpiegitura

Proiektu honetan lan egiteko ez da hardware azpiegitura berezirik behar, hortaz, edozein ordenagailu erabil daiteke, programazio-ingurune egokia instalatuta badauka. Hala ere, *ALDAPA* taldeko ikertzaileek erabiltzen duten laborategian *Pentium x* bat dago erabilgarri eta, probek eskatzen duten konputazio-ahalmena kontuan hartuz, laborategiko ordenagailua aukeratu dugu proiektuarekin lan egiteko. Hauek dira utzitako ordenagailuaren ezaugarri garrantzitsuenak:

Memoria: 8 GiB
Prozesagailua: Intel Core i7-2600 CPU @ 3.40 GHz x 8

Erabilitako softwarea

Proiektua osatzeko honako softwarea erabili da:

Sistema eragilea: Ubuntu 14.04 LTS
Programazio ingurunea: Eclipse
Programazio lengoaia: Java
Probetarako: Terminator eta Java
Plangintza eta kudeaketarako: Gantt Project eta Microsoft Office paketea
Memoriaren dokumentaziorako: Latex eta Tex motorra (Texworks)
Aurkezpena egiteko: Microsoft Power Point
Lanaren kopiak izateko: Allway Sinc, Drive eta Dropbox

2.3 Entregatu beharrekoak

Hauek dira proiektua amaitzean entregatu beharrekoak:

Memoria Proiektutik sorturiko emangarri nagusia memoria bera da. Dokumentu honek Gradu Amaierako Proiektu baten memoriaren gidalerroa jarraitu behar du, formatu egokian egon behar du eta egindako lan guztia azaldu behar du. Euskal Herriko Unibertsitatearen *ADDI* plataformara igo behar da Ekainaren 24a baino lehen, zuzendariaren oniritzia jaso ondotik.

Aurkezpena Proiektuaren defentsa egiteko prestatutako gardenkiak, nahiz eta aurkezpena entregatzea ez den nahitaezkoa.

Sistema *ALDAPA* taldeari web meatzaritzako sistema hobetua, bertsio eraginkorrena, eman behar zaio.

Gradu Amaierako Proiektua Jabego Intelektualaren Legeak babestuta dago; beraz, proiektuaren jabea sortzailea da (ikasleak kontrakoa esaten ez badu). Hortaz, memoriak eta aurkezpenak x lizentzia daukate.

2.4 Antolakuntza

Helburuak zehaztuta dauzkagunez, lanaren deskonposaketa-egitura (LDE hemendik aurrera) erabaki dugu. Horren bitartez, proiektua osatzen duten atazak zeregin txiki eta zehatzagoetan banatu ditugu. Horrela, ondoren egin beharreko plangintza errazten da, ataza bakoitzerako beharrezko denboraren balioespen egokiagoa egitea ahalbidetzen baitigu. Datozen azpiataletan, LDE eta egutegia azaltzen dira.

2.4.1 Lanaren Deskonposaketa Egitura (LDE)

Hurrengo lerroetan, proiektuaren bizi-zikloa osatzen duten ataza nagusiak identifikatzen dira: kudeatzea, ikertzea, probak egitea, sistema beste ingurune batean aplikatzea eta proiektua ixtea. Noski, ataza bakoitzak bere barnean hainbat zeregin jasotzen ditu. 2.2 irudian, proiektuaren LDE ikus daiteke.

Jarraian dauden azpiataletan, ataza nagusiak aztertzen ditugu sakonago.

Kudeaketa

Kudeaketak, bere barnean, zeregin ugari jasotzen ditu. Hala ere, guztiak hiru multzotan bildu ditugu:

- Plangintza
- Kontrola
- Jarraipena

Plangintzaren barnean, proiektua aurrera eramateko egin behar diren zenbait prestaketa bildu ditugu: egutegia eta orduak estimatzea, beharrezko baliabideak aurreikustea, proiektuaren kalitatea definitzea, arriskuak aurreikustea eta kontingentzia-plana prestatzea, nagusiki.

Kontrolaren atalean, berriz, lanaren kalitatearen, arriskuen eta baita aldaketen kontrolarekin zerikusia duten atazak ere elkartu ditugu, berrikuspenak, batez ere.

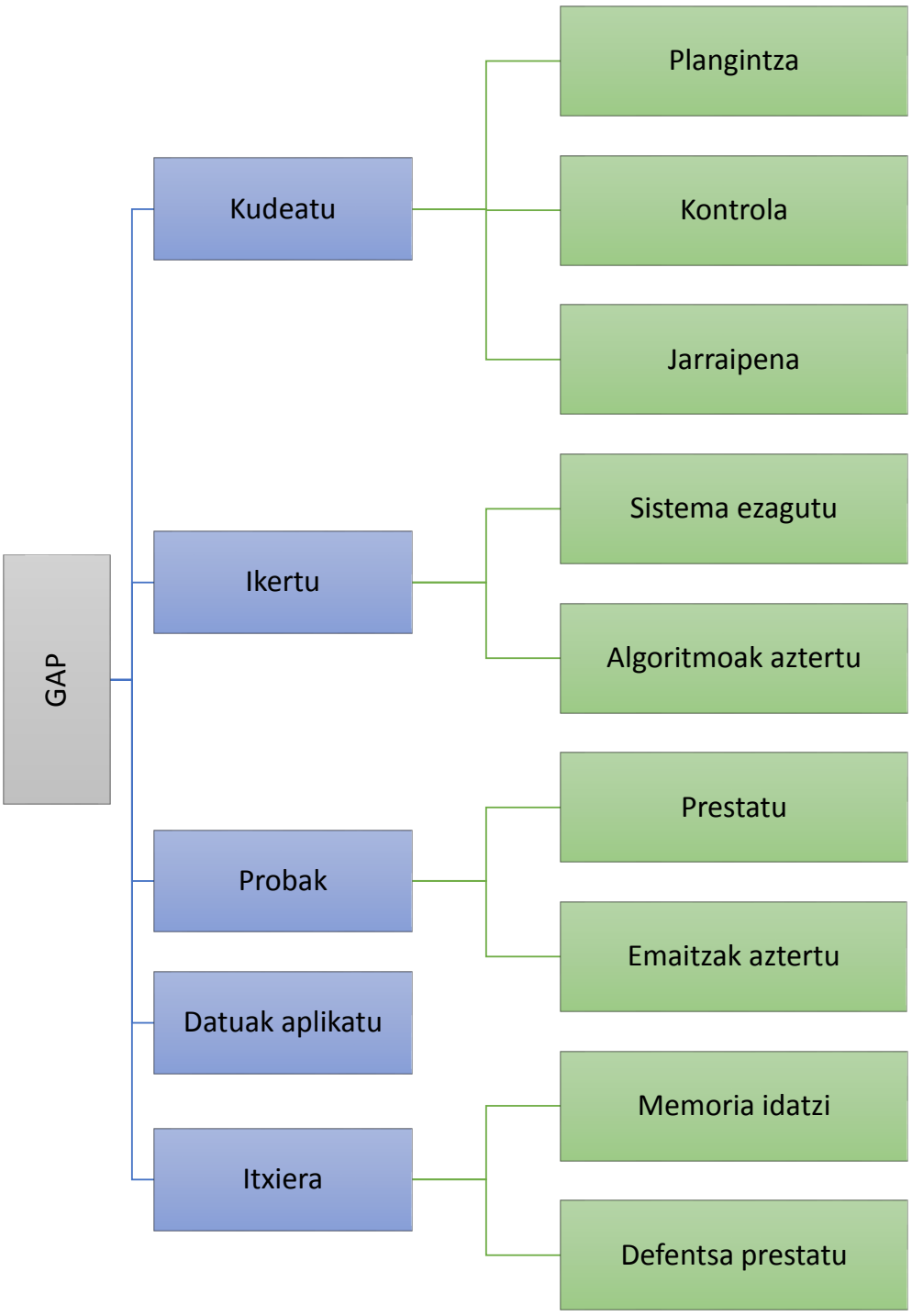
Bukatzeko, jarraipena ataza proiektuaren zuzendariarekin egindako bilerek osatzen dute gehien bat, baina elkarrekin hartutako erabakien arabera plangintza eguneratu behar da.

Ikerketa

Proiektuaren testuingurua hobeto ulertzeko asmoz, artikuluak eta ikerkuntza-txostenak irakurtzea komeni da. Kasu honetan, ikerketzeko lana honela banatuko dugu:

- Sistema ezagutzea
- Algoritmoak aztertzea

Atal honi norbanakoaren prestakuntza dei diezaiokegu. Hasteko, sistemaren funtzionamendua azaltzen duten artikuluak irakurri behar ditugu, baita antzeko sistemei buruzkoak ere. Artikulu horiek ondo ulertzea oso garrantzitsua denez, horietan agertzen diren edo



Irudia 2.2: Proiektuaren Lanaren Deskonposaketa Egitura

.

sistemak barneratzen dituen kontzeptu ezezagunak landu behar ditugu. Jarraian, sistema ordenagailuan martxan jarri ondoren, sistemaren ezagutza sakonagoa egiteko, zenbait proba egin behar ditugu, sistemaren funtzionamenduarekin ohitzen joateko helburu bakarrarekin. Sistemaren ahulguneak identifikatzeko, hala ere, jatorrizko sistema era metodologikoan exekutatu behar dugu.

Bigarren atalean, berriz, sistemaren ahuleziak konpontzeko egin beharreko ikerketa bildu dugu. Bertan, batez ere, ahulezientzat proposatutako konponbideak inplementatzeko landu beharreko algoritmoen ikerketa biltzen dugu, bai gainbegiratu gabeko algoritmoena eta bai gainbegiratutakoena, nahiz eta beharrezko beste edozein lanketa ere (arazoentzako konponbideak bilatzeko, esaterako) hemen sartzen dugun.

Probak

Probetan ere, bi atal nagusi dauzkagu:

- Prestaketa
- Emaitzak aztertzea

Algoritmoak aztertu ondoren, sistemari hobekien datozkionak zein diren erabaki behar dugu, baita horiekin egin daitezkeen konbinazioak ere, eta mota bakoitzeko algoritmoentzat parametroak eta aldagaiak finkatu. Gauza horiek zehaztu ondoren, probak sisteman txertatzeko inplementazioa aldatu behar dugu eta exekutatu, hasierako sistemari aplikatutako proba metodologiko berak errepikatuz. Prozesu hori guztia "prestaketa" deitu dugun atalean sartzen dugu. Jarraian, sistemak ematen dituen emaitzak bildu eta, bukatzean, denak konparatu behar ditugu, hoberena aukeratu ahal izateko.

Aipatzekoa da, edonola ere, probena prozesu berrelikatua izan daitekeela, proba batzuen emaitzek beste proba mota bat egitera eraman gaitzakete eta.

Datuak aplikatzea

Sistemak beste ingurune batzuetan duen eraginkortasuna eta erabilgarritasuna ikusteko, sistema beste ingurune batzuetako datuekin probatzea beharrezkoa da. Hori dela eta, hasteko, sistema *Discapnet* inguruneko (behar bereziak dituzten pertsonentzako ingurunea) datuekin probatu behar dugu, eta, gero, nahikoa denbora badaukagu eta datuak lortzen baditugu, hobekuntza moduan, baita *gipuzkoa.eus* erakunde-ingurunearen datuekin ere.

Atazarekin bukatu baino lehen, ondorioak atera behar ditugu: sistema espero bezain ondo moldatu den edo ez, zein izan daitezkeen arrazoiak, konponbideak proposatu...

Itxiera

Azken atazan, zereginak bi multzotan bildu ditugu:

- Memoria idaztea
- Defentsa prestatzea

Proiektua amaitzear dagoenean egiten den ataza honetan memoriarekin eta defentsarekin zerikusia duten zereginak jasotzen ditugu. Honakoak aipatzen dira, besteak beste: egindako lana, sortutako arazoak eta nola egin zaien aurre, zer ikasi den proiektuan zehar eta ondorioak. Horretaz gain, proiektuaren bizi-zikloan ikasitako lezioak zerrendatu eta azaldu behar ditugu.

Bukatzeko, defentsarako prestaketak egin behar dira, batez ere, aurkezpenerako materiala eta saioa prestatu.

2.1 taulan ataza nagusien deskribapenak bildu ditugu.

2.4.2 Egutegia

Proiektuaren txostena entregatzeko azken eguna Ekainaren 24a da. Beraz, ordurako proiektu osoak bukatuta egon behar du eta hurrengo bi asteetan, defentsa prestatzeko aukera daukagu.

Plangintza zehaztuagoa eta egutegia hurrengo atalean azaltzen ditugu.

2.5 Proiektuaren atalak

Proiektuaren bizi-zikloa lau atal nagusiz edo fasez osatuta dago:

- Hasieratzea
- Plangintza
- Gauzatzea
- Itxiera

Atal bakoitzari eskainitako esfortzua proiektuan aurrera egin ahala aldatuz doa. Gainera, baliteke bata bestearekin gainjartzea. Izan ere, lehenengo egunetan hasieraketarekin lan egiten da batez ere, baina baita plangintzarekin ere. Plangintza bukatu ondoren, proiektua gauzatzearekin jarri behar dugu buru-belarri, baina jarraipena egin behar denez, tarteka plangintza pixka bat ikutu behar dugu. Bukatzeko, proiektua bukatzen goazela, itxiera fasearekin hasi behar da pixkanaka.

Kodea	Zeregina	Azalpena
1 Kudeaketa		
1.a	Plangintza	Egutegia eta orduak estimatu, beharrezko baliabideak aurreikusi, kalitate-plana, arriskuak kudeatzeko plana eta kontingentzia plana garatu.
1.b	Kontrola	Kalitatearen, arriskuen eta aldaketen kontrola eramateko aldizka azterketa egin.
1.c	Jarraipena	Aldizka proiektuaren zuzendariarekin (edota *ALDAPA* taldearekin) bildu, eta, behar bada, plangintza eguneratu.
2 Ikerketa		
2.a	Sistema ezagutzea	Artikuluak irakurri, ulertzen ez diren kontzeptuak landu, sistema ordenagailu propioan martxan jarri, sistema ulertzeko proba batzuk egin eta emaitzak interpretatu.
2.b	Algoritmoak aztertzea	Arazoak konpontzeko inplementatu beharreko algoritmoei buruzko artikuluak irakurri.
3 Probak		
3.a	Prestaketa	Probak zein algorimorekin egingo diren erabaki, bakoitzarentzako aldagaiak finkatu eta inplementatu.
3.b	Emaitzak aztertzea	Lortutako emaitzak begiratu eta ondorioak atera.
4 Datuak aplikatzea		
-	-	Sistema hoberena *Discapnet* eta, ahal izanez gero, *gipuzkoa.eus* inguruneetako datuekin probatu eta ondorioak atera.
5 Itxiera		
5.a	Memoria idaztea	Egindako lana, sortutako arazoak eta nola egin zaien aurre, proiektuan zehar ikasitakoa eta ondorioak txostenean azaldu.
5.b	Defentsa prestatzea	Defentsarako beharrezko materiala prestatu eta saioak egin.

Taula 2.1: Lanaren Deskonposaketa Egituraren ataza nagusien azalpena.

Hasieratzea

Atal honi hasiera emateko, proiektuan egin beharrekoa ulertu behar dugu eta lanari ekin. Horretarako, proiektuaren zuzendariarekin eta ikerketa-taldearekin lanaren nondik norakoak adostu eta zereginak definitu behar ditugu. Proiektuaren helburuak, irismena eta inplementazioa finkatu behar ditugu, interesatuek zer eta noiz jasoko duten jakin dezaten.

Plangintza

Zuzendariarekin bilera egin ondoren egin beharrekoa garbi daukagula, atazak, ekintzak eta zereginak identifikatu behar ditugu. Gainera, horiek aurrera eramateko behar den esfortzua balioetsi behar dugu, egutegian kokatu eta sor daitezkeen arazoak aurreikusi behar ditugu. Noski, aurreikustea bakarrik ez da nahikoa, kontrolatzeko eta aurre egiteko planak ere garatu behar ditugu.

Gauzatzea

Proiektua definituta dagoenean eta plangintzaren lehen bertsioa bukatu dugunean, proiektuaren mamiarekin hasi behar dugu. Hau da, proiektuaren helburuak lortzeko egin beharreko lanari ekin behar diogu. Atal honetan, proba desberdinak erabaki, sistemaren bertsio desberdinak sortu, probak martxan jarri, datuak aplikatu... egin behar ditugu. Hala ere, zeregin horiek denak egin ahal izateko, nahitaezkoa izango da lehenik gure burua prestatzea, horretarako beharrezkoa delarik zenbait gauza ikertzea eta ikastea.

Noski, gauzatzea ez dago martxan bakarrik egoterik, prozesu osoan zehar, kontrola eta jarraipena egitea beharrezkoa baita.

Itxiera

Proiektuaren bizi-zikloaren amaieran, txostena idatzi eta entregatu behar dugu, aurkezpenaren prestaketarekin batera.

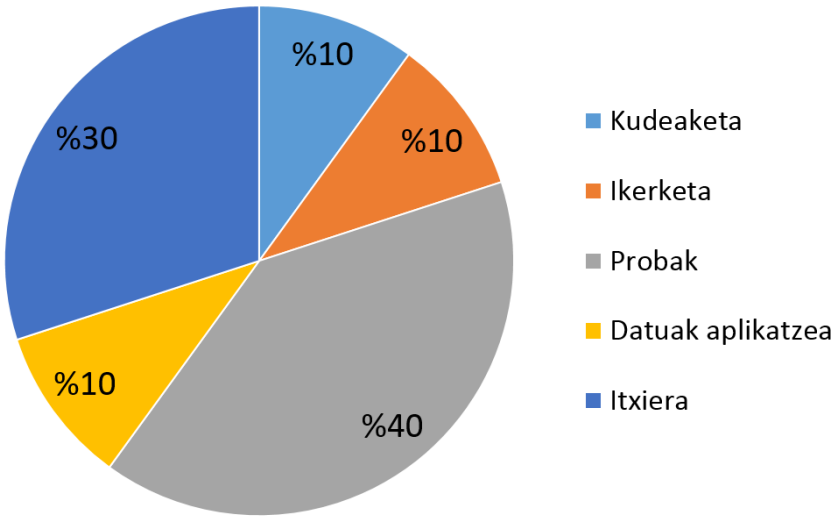
2.6 Proiektua gauzatzeko plana

Atal honetan, proiektua osatzeko egin beharreko lanaren denbora eta esfortzua estimatzen ditugu. Ondoren, ordutegi edo programa bat proposatu dugu, LDEan (ikus 2.2 irudia) adierazitako zeregin bakoitza noiz egin adierazten duena.

2.6.1 Egutegia

Proiektuaren atal nagusietarako beharrezko orduen estimazioak antzeko proiektuekin izandako esperientziatik egin ditugu, beraz, desbiderapenak egoteko aukera asko daude, ez baitugu inoiz hainbeste orduko proiektu batean parte hartu.

2.3 irudian, ataza nagusien lan-kargaren balioespena erakusten da, ehunekotan, grafikoki. 2.4 irudian, aldiz, zeregin bakoitzaren estimazioaren informazio gehiago adierazi dugu: beharrezko denbora, hasiera- eta bukaera-data eta tartean lan egiteko dauzkagun egun-kopurua. Ikus daitekeenez, 430 ordu estimatu ditugu. Proiektuarentzat, gutxienez, 300 ordu lan egin behar badira ere, lankidetza-beka dela eta, 450 ordukoa da minimoa. Beraz, falta diren 20 orduak eta hortik gorakoak, koltxoi moduan erabiliko ditugu, ezustekoak gertatzen badira ere.



Irudia 2.3: Ataza nagusien karga ehunekotan.

Egindako lanaren ahalik eta informazio gehien eta ahalik eta zehatzena gordetzeko, kalkulu-orri bat sortu dugu, egun bakoitzean zenbat ordu eta zertan eman ditugun adierazteko. Horrela, jarraipen estuagoa egin dezakegu eta amaieran, grafikoak eta ondorioak ateratzeko aukera daukagu.

OHARRA: Benetako lan-orduen eta estimazioen arteko konparazioa 7. kapituluan ikus daiteke.

2.6.2 GANTT diagrama

2.4 irudiko taulan ikusi ditugun datei jarraituz, proiektuko atazekin sortu dugun Gantt diagrama 2.5 irudian ikus daiteke.

2.7 Kalitate-plana

Proiektuaren kalitate-maila minimoa betetzen dela ziurtatzeko, egiaztapen-zerrenda bat sortu dugu, proiektuaren ezaugarriak kontuan hartzen dituena (ikus 2.6 irudiko zer-

		Estimazioa			
Kodea	Ataza	Denbora	Hasiera	Egunak	Bukaera
-	-	430:00	Urt-12	114	Uzt-7
1	KUDEAKETA	43:00	Urt-12	105	Eka-24
1.a	Plangintza	18:00	Urt-12	5	Urt-16
1.b	Kontrola	10:00	Urt-19	100	Eka-24
1.c	Jarraipena	15:00	Urt-19	100	Eka-24
2	IKERKETA	43:00	Urt-12	29	Ots-20
2.a	Sistema ezagutzea	13:00	Urt-12	9	Urt-23
2.b	Algoritmoak aztertzea	30:00	Urt-26	20	Ots-20
3	PROBAK	172:00	Ots-2	53	Api-30
3.a	Prestaketa	120:00	Ots-2	49	Api-24
3.b	Emaitzak aztertzea	52:00	Ots-16	43	Api-30
5	DATUAK APLIKATZEA	43:00	Mai-4	20	Mai-29
6	ITXIERA	129:00	Mai-4	47	Uzt-7
6.a	Memoria idaztea	110:00	Mai-4	38	Eka-24
6.b	Defentsa prestatzea	19:00	Eka-25	9	Uzt-7
	GUZTIRA:	430:00:00			

Irudia 2.4: Jarraipen-taula.

renda). Proiektuaren jarraipen- eta kontrol-prozesuetan zehar, zerrenda hau betetzen dugu, proiektuaren egoera zein den ikusteko. Gainera, arriskuak kudeatzeko plana eta kontingentzia-plana ere definitu ditugu.

2.7.1 Eraginkortasunaren adierazleak

Proiektuaren helburuen egikaritzea kalitatearekin parekatuta neurtzeko, ezaugarri adierazgarri batzuk definitu ditugu. Aldi berean, ezaugarri hauek proiektuaren arrakasta neurtzeko balio dute.

Adierazle kuantitatiboak:

- Ordutegia mantentzea: proiektuaren helburua lana aurreikusitako egutegiarekin bat bukatzea da, epeak betez.
- Lan egindako orduak: lan hau betetzeko denbora plangintzan estimatuta dago. Hala ere, desbiderapenak gertatzen direnez, 450 izan ordez, %10 gutxiago edo gehiago izatea onartzen dugu.
- Proba-kopurua: izatez adierazgarria ez bada ere, egindako probak kalitate onekoak direla suposatuz, proba-kopuruak proiektuaren kalitatea neurtzeko balio du. Mo-

mentuz proba bakoitzak eskatzen duen ordu-kopurua ez dakigunez, hasiera batean kopurua hiru eta zortzi artean izatea zehaztu dugu; hala ere, lehenengo probak egitean egunera dezakegu, behar izanez gero.
- Sistemaren eraginkortasuna: sistemaren bertsio berriak ez du exekutatzeko arazorik izan behar eta datu zuzenak adierazi behar ditu.

Adierazle kualitatiboak:

- Kodearen argitasuna: garatutako bertsioaren kodeak garbia izan behar du, beharrezko iruzkinak idatzita ditu, eta aldagaientzat zentzuzko izenak aukeratu behar dira; hau da, sistema ezagutzen ez duen hirugarren pertsona batek zatikako inplementazioa ulertzeko gai izan behar du.
- Lortutako emaitzekin gogobetetzea: alde batetik, sisteman lortutako hobekuntza kontuan hartu behar da. Bestetik, emaitza asko hobetzea lortu ez bada ere, ateratako ondorioak esanguratsuak izan daitezke.
- Lortutako esperientzia: proiektuaren emaitza edozein dela ere, web meatzaritzari buruzko proiektu honekin ikututako arlo guztien inguruan lortutako ezagutza eta ikerketa-esperientzia emaitza onak dira.

Adierazle garrantzitsuak:

- Zuzendariaren oniritzia: proiektua ezin da amaitutzat eman zuzendaria ados ez dagoen arte. Aldi berean, zuzendariaren iritzi positiboa proiektuaren arrakastaren adierazletzat har dezakegu.

Adierazle gehiagarriak:

- Proiektuan lortutako nota: proiektua aurkeztu eta defendatu ondoren epaimahaiak jarritako kalifikazioa esanguratsua da proiektuaren kalitatea neurtzeko garaian.
- Lortutako lanaren eragina zientzia-munduan: proiektua txikia dela jakinik, bistakoa da ez duela zientziaren ezagutza-mundua irauliko, baina baliteke proiektutik ateratako ondorioak beste norbaiten ikerketan laguntzazkoak izatea.

2.7.2 Arriskuak kudeatzeko plana

Proiektu honen tamaina txikia eta betebeharrak urriak badira ere, beste edozein proiektuk bezala, baditu bere arrakasta zalantzan jartzen duten mehatxuak. Hori dela eta, arris-

kuen kudeaketarako plan egoki bat behar da, ezustekoek proiektuaren egikaritzea arriskuan jar ez dezaten, dena aurreikusitako egutegiaren arabera bukatzea ahalbidetuz.

Alde batetik, makinak dauzkagu. Proiektuko lan guztia digitalki gordetzen denez, arazo garrantzitsu bat informazio guztia galtzea izango litzateke. Hori dela eta, egindako lana seguru mantentzeko, *Dropbox* eta *Google Drive* (edo hodeiko beste edozein biltegiratze mota) erabili behar dugu, fitxategien kopiak hodeian automatikoki egiteko. Gainera, *Git*[1] eta *GitHub*[2] erabilita, kodearentzako beste segurtasun-geruza bat gehitzeaz gain, kodearen bertsio desberdinen kontrola eta antolaketa erraz dezakegu. Hala ere, astero eskuzko kopia bat egin behar dugu, disko desberdinetan.

Hala ere, arrisku bakarra ez da hardwarea; askoz kezkagarriagoa den arrisku bat badago: helburuen eta betebeharren eguneraketak. Proiektuaren bizi-zikloan zehar, betebehar berriak agertzen badira, proiektuan izan dezaketen eragina oso txarra izan daiteke. Hori dela eta, arrisku hau minimizatzeko asmoz, hasierako helburuak, betebeharrak eta irismena ahalik eta zehatzen definitu behar dira. Proiektuaren jarraipen-faseak ere erabakigarriak dira. Izan ere, zenbat eta beranduago eguneratzen den dena, orduan eta handiagoa da eragina.

Bestalde, informatikako proiektu guztietan gertatzen den bezala, sistemaren inplementazioa aldatzeko garaian, espero baino denbora gehiago pasa dezakegu arazoren batekin blokeatuta.

Hori gutxi balitz, aldaketak egin arren, baliteke sistemak ondoren ematen dituen emaitzak ez izatea espero ditugunak eta, beraz, ez lortzea nahi ditugun hobekuntzak. Nahiz eta egindako proposamenek ustez hobekuntza batera bideratzen gaituzten, datuek edo sistemaren funtzionamenduak gure usteak zaputz ditzakete.

Gainera, sistemaren bertsio egokiena aukeratu ondoren, gainontzeko inguruneetako (*Discapnet* eta *Gipuzkoa.eus*) datuak prest ez egotea gerta daiteke, hirugarren pertsonen esku baitago.

Jarraian, arriskuak zerrendatuta daude (hurrenkera ez da esanguratsua):

- Hardware arazoak (sistema martxan dagoen makinan edo ordenagailu pertsonalean)
- Lana galtzea (kodea, memoria edota hori idazteko beharrezko materiala)
- Betebeharrak, helburuak edo irismena eguneratzea
- Ezusteko programazio arazoak

[1] *Git* aldaketak kontrolatzeko eta kodea kudeatzeko sistema libre eta dohakoa da.
[2] *GitHub* webean biltegiratzeko *Git*-ek erabiltzen duen *hosting* zerbitzua da.

- Esperotako hobekuntzak ez lortzea
- Ingurune berrietako datuak ez lortzea

2.7.3 Kontingentzia-plana

Proiektuaren bizi-zikloan zehar aurreko atalean aipatutako arazoren bat gertatzen bada, proiektuaren irismena arriskuan jarriz, lehenengo gauza arazoa konpontzen saiatzea da. Hau da, makinaren batek ez badu funtzionatzen, esaterako, beste makina batean martxan jartzea edo programazio arazorik izanez gero, inguruko norbaiti laguntza eskatzea.

Aldiz, arazoa hobekuntzak ez lortzea bada, beste mota bateko probak edo aldaera desberdinak egitea probatu behar dugu, emaitzen jokabidea aztertu ahal izateko batean eta bestean.

Ingurune berrietako datuei dagokienez, hirugarren instituzioen esku dagoenez, gogorarazi besterik ezin dugu egin. Hala ere, ez badizkigute datuak bidaltzen, azken txanpan proiektuaren helburuak aldatu eta sarean bilatutako beste datu batzuekin probatzea hausnar dezakegu. Kasu horretan, bi ingurune horiei buruzko ondorioak aurreragorako utzi beharko dira.

Arazoa edozein dela ere, proiektuaren zuzendariarekin eta laborategiko interesatuekin bilera bat antolatu behar da, arazoari buruz hitz egin eta konponbidea adosteko.

2.8 Proiektuaren hartzaileak edo interesatuak

Proiektuaren interesatuak edo hartzaileak lau mailatan banatu ditugu, interesaren eta inplikazioaren arabera. Interesatu nagusia ni neu naiz, proiektuaren egilea. Proiektua egikaritzearen erantzukizuna nire gain dago. Nire eginbeharra da proiektua aurrera ateratzeko lana egitea, nahiz eta kideen eta zuzendariaren laguntza eta bideratzea eskertuko dudan.

Hurrengo multzoan, *ALDAPA* laborategiko kideak daude, proiektu honen emaitza interesatzen baitzaie. Izan ere, emaitzak erabil ditzakete etorkizunean. Nahiz eta proiektua ez den beren erantzukizuna eta ez duten zertan, laguntzeko prest daude, behar izanez gero.

Hirugarren multzoan, proiektuaren zuzendaria dago: Olatz Arbelaitz. Bere zeregina ikaslea proiektuan zehar bideratzea da. Bi erantzukizun dauzka: proiektuaren inguruko zalantzak argitzea eta gainbegiratzen duen ikasleak proiektua arrakastarekin amaitzen duela ziurtatzea. Behin memoria aurkezten denean, txosten bat aurkeztu behar du, proiektuaren amaierako nota erabakitzerakoan kontuan hartzen dena.

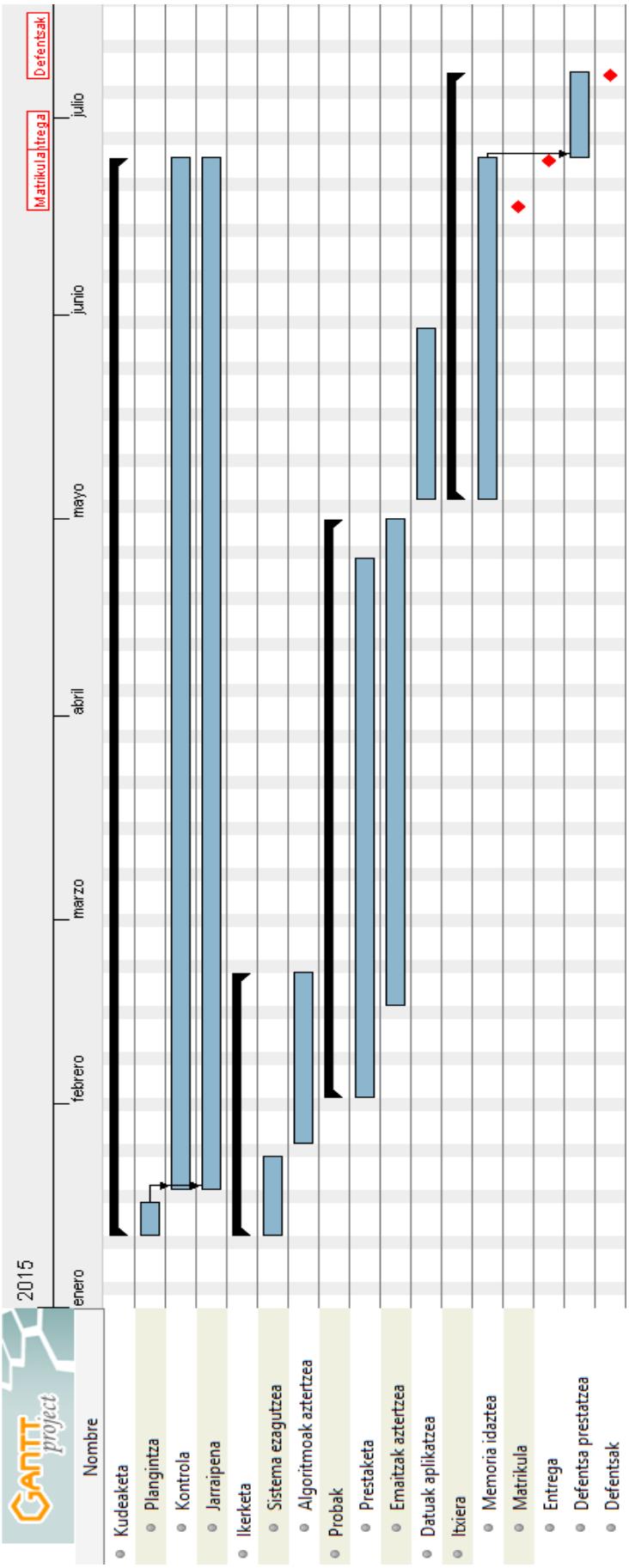
Bukatzeko, proiektua ebaluatzen duen epaimahaia dago. Partaideek Informatika Fakultateko irakasle izan behar dute. Beren lana proiektua aztertzea eta kalifikatzea da, memorian eta defentsan aurkeztu zaien lanean oinarrituta. Epaimahaiko kide horiek Donostiako Informatika Fakultateko batzordeak aukeratzen ditu eta proiektuaren espezialitate berekoak izan behar dute. Kasu honetan, proiektua Konputazio adarrekoa denez, epaileek ere *Konputazio Zientziak eta Adimen Artifiziala* sailekoak izan behar dute; LSIko irakasleren bat ere egon daiteke, baina proiektuaren espezialitatean irakasten duena. Azken interesatu hauek ez dute ikaslearekin inolako harremanik proiektuaren defentsaren egunera arte.

Edonola ere, interesatu gehiago egon daitezke. Proiektuaren gaian interesa daukaten irakasle edo ikasleak, edota ikerlariak. Gainera, proiektuaren defentsa irekia denez, edonor etor liteke proiektuaren aurkezpena entzutera.

Komunikazio-plana

Hiru komunikazio bide nagusi aukeratu ditugu: posta elektronikoa, Skype eta aurrez aurreko komunikazioa.

- Laborategiko kideekin komunikazio gehiena aurrez aurre egitea aurreikusten da, laborategian bertan lan egiteko asmoa baitugu. Hala ere, edozein kasutan, posta elektronikoa erabiltzeko aukera ere badago.
- Olatz Arbelaitzekin, hasiera batean, posta elektronikoz edo, azalpen luzeagoak behar izanez gero, Skype teknologia erabiliz komunikatzea adostu dugu. Proiektuarekin buru-belarri hastean, hau da, proiektuaren zatirik handienean, ordea, aurrez aurre komunikatzeko asmoa daukagu.
- Epaimahaiarekin, defentsaren egunean, aurrez aurre komunikatu behar gara.



Irudia 2.5: Proiektuaren Gantt diagrama.

Proiektuaren kalitatearen egiaztapen-zerrenda	B/E
Proiektuaren egikaritzapena plangintzarekin bat dator	
Orduen estimazioak zehatzak edo onargarriak dira	
Egindako proba kopurua nahikoa da	
Irismena zehatza da eta eguneratuta dago	
Pentsatutako egutegia bete daiteke	
Egindako lana irismenarekin bat dator	
Sistemaren errendimendua irismenarekin bat dator	
Proiektuaren arriskuak kontrolatuta daude	
Ez dago produktuari edo proiektuari eragin diezaioketen aldagai berririk. Baiezko kasuan, dokumentatuta dago	
Interesatuak behar bezala informatuta daude	
Proiektuaren azken segurtasun-kopia azken astean egin da	

Produktuaren kalitatearen egiaztapen-zerrenda	B/E
Kodea ulergarria da	
Sistemak behar bezala funtzionatzen du	
Inplementatutako aldaketak irismenean definitutakoarekin bat datoz	

Irudia 2.6: Jarraipena eta kontrola egiteko egiaztapen-zerrenda.

3. kapitulua

Jatorrizko sistema

Kapitulu honetan, proiektuan hobetu beharreko jatorrizko sistemaren xehetasunak eta proiektuaren hasiera azaltzen ditugu. Atal desberdinetan banatuta, jatorrizko sistemaren deskribapena, erabiltzen diren ebaluazio-neurriak, lehenengo probak eta ondorioak aurki daitezke.

3.1 Testuingurua

Jatorrizko sistema turismoaren testuinguruan eraikitako web meatzaritzarako sistema da. Hain zuzen ere, Bidasoa Turismoko webgunearen gainean egiten du web meatzaritza. Sistema hau eraikitzeko arrazoi nagusia turismoaren industriak azken urteetan, bidaiariak sarean ibiltzen hasi direnetik, jasan duen aldaketa ikaragarria da. Horregatik, sistema adimentsuen erabilera nahitaezkoa bihurtu da. Informazio-sistema horiek bezero eta zerbitzariei informaziorik garrantzitsuena, erabakitzeko laguntza, mugimendu erraztasuna eta bidaia-esperientziarik onena eskaini behar dizkiete. Hori dela eta, Helmugako Marketing Erakundeek, (DMO, *Destination Marketing Organization*) teknologia berriak erabiltzeaz gain, teknika berri hauen informazioa interpretatzen eta erabiltzen ere ikasi behar dute.

3.2 Jatorrizko sistemaren deskribapena

ALDAPA taldearen sistema web meatzaritzako sistema orokor eta ez-inbasiboa da [Arbelaitz, 2013], web zerbitzari batean bildutako informazio minimotik eta Bidasoa Turismo webgunearen (BTw) erabileratik abiatuta sortu zena. Sistemak webaren erabilpena eta edukia aztertzeko teknikak konbinatzen ditu eta honako hiru helburuak dauzka:

- Erabiltzaileen nabigazio-profilak sortzea, gero, profil horiek esteken aurreikuspenean erabiltzeko.
- Profilak informazio semantikoarekin aberastea, profil horiek dibertsifikatzeko. Tresna honen bitartez, DMOek erabiltzailearen gustoekin eta beren interesekin bat egiten duten estekak proposatzeko aukera daukate.
- Hizkuntzaren araberako erabiltzaileen interes-profilak lortzea, DMOei etorkizuneko weben diseinurako informazio garrantzitsua emateko, baita helburu zehatzeko marketing kanpainak egiteko ere.

Proiektu honetan, abiapuntua eta hobetzen saiatuko garena lehenengo atala da, hots, nabigazio-profilak sortzeaz arduratzen den atala.

Sistemak arrakasta izan du, lortu dituen profilek neurri batean benetako erabiltzaileen nabigatze-sekuentziekin bat egiten baitute, eta neurri handiagoan, erabiltzaileen interesekin. Gainera, sistemak automatikoki ateratako webaren egitura-semantikoa eta interes--profilak oso erabilgarriak direla baieztatu du BTwko DMO lan-taldeak.

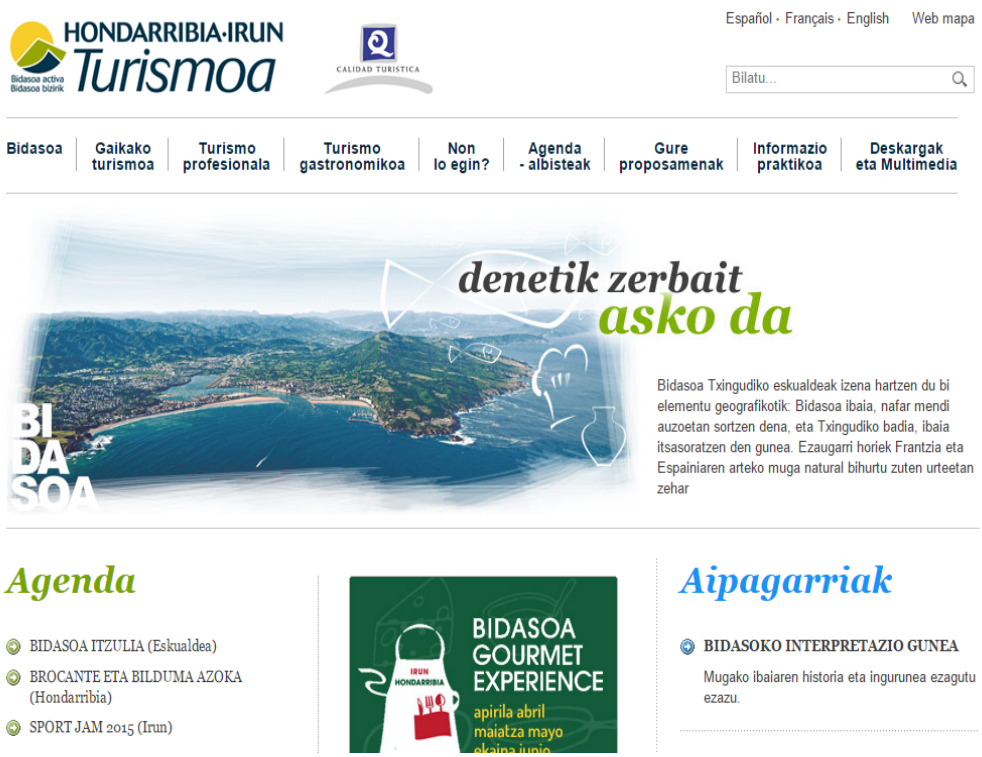
3.1 irudian ikus daiteke Bidasoa Turismo webgunearen orri nagusia eta 3.2 irudian, jatorrizko sistematik landuko dugun atalaren eskema.

3.2.1 Datuen prestaketa

Lehenengo, sistemak jaso beharreko datuak prestatu behar dira. Guk jasotakoak Bidasoa Turismo webgunearen erabiltzaileen eta horren web zerbitzariaren arteko informazio--truketik sortuak dira, 10 hilabetetan zehar. Datu horiek *log* fitxategietan gordetzen dira. Fitxategi horiek formatu estandarra daukate, hortaz, datuak gordetzeko eremuak zein diren finkatuta dago. Hala ere, aurreprozesaketan, sistemak behar dituen eremuak bakarrik gordetzen dira: eskaera egiten duen erabiltzailearen IP helbidea, eskaera noiz egin den, eskatutako URLa eta eskaeraren egoera.

Hasiera batean 3.636.233 eskaera bazeuden ere, aurreprozesaketaren ondoren kopuru hori murriztu egin da. Horretarako, eskaera akastunak eta administrazio-lanetako URL-ak kendu dira, baita nabigatzailean webguneko irudiak, bideoak edo antzekoak kargatzeko egiten diren eskaerak ere.

Gainera, IP helbidearen eta eskaeren artean pasatako denbora-tartearen bitartez, erabiltzaile-saioak identifikatu dira. Hortik, ezer egin gabe 10 minutu baino gehiago pasa dituzten saioak bukatutzat eman dira. Bestalde, 3 klik baino gutxiago eta 86 baino gehiago zituztenak, hau da, %98-ko ehunekotik kanpo geratzen direnak, ezabatu dira



Irudia 3.1: Bidasoa Turismo webgunearen orri nagusia.

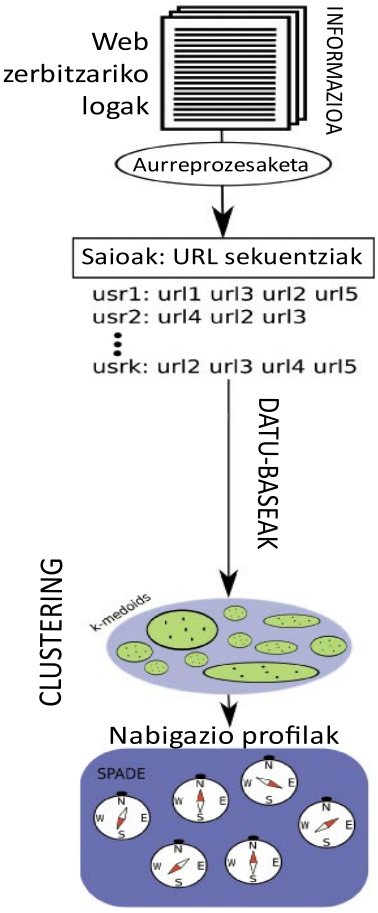
(gehiagokoak robotek sortuak izan daitezkeelakoan). Bukatzeko, kultura-egitarauari edo berriei dagozkien klikak baztertu dira, horrelako esteken edukia aldakorra delako.

Bestalde, saio gutxitan agertzen diren URL-ak kendu eta URL berean egindako ondoz ondoko klik-ak bakarrean elkartu dira.

Aurreprozesaketa bukatu ondoren, beraz, saio bakoitza adierazten duen URL sekuentzia ordenatuak dituen datu-basea lortu da; adierazpen hau hautatu dugu URL-ak bisitatu diren ordenari garrantzia ematen zaiolako. Datu-basean, guztira, 10790 kasu daude eta bakoitzak, batezbestean, 10 URL ditu.

3.2.2 Profilak lortzea

Tresna komertzialek analisi estatistikoa egiten badute ere, ikasketa automatikoak informazio gehiago atera dezake analisi horretatik. *ALDAPA* taldeak diseinatutako sistemaren kasuan, PAM (*Partitioning Around Medoids*) [Kaufman, 1990] algoritmoa erabiltzen da *clustering*-a egiteko, sekuentziak konparatzeko *Edit Distance* erabiliz, sekuentziekin lanean ari garenez, ezin baita bi punturen arteko distantzia bektoreak konparatuz kalku-



Irudia 3.2: Jatorrizko sistematik profilak lortzeko sistemaren arkitekturaren eskema.

latu; horren ordez, sekuentziak konparatzen dituzten distantziak erabili behar ditugu. Antzeko nabigazio-patroia duten erabiltzaileak multzokatzea da helburua.

Datu-base horretatik, *cluster* edo multzo desberdinak sortzen dira, PAM algoritmoa erabiliz. Horretarako, PAM algoritmoari k parametroa (zenbat *cluster* sortu behar dituen) pasa behar zaio. Parametro horrek datu-basearen egiturarekin eta sortutako profilen zehaztasunarekin du zerikusia. Zenbat eta altuagoa izan k, orduan eta zehatzagoak dira profilak.

Aurrera jarraitu baino lehen, ikus dezagun sakonago zer den PAM algoritmoa eta nola funtzionatzen duen *Edit Distance*-ek.

PAM algoritmoa (*K-medoids*)

K-medoids multzokatze-teknika klasikoa da; datu-baseko n objektuak k multzo edo *cluster*-etan banatzen ditu, k aurretik finkatuta dagoelarik. *K-means* algoritmoarekin lotua dago. Izan ere, biak algoritmo partizionalak dira, hau da, datuak multzokatzen di-

tuzte, eta biak saiatzen dira puntu bat eta bere *cluster*-eko erdigunearen arteko distantzia minimizatzen. *K-means* algoritmoak ez bezala, ordea, *K-medoids*-ek datupuntuak, hots, multzoko objektuak, aukeratzen ditu erdigunetzat (medoideak) eta datupuntuen arteko distantzien matrizearekin egiten du lan.

Medoidea *cluster* bateko objektua da, beraz, *cluster*-eko objektu guztiekin batezbesteko distantzia minimoa daukana. Horiekin lan egiteko algoritmo ezagunena da *Partitioning Around Medoids* (PAM). Honela funtzionatzen du:

Algoritmoa 1 PAM()

1: Hasieratu: ausaz, n objektuetatik k aukeratu, medoide moduan hasieratzeko, ordezkapenik egin gabe
2: Datu-baseko objektu bakoitza hurbileneko medoidearekin elkartu
3: **for all** $medoide \in medoideak$ **do**
4: **for all** $objektu \notin medoideak$ **do**
5: *medoide* eta *objektu* trukatu eta kalkulatu konfigurazio-kostu totala
6: **end for**
7: **end for**
8: Kostu baxuena duen konfigurazioa aukeratu
9: Errepikatu 2-8 pausoak aldaketarik ez dagoen arte

Algoritmoko aldagaiei dagokienez, *medoideak* multzoan momentuko k medoideak biltzen dira. *medoide* aldagaian, banaka, *medoideak* multzoko osagaiak pasatzen ditugu; modu berean, *objektu* aldagaian, pausoka, datu-baseko osagaiak jasotzen ditugu, *medoideak* multzoan ez daudenak. Bestalde, konfigurazio-kostua *cluster* bakoitzeko elementuek dagokien medoidera duten distantzien baturari deritzo.

Badakigu gure proiektuan objektuak URL sekuentziak direla. Beraz, sekuentzia horietako bat hartzen dugu medoidetzat. Algoritmoaren 2. puntuan, datu-baseko objektu bakoitza medoide hurbilenarekin elkartzeko, ordea, distantzia neurtzen duen metrika bat behar dugu eta ez digu edozein metrikak balio, bi sekuentzien arteko distantziak kalkulatu behar ditugulako. Horretarako, *Edit* [Gusfield, 1997] [Chordia, 2011] distantzia da erabiliena. Ikus dezagun zertan oinarritzen den.

Edit edo Levenshtein distantzia

Edit distantzia bi karaktere-kateren arteko desberdintasuna neurtzeko metodoa da. Horretarako, karaktere-kate bat bestea bihurtzeko behar diren eragiketa kopuru minimoa zenbatzen du. Aldaera desberdinak badaude ere, guk Levenshtein-en distantzia erabiltzen dugu:

- **Txertaketa**: karaktere bat gehitzea. $a = uv$ bada, x karakterea txertatuta, $a = uxv$ lortzen dugu. Hau da, $\varepsilon \rightarrow x$, ε testu hutsa delarik.
- **Ezabaketa**: karaktere bat ezabatzean *uxv uv* bihurtzen da ($x \rightarrow \varepsilon$).
- **Ordezkapena**: x karakterea $y \neq x$ karakterearekin ordezkatzean, *uxv uyv* bihurtzen da ($x \rightarrow y$).

Levenshtein-en jatorrizko definizioan, eragiketa bakoitzak unitateko kostua dauka (karaktere bat bere buruarekin ordezkatzeak, zero), beraz, Levenshtein-en distantzia a sekuentzia b sekuentzia bihurtzeko behar den eragiketa-kopuru minimoa da. Adibidez, "datu" eta "katea" arteko Levenshtein distantzia hiru da. Ikus dezagun:

1. datu $\rightarrow$ **k**atu
2. katu $\rightarrow$ kat**e**
3. kate $\rightarrow$ kate**a**

Clustering prozesuak antzeko portaera duten erabiltzaileen sekuentzien multzoak ematen dizkigu. Multzo horiek lortuta, bakoitzarentzako ereduak izango diren URLak bilatzen ditu sistemak. Horretarako, SPADE [Zaki, 2001] algoritmoa erabiltzen da. Algoritmo horri esker, *cluster* bakoitzeko klik sekuentzia komunak bilatzen dira, hots, bertako saioek maizen bisitatu ohi dituzten URL sekuentziak.

Ikus dezagun, bada, nola funtzionatzen duen SPADE-k.

SPADE algoritmoa

Frequent Sequence Mining algoritmoa da SPADE, ordenatuta dauden objektuen artean patroiak aurkitzeko erabiltzen dena, hain zuzen ere. Honela funtzionatzen du *Frequent Sequence Mining*-ek:

Sekuentzia bat, α, $< a_1, a_2, ..., a_m >$ gertaeren zerrenda ordenatua da. Gertaera bat ordenatu gabeko elementuen multzo ez-hutsa da, $a_i \subseteq i_1, i_2, ..., i_k$. $\alpha =< a_1, a_2, ..., a_m >$ sekuentzia bat $\beta =< b_1, b_2, ..., b_n >$ sekuentziaren azpisekuentzia da baldin eta soilik baldin $i_1, i_2, ..., i_m$ existitzen bada, non $1 \leq i_1 < i_2 < ... < i_m \leq n$ eta $a_1 \subseteq bi_1$, $a_2 \subseteq bi_2$ eta $a_m \subseteq bi_m$. Sekuentzien datu-base bat emanik, $D = s_1, s_2, ..., s_n$, α sekuentziaren euskarria (*support*-a) α azpisekuentziatzat duten D-ko sekuentzia kopurua da, hau da, sekuentzia hori datu-baseko zenbat sekuentziatan agertzen den azpisekuentzia moduan. α-ren *support*-a *minimum support* (*minsup*) edo euskarri minimoa deitzen den parametroa baino altuagoa bada, orduan α ohiko sekuentzia da, maiz agertzen den sekuentzia.

Ohiko sekuentziak edo maiz agertzen direnak bilatzeko algoritmo erabilienetako bat SPADE (*Sequential PAttern Discovery using Equivalence classes*) da. Datu-basearen formatuko identifikatzaile zerrenda (*idZerrenda*) bertikala erabiltzen du. Beste modu batera esanda, datu-baseko sekuentzia bakoitzari zerrenda bat esleitzen zaio, datu-baseko beste zein sekuentzietan agertzen den gordetzeko. Orduan, erraz jakin daiteke sekuentzia bakoitza zenbatetan agertzen den, eta, ondorioz, ohiko sekuentziak zein diren.

SPADE algoritmoaren lehen pausoa 1-sekuentziak, zeinak elementu bakarreko sekuentziak diren, kalkulatzea da. Bigarren pausoan, 2-sekuentziak (bi elementuko sekuentziak) aztertzen dira. Hori lortzeko, errepresentazio bertikala horizontal bihurtu behar da memorian, eta elementu pare bakoitzeko sekuentzia kopurua kontatzen da, bi dimentsioko matrizea erabiliz.

n-sekuentziak (n-1)-sekuentziak elkartuz osa daitezke, *idZerrenda* erabiliz. *idZerrenda*ren tamainaren bitartez, sekuentzia jakin hori datu-basean zenbatetan agertzen den jakin dezakegu. Errepikapen-kopurua *minsup* baino handiagoa bada, sekuentzia ohikoa da. Ohiko sekuentzia gehiago bilatzen ez dituenean, algoritmoa gelditu egiten da.

Algoritmoak zabalerako edo sakonerako bilaketa egin dezake.

PAM-ekin gertatzen den moduan, SPADE-k ere zenbait parametro doitzea eskatzen du, ikusi dugunez. Gure proiektuan zehar, balio hauekin lan egiten du gure sistemak:

- *minsup* edo sekuentzia bat gutxienez zenbat aldiz agertu behar den onargarria izateko: 0.2
- Proposatu beharreko URL kopurua: 4

minsup aldagaiak baldintzatzen du *cluster* bakoitzeko erabiltzaile mota. Hortik, proposatu beharreko URL kopurua lau izatea jo da egokitzat.

Puntu honetara helduta, 3.2 irudian ikusi dugun moduan, erabiltzaile-saioak edo URL sekuentziak dituen datu-basea izanik, PAM algoritmoa aplikatuz, sekuentziak edo saioak *cluster* desberdinetan multzokatzen dira. Ondoren, *cluster* horiekin SPADE algoritmoa jartzen da martxan multzo bakoitzeko profilak lortzeko, eta, bukatzeko, profil bakoitzeko URL erabilienak kontuan hartuz, zein URL proposatu erabakitzen da.

3.2.3 Sistemaren erabilera

Profil antzekoena bilatzeko, aztertu beharreko sekuentziaren ehuneko jakin bat hartzen da (proiektuan zehar %25 eta %50). Zatiaren luzera, hortaz, sekuentziaren beraren luzeraren arabera aldatzen da. Horrela, webgune batean gertatuko litzatekeen egoera erreala simulatzen da. Sekuentzia berrien lehen klikak erabiltzaile berrien nabigazioak bailiran hartu eta sistemak proposatuko liekena eta benetan egindako nabigazioa konparatzen dira.

Sekuentziaren zati hori aztertuta, beraz, erabiltzaile berriari profil bat esleitzen zaio K-NN [Dasarathy, 1991] sailkatzailea erabiliz. Sailkatzaileak, profil (*cluster*) antzekoena bilatzeko, sekuentziatik profiletara dagoen distantzia hartzen du kontuan. Jarraian, profil horretatik jasotzen dituen 4 URLak proposatzen ditu. Kasuren batean, hurbileneko profilean ez badaude 4 URL onargarri proposatzeko, hurrengo profil antzekoenera jotzen du, eta, horrela, harik eta 4 URL desberdin lortzen dituen arte edo profilak bukatzen zaizkion arte.

3.2.4 Ebaluazioa

Lortutako profilak ebaluatzeko, *cross validation* edo balidazio gurutzatua erabiltzen da. Datu-basea hamar geruzatan banatzen da. Horietatik zazpi ikasteko erabiltzen dira; bi, balidazioa egiteko, eta, azkena, azterketa egiteko. Modu horretan, ikasteko datuekin, profilak sortzen dira, eta balidazioko eta aztertzeko datuekin, sortutako profilak ebaluatzen dira.

Sekuentziek, batezbestean, 10 URL dauzkate, honela adieraziak: URL1, URL2, URL3, URL4, URL5, URL6, URL7, URL8, URL9, URL10. Hortaz, erabiltzailea ezagutzeko sekuentzia horren %25a (URL1, URL2) edo %50 (URL1, URL2, URL3, URL4, URL5) hartuta, proposatutako estekak gainerako sekuentziaren zatiarekin konparatzen dira, %25aren kasuan, URL3, URL4, URL5, URL6, URL7, URL8, URL9, URL10 sekuentzia zatiarekin edo %50aren kasuan, URL6, URL7, URL8, URL9, URL10 zatiarekin. Eredu osoa kontuan hartzen den kasuetan, erabiltzailea ezagutzeko, aurreko arazoan azaldutako zati berak hartzen dira, baina konparatzeko garaian, URL sekuentzia osoa erabiltzen da.

Sistemaren prozesuarekin bukatzeko, sekuentzien konparazio horretatik ateratako emaitzetan oinarrituz, ondorioak ateratzeko zenbait ebaluazio-neurri kalkulatzen dira: *precision*, *recall* eta *f-measure*. Hurrengo atalean ikusiko ditugu ebaluazioaren xehetasun gehiago.

3.3 Ebaluazio-neurriak

Sistemaren emaitzak eta errendimendua hobeto ulertu ahal izateko, beharrezkoa da, lehenengo, kontingentzia-taula zer den ulertzea.

	Zuzenak	**Okerrak**
Aukeratuak	TP (*True Positive*)	FP (*False Positive*)
Aukeratu gabeak	FN (*False Negative*)	TN (*True Negative*)

Taula 3.1: Kontingentzia-taula.

Proiektu honen ingurunean, zuzenak erabilitako URLak dira eta okerrak, berriz, erabili ez direnak. Bestalde, aukeratuak proposatu diren URLak dira eta aukeratu gabeak, aldiz, proposatu ez direnak. Beraz, *True Positive* balioa lortzeko, erabilitako URLetatik zenbat proposatu diren kalkulatu behar da, edo, beste modu batera esanda, proposatu diren guztietatik zuzenak zenbat diren. FP balioak proposatu diren URLetatik zein ez diren erabili adierazten du. Proposatu ez diren URLei dagokienez, horietatik zenbat erabili diren FN balioak neurtzen du, eta, zenbat ez diren erabili, TN balioak. Balio hauetan oinarrituz, sistemaren funtzionamenduaren berri ematen duten neurriak kalkulatu ditugu.

Jar dezagun adibide bat. Demagun webgune bateko hitzen artetik oinetakoen markak atera nahi ditugula. Webgunean dauden 100.000 hitzetatik 10 markak dira eta gainontzekoak, 99.990, ez. 3.2 taulan ikus dezakegu adibideari dagokion kontingentzia-taula.

	Zuzenak (10)	**Okerrak (99990)**
Aukeratuak (0)	TP (0)	FP (0)
Aukeratu gabeak (100000)	FN (10)	TN (99990)

Taula 3.2: Kontingentzia-taularen adibidea.

Hortaz, sistemak hitz guztiak marka ez diren hitzak bailiran sailkatzen baditu, honako asmatze-tasa lortzen da:

$$\text{Asmatze-tasa} = \frac{TP+TN}{TP+FP+FN+TN} = \frac{0+99990}{0+0+10+99990} = \frac{99990}{100000} = \%99.99$$

Bistan denez, ebaluazio-neurri hori ez da oso egokia, ez baitu sistemaren funtzionamendua behar bezala epaitzen. Hori dela eta, beste neurri batzuk azaltzen ditugu jarraian; gure sisteman erabiltzen direnak, hain zuzen ere. Neurri hauei esker, sailkatzaileak kasuak zuzen sailkatzeko duen gaitasuna neurtzen da. Emandako azalpenak aztertzen ari garen testuinguruan kokatuta daude.

- ***Precision (P)*:** Aukeratu diren URLetatik benetan erabili direnen ehunekoa.
$$\frac{TP}{TP+FP}$$
Aurreko adibidearen kasuan, $TP = 0$ denez, *precision*-en balioa ere zero da.
$$\frac{0}{0+0} = 0$$

- ***Recall (R)*:** Erabilitako URLen artetik zuzen proposatu direnen ehunekoa (*sensitivity*).
$$\frac{TP}{TP+FN}$$
Oraingoan ere, arrazoi beragatik, *recall*-aren balioa zero da.
$$\frac{0}{0+10} = 0$$

- ***F-measure*:** *Precision* eta *recall* erlazionatzeko neurri konbinatua.
$$\frac{(\beta^2+1)PR}{\beta^2 P+R}$$
Oro har, β-ren balioa zero dela kontuan hartuz, honakoa da *f-measure*-ren balioa:
$$\frac{(1^2+1)*0*0}{1^2*0+0} = 0$$

Hala ere, gure sisteman $\beta = 0.5$ da, *precision*-i indar handiagoa eman nahi diogu eta. Izan ere, URL asko proposatuta erraza da $recall$-aren balioa handiagotzea, baina *precision* jaisten da, eta, ondorioz, ez da erabiltzailearentzat baliagarria, ezin baititu hainbeste proposamen kontuan hartu. Modu berean, URL gutxiago proposatuta *precision* (zehaztasuna) handiagotzen da eta *recall*, txikiagotu.

3.4 Probak

Sistemaren helburua erabiltzaileari interesa dakizkiokeen URL-ak proposatzea denez, egin ditugun probak horretara bideratuak daude. Hori dela eta, aurreko atalean azaldu dugun moduan, ezin diogu sistemari nahi adina proposamen egiten utzi, bestela emaitzak hobe ditzakeelako *recall*-a handituz, baina proposamenak baliagarriak izan gabe. Bestalde, datuek PAM-en k parametroan eta k parametroak datuetan duten eragina ere aztertu nahi dugu. Azken finean, proiektuan zehar egin ditugun proba guztien helburua hobekuntzak lortzea da.

Jarraian, sistema ezagutzen eta emaitzak interpretatzen hasteko, ikus dezagun jatorrizko sistema BTw-ko datu-basearekin martxan jartzeko sistemaren parametroak zein diren (aurreko ataletan aipatu ditugu):

- *Minsup* edo URLa onargarria izateko gutxieneko *support*-a: 0.2

- F-measure kalkulatzeko β: 0.5 (*Precision*-i indar gehiago emateko)
- Balidazio gurutzatua: 10 geruza (Ikasteko, 7; balidaziorako, 2, eta testerako, 1)

Balidazio gurutzatua balio horiekin eginda, datu-baseko 10790 sekuentzien banaketa 3.3 taulan ikus daiteke.

	Klik kopurua	**Sekuentzia kopurua**
Ikasteko	75158.3	7553
Balidaziorako	21473.8	2158
Testerako	10736.9	1079

Taula 3.3: Balidazio gurutzatuaren banaketa.

Bestalde, PAM algoritmoak ere k parametroa finkatzea eskatzen du. Ez dakigunez zein k baliorekin funtzionatuko duen hobekien, balio desberdinak probatzea erabaki dugu. Balioak aukeratzeko ohiko irizpidea $\sqrt{n}$-rainoko balioak aztertzea da (n objektu edo sekuentzia kopurua da). Hala ere, guk azterketa zabalagoa egin nahi izan dugu eta, hortaz, bai balio txikiak eta bai altuak probatu ditugu. Honakoak dira PAM algoritmoari eskatzen dizkiogun *cluster* kopuruak: 50, 100, 150, 200, 300, 400, 500.

Emaitzak ikusteko garaian, kasu honetan neurri guztiak erakutsi baditugu ere, oro har *f-measure* bakarrik aztertzen dugu. Gainera, jarraian lau taula dauzkagu ikusgai. Alde batetik, balidazioaren eta testaren emaitzak erakusten ditugulako eta, bestetik, emaitzak sekuentziaren tamaina desberdinak ezagutuz kalkulatzen ditugulako. Gure probetan, aurretik aipatu dugun bezala, sekuentziaren %25 eta %50 aztertuz proposatzen ditugu estekak.

Azpian erakusten ditugun taulen lehenengo gelaxkan, emaitzak zein algoritmorekin inplementatutako sistemarenak diren ikus daiteke. Lehenengo zutabean, *cluster* kopurua azaltzen da. Hurrengo hiruetan (berdea, sekuentziaren %25 aztertzen ari bagara eta, urdina, %50 bada), aurreikuspen-sistemaren *precision*, *recall* eta *f-measure* daude. Azken hiruetan, gorrietan, neurri berdinak erakusten dira, baina eredu osoa kontuan hartuta neurtuz.

Balidazio gurutzatua behar bezala egiten dugu eredu jatorra jarraitu nahi dugulako, baina proiektu osoan zehar aztertzen diren emaitzak balidazioarenak dira, aurkakoa adierazten ez bada. Izan ere, proiektuaren helburua sistemaren emaitzak hobetzea da eta hori da balidazio prozesuaren helburua, hain zuzen ere. Hala ere, azken sistemaren kasuan, testarekin lortutako emaitzak ere erakusten ditugu.

Bestalde, aipatu beharrekoa da oro har ez ditugula balidazio gurutzatuko exekuzio bakoitzaren emaitzak erakusten, *cluster*-kopuru bakoitzeko 10 exekuzioen

batezbestekoak baizik, alferrikakoa iruditzen baitzaigu hainbeste datu jartzea, baina adierazgarriak diren kasuetan ikusgai daude. Edonola ere, datu guztiak gordeta daude, beraz, norbaiti interesatzen bazaizkio, eska ditzake.

Lehenengo, ikus ditzagun sekuentziaren %25a aztertuz balidazioan lortzen diren emaitzak 3.3 irudiko taulan:

PAM						
Cluster kopurua	Precision %25	Recall %25	F-measure %25	MPrecision %25	MRecall %25	MF-measure %25
50	**0.24499539**	**0.2351073**	**0.22991998**	**0.34354728**	**0.30298418**	**0.31587023**
100	0.23761585	0.22470951	0.22214594	0.3359592	0.29350916	0.3076754
150	0.23941147	0.22711086	0.22404571	0.33809084	0.2959954	0.3098806
200	0.24072056	0.22831412	0.22535339	0.339678	0.29708213	0.31139687
300	0.23206672	0.21748753	0.21639541	0.3297498	0.28691357	0.30144376
400	0.22656396	0.21116653	0.21064845	0.32358664	0.2811579	0.2951656
500	0.22724745	0.21232665	0.21142069	0.32444394	0.28193983	0.29593846

Irudia 3.3: Balidazioaren emaitzak sekuentziaren %25 ezagututa.

Precision balioak adierazten duen moduan, proposatu diren URLetatik %24a erabili dira, *cluster*-kopurua 50 den kasuan. *Recall*-aren kasuan, berriz, erabili diren URLen artetik %23a proposatuak izan dira. Gu, hala ere, *f-measure* balioan zentratzen gara. Hortaz, ikus dezakegunez, emaitzarik hoberenak *cluster* kopuru txikienarekin, 50ekin, lortzen dira, bai aurreikuspenean eta bai ereduan.

PAM						
Cluster kopurua	Precision %25	Recall %25	F-measure %25	MPrecision %25	MRecall %25	MF-measure %25
50	**0.24409175**	**0.234031**	**0.22886083**	**0.34251618**	**0.30191773**	**0.31468096**
100	0.23644575	0.22504687	0.22120383	0.33477756	0.2937173	0.30686373
150	0.23716402	0.22656178	0.2220982	0.33549586	0.29537106	0.30784494
200	0.24091749	0.23062465	0.225823	0.3395505	0.29878932	0.311774
300	0.23280816	0.2200929	0.21743496	0.3306302	0.28910148	0.30260143
400	0.22764134	0.21410517	0.21209511	0.32446712	0.2834149	0.29654506
500	0.22671457	0.21334478	0.21126369	0.32402688	0.2828032	0.29604563

Irudia 3.4: Testaren emaitzak sekuentziaren %25 ezagututa.

Testaren emaitzetan (ikus 3.4 irudia) ikusten dugunez, ondorioa bera da.

Jarraian, 3.5 eta 3.6 irudietan, sekuentziaren erdia ezagutuz lortzen ditugun emaitzak ageri dira, balidaziokoak eta testekoak, ondoz ondo:

PAM						
Cluster kopurua	Precision %50	Recall %50	F-measure %50	MPrecision %50	MRecall %50	MF-measure %50
50	**0.18054913**	**0.2398551**	**0.18031698**	**0.33181185**	**0.29229423**	**0.3044352**
100	0.17978452	0.2367163	0.17923039	0.3307113	0.28800717	0.30237308
150	0.17871872	0.23609082	0.17830396	0.32917053	0.28780308	0.3013638
200	0.18025948	0.23750082	0.17976962	0.33095458	0.28926262	0.30307335
300	0.17472197	0.22980992	0.17409018	0.3227062	0.28150535	0.29504174
400	0.1710496	0.22477038	0.17020498	0.3175973	0.2759383	0.28957647
500	0.16921918	0.22292161	0.16842486	0.31497914	0.27398297	0.28714472

Irudia 3.5: Balidazioaren emaitzak sekuentziaren %50 ezagututa.

PAM						
Cluster kopurua	Precision %50	Recall %50	F-measure %50	MPrecision %50	MRecall %50	MF-measure %50
50	**0.17903152**	**0.23857717**	**0.17882074**	**0.32831326**	**0.2903695**	**0.30121842**
100	0.17787305	0.23570481	0.17766345	0.32740963	0.2876335	0.2999147
150	0.17578776	0.23371534	0.17571619	0.32509264	0.28665584	0.29810905
200	0.17863762	0.23741376	0.1785405	0.32912415	0.2899794	0.30191368
300	0.17293791	0.22984931	0.1727228	0.3207831	0.2820376	0.29370543
400	0.17025024	0.22575843	0.16981456	0.31735402	0.27761346	0.2898272
500	0.1678406	0.22262426	0.16732085	0.31522244	0.2750147	0.2875944

Irudia 3.6: Testaren emaitzak sekuentziaren %50 ezagututa.

Oraingoan ere, emaitzarik hoberenak k=50 denean lortzen dira. Gainera, aurreikuspeneko emaitzak askoz kaxkarragoak dira, izan ere, esteka gehiago hartu ditugunez kontuan, klik gutxiago geratzen zaizkio erabiltzeaileari eta, hortaz, zailagoa da horiek zehazki zein diren asmatzea.

Hala ere, ez ditugu emaitzak k-ren arabera bakarrik aztertu. 3.7 irudiko taulan, aurreko proba bereko balidazioaren k-ren bi balioetarako azterketak begiratzen ditugu. Lehenik, k=50 den kasuan, balidazio gurutzatuko 10 exekuzioetako (*run*) bakoitzean lortzen diren emaitzak ikusten dira, eta, azpian, bera, baina k=300 den kasurako. Kasu guztiak aztertu baditugu ere, zentzugabea iruditu zaigu datu guztiak hemen erakustea eta, ondorioz, adierazgarriak direlako adibidetzat aukeratu ditugun bi multzokatzeak bakarrik daude ikusgai taulan.

Zehazki ebaluazio-neurri jakin bat begiratu behar izan gabe, argi ikus dezakegu, k-ren balioen arabera, emaitza hoberenak lortzen dituen exekuzioa aldatu egiten dela. Hau da, datu-multzoak modu desberdinean eragiten dio sistemari sortzen duen *cluster*-

PAM						
Cluster kopurua	Precission %25	Recall %25	F-measure %25	MPrecission %25	MRecall %25	MF-measure %25
50						
run_0	0.2673772	0.25512582	0.25178576	0.3781279	0.33223498	0.34907275
run_1	0.22891566	0.2200658	0.21475345	0.32692307	0.28870934	0.29998863
run_2	0.24791473	0.24496429	0.23479575	0.3400139	0.3088918	0.31604183
run_3	0.26367006	0.25978556	0.24836446	0.3598239	0.32411754	0.3331546
run_4	0.26193234	0.2520987	0.24673386	0.3667748	0.31926903	0.3367046
run_5	0.2299583	0.2220132	0.21554923	0.32182577	0.2853406	0.29541487
run_6	0.24791473	0.23312601	0.23060673	0.34476367	0.2996551	0.31496295
run_7	0.19972196	0.17757587	0.18323354	0.29286376	0.2506455	0.26481092
run_8	0.22381835	0.21730712	0.21070991	0.3221733	0.2854105	0.29567173
run_9	**0.2787303**	**0.26901063**	**0.2626671**	**0.38218257**	**0.3355676**	**0.35287967**
300						
run_0	0.25463393	0.24191757	0.23955162	0.3480074	0.30425334	0.31997427
run_1	0.24721965	0.2539815	0.23582098	0.33873957	**0.31857294**	0.31736305
run_2	0.24026877	0.24021031	0.22634223	0.34151992	0.308962	0.31544602
run_3	0.21895273	0.19607995	0.20130782	0.31811863	0.26160765	0.28478116
run_4	0.22961076	0.21699375	0.21268907	0.31788692	0.28333235	0.29073015
run_5	0.24860983	0.24371745	0.23561084	0.35750696	0.31615028	0.33049154
run_6	**0.2673772**	**0.24775027**	**0.2501691**	**0.365848**	0.31804383	**0.33606863**
run_7	0.20759962	0.19032477	0.19237812	0.29958296	0.25758955	0.27327183
run_8	0.19578314	0.1744596	0.17864317	0.2951807	0.25063807	0.26471987
run_9	0.21802595	0.19549367	0.20183662	0.32391104	0.27186492	0.2931679

Irudia 3.7: Balidazio gurutzatuko exekuzio bakoitzaren emaitzak.

-kopuruaren arabera. Izan ere, 50 multzo egiterakoan, exekuzio hoberena, edozein neurrirentzat, azkenekoa da (*run_9*). Aldiz, 300 *cluster* sortzean, *run_6* edo 7. exekuzioa da hoberena. Nolanahi ere, $k = 50$-eko exekuzioek beti hobeak izateko joera daukate, aurrerago argiago ikusten den moduan.

Gainera, datuek *clustering*-ean duten eragina aztertzeko asmoz, balidazio gurutzatuko exekuzio bakoitzarentzat emaitza hoberenak lortu dituen k zein den aztertu dugu. 3.8 taulan, lehenengo zutabean zein exekuzio aztertzen ari garen adierazten da. Zutabe berdean, sekuentziaren %25a ezagutuz lortzen diren emaitzak ageri dira (ezkerreko zatian k-ren balioa eta eskuinean, lortutako emaitza). Berdina zutabe urdinean, baina sekuentziaren %50a ezagutuz. Zutabe gorrietan gauza bera egiten da, baina eredu osoa kontuan hartuz lortutako datuekin. Taularen bukaeran, bi lerro grisetan, k-ren batezbestekoa eta moda ageri dira.

PAM	F-measure %25		F-measure %50		MF-measure %25		MF-measure %50	
Exekuzioak	k	Emaitza	k	Emaitza	k	Emaitza	k	Emaitza
run_0	50	0.25178576	50	0.19322124	50	0.34907275	50	0.33921468
	100	0.24570613	100	0.1843011	100	0.3391754	100	0.32219538
run_1	100	0.22729988	100	0.16138725	100	0.31456938	50	0.28182936
	400	0.22549525	50	0.16085339	150	0.3122563	100	0.27993068
run_2	50	0.23479575	50	0.17146432	50	0.31604183	100	0.28536284
	100	0.22103097	100	0.16898225	100	0.30357286	200	0.28536284
run_3	50	0.24836446	50	0.18937631	50	0.3331546	50	0.3148699
	150	0.21115777	400	0.18460074	400	0.29407987	400	0.30590412
run_4	50	0.24673386	50	0.2032876	50	0.3367046	50	0.33449334
	150	0.22972836	200	0.17760046	150	0.3170705	200	0.29494682
run_5	100	0.24322283	100	0.18844044	100	0.32695192	100	0.31140482
	150	0.24322283	150	0.18844044	150	0.32695192	150	0.31140482
run_6	500	0.25472564	150	0.20494623	150	0.34430328	150	0.34337705
	150	0.25472564	200	0.20494623	200	0.34430328	200	0.34337705
run_7	500	0.20831847	100	0.1854409	500	0.29113257	100	0.30266288
	100	0.19822578	150	0.1854409	100	0.28119788	150	0.30266288
run_8	50	0.21070991	50	0.15442432	50	0.29567173	50	0.27140343
	200	0.18240589	200	0.14864516	200	0.26472098	200	0.25861993
run_9	**50**	**0.2626671**	**50**	**0.20792307**	**50**	**0.35287967**	**50**	**0.34685954**
	100	0.2626671	100	0.20792307	100	0.35287967	100	0.34685954
Batezbes.	155,00		120,00		140,00		127,50	
Moda	50,00		50,00		50,00		50,00	

Irudia 3.8: Balidazio gurutzatuko exekuzio bakoitzerako bi *k* hoberenak.

Ikus dezakegunez, *k* guztiak desberdinak dira, bai ebaluazio-neurri desberdinen artean eta baita neurri jakin bateko emaitzen artean ere, exekuzio desberdinentzat. Hori dela eta, argi ikus dezakegu laginak eragina duela *k*-rengan; izan ere, lehen esan bezala, laginak ondo eratuta baleude, *k* berdinak emango luke beti emaitza hoberena.

Modan ikusten denez, *k*=50 da gehienetan emaitza hoberena ematen duena.

3.5 Ondorioak

Emaitzak aztertu ondoren, honako ahulezia hauek aurkitu ditugu sisteman:

- *Clustering*-aren edo multzokatzearen emaitza hurrengo bi arrazoien arabera aldatzen da:
    1. Exekuzioaren arabera edo, zehazkiago esanda, exekuzio jakin horretan erabil-

tzen diren geruzen arabera. Hau da, balidazio gurutzatua egiteko, datu-basea hamar geruzatan banatzen da. Ikasteko zazpi geruza erabiltzen dira; balidazioa egiteko, bi, eta, testerako, bat. Hortaz, geruzen konbinazio desberdinak eginez, hamar exekuzio ezberdin sortzen dira.

3.8 irudiko taulan ikusi dugun moduan, exekuzio batetik bestera, emaitza hoberena ematen duen k aldatzen da.

2. Multzo kopuruaren (k parametroaren) arabera. Gure probetan, k-ri balio jakin batzuk eman dizkiogu. Balio hauek aukeratuak izan dira, balio txikien eta handiagoen portaera aztertzeko. Hala ere, ezin dugu inolaz ere ziurtatu errendimendu hoberena lortzen duen k balioa horien artean dagoenik; beraz, hurbilpen moduan baino ez digute balio.

   Datu-kopurua, datu-basea edo testuingurua aldatuko balitz, berriro ere k-ren azterketa guztia egin beharko genuke, egokiena zein den jakiteko eta, kasu honetan bezala, batezbestean ondoen portatzen dena itzul genezake, kasurik hoberenean.

   Dakigunez, konponbiderik onena k automatikoki lortzea izango litzateke, baina momentuz horrelakorik egiterik ez daukagunez, zer gertatuko litzateke beste *clustering* algoritmo batekin? Emaitzak hobetuko lirateke?

- Profilak sortzea (proiektu honetatik at gelditzen da).
- Profil antzekoena bilatzea. Sistemak sekuentziak k-ren arabera multzokatu dituenean, profilak sortu eta aztertu beharreko sekuentzia hartzen du. Sekuentzia honetatik, interesatzen zaigun zatia (%25 edo %50) hartu eta sortu berri dituen zein profiletara egokitzen den hobekien begiratzen du. Edonola ere, baliteke profil egokienaren bilaketa hobetzeko aukerak egotea.

Sistemaren puntu horiek ikusita, beraz, badaukagu nondik hasi sistema hobetzen. Hurrengo kapituluan, *clustering*-aren atala hobetzen saiatu gara.

4. kapitulua

Clustering atalean aldaketak

Kapitulu honetan, sistemak *clustering*-a egiteko erabiltzen duen atala moldatu dugu, sistemaren errendimendua hobetzeko asmoz. Aurreneko probetatik ateratako ondorioetan ikusi dugun moduan, bi arazo nagusi dauzkagu: alde batetik, balidazio gurutzatuaren exekuzio bakoitzean emaitza hoberena ematen duen *cluster* kopurua aldatzen dela, eta, bestetik, *k* hoberena bilatzea zaila dela, hainbeste aukeren artean, gutxi batzuk besterik ez baititugu probatu.

Hasteko, lehenengo arazoa aztertzeari ekin diogu.

4.1 Algoritmoa aldatzea

Dakigunez, sistemak PAM algoritmoa erabiltzen du URL sekuentziak multzokatzeko. Algoritmo horrek, aurretik esan bezala, *k* parametroa behar du. Sistema ideala parametro hori automatikoki egokituko lukeena da. Horretarako, ordea, beharrezkoa da lehenengo bere portaera aztertzea. Hala ere, nola dakigu PAM dela algoritmo egokiena? Ez dakigu beste mota bateko algoritmo batekin funtzionamendua hobetuko litzatekeen edo ez. Hori dela eta, sisteman algoritmo hierarkiko bat inplementatzea pentsatu dugu.

Datu-meatzaritzan, *clustering* hierarkikoa *cluster*-ak analizatzeko metodo bat da, soilik multzoak egin ordez, *cluster*-en hierarkia osatzen duena. Bi estrategia daude:

1. **Aglomeratiboa**: Objektu bakoitza *cluster* bat da hasieran eta, hierarkian maila bat igo ahala, bi *cluster* elkartzen dira, guztiak *cluster* bakarrean elkartuta gelditzen diren arte.

2. **Zatikatzailea (*divisive*)**: Objektu guztiak *cluster* batean daude hasieran eta, hierarkian maila bat jaitsi ahala, *cluster* bat bitan banatzen da, harik eta objektu bakoitza *cluster* batean dagoen arte.

Zein *cluster* elkartu edo *cluster*-ak nola banatu erabakitzeko, multzoen arteko desberdintasuna neurtu behar da. Guk algoritmo hierarkiko aglomeratiboekin egin dugu lan. Hortaz, elkarketak egiteko, metrika egokia eta elkarketa-irizpideak behar ditugu. Alde batetik, metrikari dagokionez, PAM-en kasuan esan dugun bezala, datuak zenbakizkoak izan ordez, sekuentziak direnez, bi objekturen arteko distantzia kalkulatzeko Levenshtein distantzia erabiltzen dugu. Bestetik, elkarketa-irizpideak multzoen arteko distantzia kalkulatzeko bi objekturen arteko distantzia erabiltzen du, eta zein bi objektu erabili hautatzeko aukera desberdinak daude:

- **Complete Linkage**: *Cluster* bakoitzeko nodoen arteko distantzia maximoa.
- **Single Linkage**: *Cluster* bakoitzeko nodoen arteko distantzia minimoa.
- **Average Linkage**: *Cluster* bakoitzeko nodoen arteko batezbesteko distantzia.
- **Ward Linkage**: Elkarketaren ondorioz sortutako *cluster*-aren bariantza areagotzea.

Algoritmo hierarkikoetan, emaitzak dendrograma baten bidez adierazten dira, erroan datu guztiak *cluster* batean bilduta daudelarik, eta, hostoetan, datu bakoitza *cluster* batean. Horrela, partizioen sekuentzia bat adierazten dute; maila bakoitzean partizio bat. Partizio bakoitzak *cluster*-en banaketa desberdina dauka. Lehenengo partizioak n (objektuak adina) *cluster* ditu eta partizioa birfinduz joaten da, datu guztiak *cluster* bakar batean biltzen diren arte.

Algoritmo hierarkiko desberdinen artetik, SAHN (*Sequential, Agglomerative, Hierarchical and Nonoverlapping*) [William H. E. Day, 1984] aukeratu dugu sisteman inplementatzeko, *k-means* algoritmoaz gain, asko erabiltzen den beste *clustering* algoritmoa baita. Hori gutxi balitz, distantzia edo metodo desberdinak probatzeko aukera ematen digu. Hurrengo azpiatalean ikusiko dugu nola egiten duen lan algoritmoak.

4.1.1 SAHN algoritmoa

Izenak dioen moduan, algoritmo hau sekuentziala, aglomeratiboa eta hierarkikoa da eta ez da gainjartzen. Sekuentziala dela esaten da momentuko partizioa erabiltzen duelako hurrengoa kalkulatzeko. Aglomeratiboa da hasieran objektu bakoitza *cluster* batean izatetik, objektu guztiak multzo bakarrean dituela bukatzen duelako, pauso bakoitzean bi *cluster* antzekoenak elkartuz. Gainera, hierarkikoa da partizioen sekuentzia bat adie-

razten duelako, dendrogramaren bitartez. Bukatzeko, ez da gainjartzen, izan ere, partizio bakoitzak *cluster*-en banaketa desberdina dauka.

SAHN *clustering* metodoari, funtzionatu ahal izateko, bi *cluster*-en arteko distantzia balio kuantitatiboarekin adierazi behar zaio, zein bi *cluster* elkartu behar dituen erabaki ahal izateko. Distantzia horiek adierazteko, oro har bi modu daude: alde batetik, distantzien matrizea ematea, non *cluster* konbinazio guztien distantziak adierazten diren, edo, bestela, distantzia kalkulatu ahal izateko beharrezko informazioa ematea. Distantzia neurtzeko, normalean, Manhattan metrika edo metrika Euklidearra erabiltzen bada ere, guk, orain arte bezala, Levenshtein distantzia erabiltzen dugu, sekuentziekin ari garelako lanean.

Honela funtzionatzen du algoritmoak:

Algoritmoa 2 *SAHN*

1: **for** $m = n$-tik 2-ra **do**
2: Distantzien matrizean distantzia txikiena duen *cluster* bikotea bilatu (i eta j)
3: Ordezkatu hurbileneko bi *cluster*-ak (i eta j) hauek multzokatzen dituen *cluster* berriarekin (h)
4: Eguneratu distantzien matrizea, elkartu berri ditugun *cluster*-en (i eta j) distantziak ezabatu eta berriarenak (h) sartuz. Horretarako, *cluster* berriak beste guztietaraino duen distantzia kalkulatu behar da
5: **end for**
6: Erakutsi *cluster* aglomeratuen hierarkia

Hierarkia lortu ondoren, ordea, partizio bat lortu behar dugu. Hori egiteko aukera ugari daude, adibidez, hierarkia *threshold* edo muga-balio baten arabera moztea, hau da, ehuneko jakin baten arabera. Demagun dendrogramak 60 unitateko altuera duela, hau da, bi cluster urrunenak 60 unitateko distantzia daukatela. Dendrograma %80-an mozteko eskatzen badugu, $60 * 0.8 = 48$ unitateko altueran mozten da dendrograma. Horrela, 48 unitateko altueran marraztu dugun lerro zuzen horren bitartez, *cluster*-ak lortzen dira. Edonola ere, horrela ezin dugu inoiz jakin zenbat *cluster* sortzen diren eta, PAM-ekin lortutako emaitzekin konparatzekotan, beharrezkoa da bietan sortzen den *cluster*-kopurua bera izatea.

Hori dela eta, guk dendrograma k-ren arabera mozten dugu; errotik abiatuta, dendrograma beheraka zeharkatuz goaz momentuko k kopurua adina *cluster* dituen partizio bat aurkitzen dugun arte.

Distantzien matrizea nola eguneratu

Distantzien matrizea eguneratzeko, i eta j *cluster* zaharrak elkartuz sortutako h *cluster* berriaren eta beste edozein k *cluster*-en arteko distantzia kalkulatzeko proposamen bat honakoa da:

$$d(h,k) = \alpha_i d(i,k) + \alpha_j d(j,k) + \beta d(i,j) + \gamma |d(i,k) - d(j,k)|$$

non

Clustering Metodoa	α_i	α_j	β	γ
Single Linkage	$1/2$	$1/2$	0	$-1/2$
Complete Linkage	$1/2$	$1/2$	0	$1/2$
Group Average	$\frac{n_i}{n_i+n_j}$	$\frac{n_j}{n_i+n_j}$	0	0
Minimum Variance (Ward)	$\frac{n_i+n_k}{n_i+n_j+n_k}$	$\frac{n_j+n_k}{n_i+n_j+n_k}$	$\frac{-n_k}{n_i+n_j+n_k}$	0

Taula 4.1: Distantzien matrizea eguneratzeko kalkuluak egiteko datuak.

n_i, n_j eta n_k i., j. eta k. *cluster*-etan dauden objektu kopuruak dira, hurrenez hurren. $d(a,b)$ balioa d distantzia-matrizeak adierazten duen a eta b puntuen arteko distantzia da.

4.1.2 Inplementazioa

Inplementatuta dagoen PAM algoritmoa alde batera utziz, sisteman SAHN algoritmoa txertatzeko beharrezko inplementazioak azaltzen ditugu atal honetan, algoritmo hierarkikoekin nolako funtzionamendua duen aztertzeko. Gainera, sistemaren lau bertsio desberdin sortu ditugu, *cluster*-en arteko distantzia kalkulatzeko 4 metrika ezberdin erabiliz: *ward linkage*, *average linkage*, *single linkage* eta *complete linkage*.

Ezertan hasi baino lehen, partizioak gordetzeko egitura sortu behar da. Dendrogramak erroa (*root*), tarteko nodoak (*MergeNode*-ak) eta hostoak (*observation*-ak) dauzka. Erroa datu guztiak *cluster* bakarrean biltzen dituen partizioa da. Nodoek (tartekoek edo hostoek) beren ezkerreko eta eskuineko umea lortzeko funtzioak erabil ditzakete, baita nodoaren azpian zenbat hosto dauden jakiteko ere. Hostoen kasuan, ezkerreko eta eskuineko umea *null* dira, eta hosto kopurua, 1. Horretaz gainera, hostoek, identifikatzeko, zenbaki bat daukate.

Lehenengo ikus dezagun programa nagusia (3. algoritmoa). Bertan, aurrerago azaltzen ditugun funtzio desberdinei deitzen zaie SAHN algoritmoa exekuta dadin.

Aurreneko aginduarekin, datu-basetik balidazio gurutzatuaren eredua lortzen dugu $sortuCVBanaketa()$ funtzioari deituz eta emaitza $CVEredua$ aldagaian gordetzen da.

Algoritmoa 3 main()

1: $CVEredua \leftarrow sortuCVBanaketa()$
2: $dendrogramak \leftarrow sortuDendrogramak(metrika)$
3: $partizioak \leftarrow moztuDendrogramak(k)$
4: $emaitzak \leftarrow ebaluazioa()$

Funtzioak geruzak sortzeko datu-basea eta balidazio gurutzatuaren banaketaren aldagaiak erabiltzen ditu. Bigarren aginduarekin, balidazio gurutzatuaren banaketa bakoitzarentzat dendrogramak sortzen dira, hau da, 10 dendrograma. Horretarako, *sortuDendrogramak*() funtzioak elkarketak egiteko zein metrika (*ward*, *average*, *single* edo *complete*) erabili behar dugun jaso behar du parametro moduan. Sortzen dituen 10 dendrogramak *dendrogramak* aldagaian gordetzen dira. Ondoren, *moztuDendrogramak*() funtzioak, k jakin bat adieraziz, *dendrogramak* aldagaian dauden dendrogramak atzituz, 10 partizio sortzen ditu, bat dendrograma bakoitzarentzat, hots, balidazio gurutzatuaren exekuzio bakoitzarentzat, eta *partizioak* aldagaian gordetzen ditu. Azken pausoan, *ebaluazioa*() funtzioari deitzen zaio aurreko pausoan sortutako partizio bakoitzarekin sistema proba dezan eta *emaitzak* aldagaian gordetzen dira lortutako balioak.

Ondoren, dendrogramak sortzeko funtzioa daukagu (4. algoritmoa):

Algoritmoa 4 sortuDendrogramak($metrika$)

1: **for all** $geruza \in CVEredua$ **do**
2: $dendrograma \leftarrow Dendrograma(geruza$-ko hosto kopurua$)$
3: $distantziaMatrizea \leftarrow computeDissimilarityMatrix()$
4: $cluster()$
5: $dendrogramak \leftarrow dendrogramak \cup dendrograma$
6: **end for**
7: **return** *dendrogramak*

Balidazio gurutzatuaren ereduko (*CVEredua*) geruza bakoitzeko, *dendrograma* aldagaian gordetzen den dendrograma sortzen da, geruzako hosto kopurua adina *cluster* dituela hasieran. Ondoren, geruzako datuekin distantzia-matrizea kalkulatzen da, *computeDissimilarityMatrix*() (ikus 5. algoritmoa) funtzioaren bitartez eta emaitza *distantziaMatrizea* aldagaian uzten du. *cluster*() funtzioari (ikus 6. algoritmoa) deituz, *cluster*-ak elkartzen dira gero, pausoka, guztiak bakar batean, erroan, egon arte. Jarraian, sortu berri dugun dendrograma *dendrogramak* multzoan gehitzen da. Horrela, balidazio gurutzatuaren 10 exekuzioak bukatzean, *dendrogramak* multzoan dauzkagu behar ditugun 10 dendrogramak.

Dendrogramak sortzeko algoritmoan agertzen den *computeDissimilarityMatrix*() funtzioak distantzia-matrizea kalkulatzeko zer egiten duen ikus dezakegu 5. algoritmoan.

Algoritmoa 5 computeDissimilarityMatrix()

```
1: for all hosto1 ∈ hostoak do
2:     for all hosto2 ∈ hostoaketaidentifikatzailea(hosto2) < identifikatzailea(hosto1)
       do
3:         distantzia ← computeDissimilarity(metrika, hosto1, hosto2)
4:         sartuDistantziaMatrizean(hosto1, hosto2, distantzia)
5:     end for
6: end for
7: return distantziaMatrizea
```

hostoak multzoan momentuko balidazio gurutzatuaren geruzan ikasteko erabiltzen diren hosto guztiak dauzkagu gordeta. Bertan dagoen hosto bakoitzak (*hosto*1) bere identifikatzailea baino identifikatzaile txikiagoa duten hosto guztietara (*hosto*2) duen distantzia kalkulatutzen dugu *computeDissimilarity*() funtzioak itzultzen digun balioa *distantzia* aldagaian gordez. Azken funtzio horrek bi hostoren arteko distantzia kalkulatzen du, parametro moduan adierazten zaion metrika erabiliz. Ondoren, distantzien matrizean sartzen dugu balioa *sartuDistantziaMatrizean* funtzioari bi hostoak eta distantziaren balioa pasaz; amaitzeko, distantzia-matrize osatua itzultzeko.

Aurretik aipatu dugun bezala, dendrograma sortzerakoan *cluster*() funtzioari deitzen zaio *cluster*-ak binaka elkartzen joan dadin, datu guztiak multzo bakar batean gelditzen diren arte. 6. algoritmoan azaltzen da bere funtzionamendua.

Algoritmoa 6 cluster()

```
1: for elkarketa = 1 to hostoKopurua do
2:     bikotea ← findMostSimilarClusters()
3:     distantzia ← distantziaMatrizea(bikotea)
4:     for all hostoa ∈ hostoak do
5:         if hostoa ∉ bikotea then
6:             distBerria ← recomputeDissimilarity(hostoa, bikotea)
7:             sartuDistantziaMatrizean(bikotea, hostoa, distBerria)
8:         end if
9:     end for
10:    elkartuNodoak(bikotea, distantzia)
11:    eguneratuDendrograma()
12: end for
13: return dendrograma
```

Dendrograma osatzeko egin beharreko elkarketa-kopurua dendrograman bertan

dauzkagun hosto-kopurua baino bat gutxiago da. Horregatik, *for* begizta nagusia *hostoKopurua* aldagaian daukagun balioarekin mugatzen da. Begiztaren iterazio bakoitzean, *findMostSimilarClusters*() (ikus 7. algoritmoa) funtzioari esker, *bikotea* aldagaian distantzia-matrizean momentuan dauzkagun hurbileneko bi *cluster*-ak jasoko ditugu. Ondoren, *distantzia* aldagaian, distantzia-matrizea atzituz (*distantziaMatrizea* izenekoa), *bikotea* aldagaian gorde ditugun bi *cluster*-en arteko distantzia gordetzen dugu. Hurrengo *for*-aren bitartez, *hostoak* multzoan dagoen, baina momentuan elkartu behar dugun bikotekoa (*bikotea*) ez den hosto (*hostoa*) bakoitzeko, *bikotea* aldagaiko bi *cluster*-ak elkartzean sortzen den *cluster* berriaren eta *hostoa*-ren arteko distantzia berria kalkulatzen da, *recomputeDissimilarity*() funtzioari deituz. Funtzio honek 4.1.1 atalean azaldutako formulak erabiltzen ditu. Iterazio bakoitzeko lanekin bukatzeko, *bikotea* aldagaiko bi nodoak elkartzen ditugu *elkartuNodoak*() funtzioari deituz eta, ondoren, dendrograma eguneratzen da. Elkarketa guztiak egin ostean, *dendrograma* itzultzen du.

Jarraian, 7. algoritmoan, bi *cluster* antzekoenak bilatzeko *findMostSimilarClusters*() funtzioaren sasikodea erakusten dugu.

Algoritmoa 7 findMostSimilarClusters()

```
1: for all clusterra ∈ clusterrak do
2:   for all auzokoa ∈ clusterrak do
3:     if distantziaMatrizea(clusterra, auzokoa) < distantziaMin eta clusterra ≠
       auzokoa then
4:       distantziaMin ← distantziaMatrizea(clusterra, auzokoa)
5:       setBikoteAntzekoena(clusterra, auzokoa)
6:     end if
7:   end for
8: end for
9: return bikoteAntzekoena
```

Funtzio honen bitartez, *cluster* bikote guztiak aztertzen dira. Horretarako, *clusterrak* multzoan momentuko *cluster* guztiak dauzkagu bilduta. Bertatik, banaka, *clusterra* aldagaian multzoko osagai guztiak hartzen ditugu. Ondoren, multzoko beste *cluster* bat hartzen dugu *auzokoa* aldagaian. Honela konbinazio guztiak probatzen ditugu. Konbinazio bakoitzeko, *clusterra* eta *auzokoa* ez badira *cluster* bera (kasu horretan distantzia 0 delako) eta bien arteko distantzia momentura arte aurkitu duguna (*distantziaMin*) baino txikiagoa bada, *distantziaMin* aldagaia eguneratzen da, *clusterra* eta *auzokoa*-ren arteko distantziarekin. Bikote antzekoena ere eguneratzen dugu *setBikoteAntzekoena*() funtzioari deituz, *clusterra* eta *auzokoa* pasata.

Programa nagusiko bigarren pausoa aztertzez bukatuta, dendrogramak mozteko funtzioa aztertzea falta zaigu. Horretarako sasi-kodea 8. algoritmoan daukagu.

Algoritmoa 8 *moztuDendrograma*(k)

```
1:  for all geruza ∈ CVEredua do
2:      erroa ← dendrogramaren erroa
3:      nodoZerrenda ← nodoZerrenda ∪ erroa
4:      for all nodoa ∈ nodoZerrenda do
5:          if nodoZerrenda-n aztertzeke dauden nodo kopurua ≥ k then
6:              clusterZerrenda ← clusterZerrenda ∪ nodoa
7:              k = k − 1
8:          else
9:              if nodoa hostoa da then
10:                 clusterZerrenda ← clusterZerrenda ∪ nodoa
11:                 k = k − 1
12:             else
13:                 nodoZerrenda ← nodoZerrenda ∪ nodoa-ren ezkerreko umea
14:                 nodoZerrenda ← nodoZerrenda ∪ nodoa-ren eskuineko umea
15:             end if
16:         end if
17:     end for
18:     return clusterZerrenda
19: end for
```

Funtzio honi parametro moduan lortu nahi dugun *cluster* kopurua (k) pasa behar zaio. Balidazio gurutzatuaren exekuzio bakoitzeko (*geruza*), *erroa* aldagaian *geruza*-ko dendrogramaren erroa jasotzen dugu. Dendrograma goitik behera zeharkatu ahal izateko, *nodoZerrenda* izeneko aldagaia daukagu. Multzo honetan, hasieran, *erroa* gordetzen dugu. Ondoren, *nodoZerrenda*n dagoen *nodoa* nodo bakoitzeko honakoa begiratzen da: ea *nodoZerrenda*n aztertzeke gelditzen den nodo-kopurua eta lortu nahi ditugun *cluster*-kopurua (k) berdinak diren. Hasieran gertatzen den moduan, berdinak ez badira, bi aukera daude. Alde batetik, aztertzen ari garen nodoa hostoa izan daiteke; orduan, hostoa zuzenean *clusterZerrenda* multzoan sartzen dugu, aldagai honetan itzuliko dugun partizioko *cluster*-ak gordetzen baititugu. Ahaztu gabe, noski, k balioa dekrementatu behar dugu, dagoeneko *cluster* bat gehiago baitaukagu partizioan. Bestetik, ordea, hostoa ez bada, *nodoZerrenda* multzoan, momentuko *nodoa*-ren ezker eta eskuin umeak gehitzen ditugu.

Lehenengo if-eko baldintza betetzen denean, hau da, *nodoZerrenda* aldagaian aztertzeke dauden nodo-kopurua eta k aldagaia berdinak direnean, *nodoZerrenda*n dauzkagun nodoak *clusterZerrenda* multzoan sartzen ditugu. Horretarako, momentuko nodoa

partizioa adierazten duen *clusterZerrenda* multzoan gehitu eta *k* aldagaia txikiagotzen dugu, hurrengo iterazioan berriro sar dadin. Nodo guztiak pasatzen dituenean, partizioa itzultzen du *clusterZerrenda* aldagaiaren bitartez.

4.1.3 Probak

Algoritmo hierarkikoaren funtzionamendua aztertzeko, lau bertsio desberdin sortu ditugu. Bertsio bakoitzak distantzia kalkulatzeko aukera desberdina erabiltzen du: *single linkage*, *complete linkage*, *group average* eta *ward linkage*. Lehenengo, taula bakoitzean, bertsio hauetako bakoitzaren emaitzak ikus daitezke. *Cluster*-kopuru bakoitzerako, balidazio gurutzatuaren hamar exekuzioetan zehar lortutako *f-measure* emaitzen batezbestekoak bakarrik erakusten ditugu.

Average					
Cluster kopurua	Proposamen kopurua	F-measure %25	F-measure %50	MF-measure %25	MF-measure %50
50	4.0	0.2278604	0.18827735	0.3137204	0.3137204
100	4.0	**0.23346174**	**0.19218656**	**0.31990647**	**0.31990647**
150	4.0	0.22860663	0.18994637	0.31359985	0.31359985
200	3.8	0.2251368	0.18837336	0.3134902	0.31343848
300	4.0	0.20716651	0.17213771	0.29012886	0.28987005
400	4.0	0.22061086	0.18287124	0.30506527	0.30477872
500	4.0	0.21696818	0.17998597	0.3011903	0.30085266

Irudia 4.1: *Average* bertsioaren emaitzak.

Complete					
Cluster kopurua	Proposamen kopurua	F-measure %25	F-measure %50	MF-measure %25	MF-measure %50
50	4.0	0.22283301	0.18439864	0.30823025	0.3080468
100	4.0	0.23000637	**0.19087751**	0.31511056	**0.31496984**
150	4.0	0.21825686	0.18137458	0.3014229	0.3014441
200	4.0	0.21908538	0.18052328	0.30244	0.30159646
300	4.0	0.21486004	0.17738467	0.29846758	0.29664737
400	4.0	0.22933969	0.18907689	0.31552666	0.3145131
500	4.0	**0.23005779**	0.18734898	**0.31629854**	0.3129538

Irudia 4.2: *Complete* bertsioaren emaitzak.

Argi ikus daitekeenez, emaitzarik onenak *single* aukera erabiltzen duen bertsioak ematen ditu, eta alde handiarekin, gainera. Hainbesteko diferentziaren nondik norakoak

Single					
Cluster kopurua	Proposamen kopurua	F-measure %25	F-measure %50	MF-measure %25	MF-measure %50
50	2.3	0.30826843	0.26477796	0.43022522	0.43022522
100	2.4	0.2995761	0.25758642	0.4191617	0.4191617
150	2.1	0.31411093	0.2720245	0.440703	0.440703
200	2.0	**0.31690663**	**0.27510324**	**0.4457**	**0.4457**
300	2.0	**0.31690663**	**0.27510324**	**0.4457**	**0.4457**
400	2.8	0.27490705	0.23496208	0.3860327	0.3860327
500	2.0	**0.31690663**	**0.27510324**	**0.4457**	**0.4457**

Irudia 4.3: *Single* bertsioaren emaitzak.

Ward					
Cluster kopurua	Proposamen kopurua	F-measure %25	F-measure %50	MF-measure %25	MF-measure %50
50	4.0	**0.21812049**	**0.177389**	**0.3031231**	**0.29823348**
100	4.0	0.21215145	0.17026995	0.29592267	0.28952256
150	4.0	0.19356516	0.1649138	0.27377686	0.276438
200	4.0	0.19449829	0.16027534	0.27675036	0.2737207
300	4.0	0.19061247	0.16018227	0.27130064	0.27196527
400	4.0	0.18391979	0.15255234	0.2642625	0.26014158
500	4.0	0.19380079	0.16104816	0.27604362	0.2730146

Irudia 4.4: *Ward* bertsioaren emaitzak.

bilatzen hastean, berehala topo egin dugu arrazoiarekin. Zutabe laranjan adierazten den moduan, *single* bertsioa ez da gai eskatzen zaizkion 4 URLak proposatzeko; batezbeste, 2.22 URL proposatzen ditu, eskatutakoaren erdia ia. Horrek esan nahi du bertsio horrek egiten dituen *cluster*-ak ez direla homogeneoak, hots, *cluster*-etan elkartzen dituen sekuentziek ez dutela antzekotasunik elkarren artean. Horregatik, sistemak ez ditu *SPADE*-ren parametroa den *minimum support*-a gainditzen duten URLak aurkitzen, hau da, *cluster*-eko sekuentzien %20an agertzen diren URLak. 3.3 atalean azaldu dugun bezala, URL gutxiago proposatzean, *precision* handiagotu egiten da eta *recall*-a txikiagotu. Gure kasuan, *precision*-i indar gehiago ematen zaionez, emaitzak asko hobetzea lortzen du bertsio horrek. Ondorioz, emaitzak ez direnez ez baliozkoak eta ez konparagarriak, *single* metrika erabiltzen duen bertsioa erabat bazter dezakegu.

Beste hiru bertsioei dagokienez, emaitzak antzekoak dira, nahiz eta *ward*-enak beste biak baino zertxobait kaxkarragoak diren.

Algoritmo hierarkikoaren bertsio desberdinak probatu arren, emaitza hoberenak

ematen dituen k bertsioaren arabera (baita bertsioan bertan ere, aztertzen dugun arazoaren arabera) aldatzen da. Parametroak hainbesteko aldakortasuna daukanez, ezinezkoa da sistema datu hauentzat bakarrik ere egonkortzea. Bestalde, taularik erakutsi ez badugu ere, balidazio gurutzatuko exekuzioak aztertzean, *cluster* kopuruaren arabera, exekuziorik onena zein den ere aldatzen da, baita exekuzio bakoitzarentzat k hoberena zein den ere. Beraz, sistema PAM-ekin inplementatuta genueneko egoera berean gaude.

4.1.4 Ondorioak

Clustering algoritmoa aldatuta ere, ikusi dugu ez dela gure arazoa konpontzen. Gainera, lortu ditugun emaitzak PAM-ekin lortzen direnak baino kaxkarragoak izan dira. Oraindik ere, datuek daukaten eragina dela eta, balidazio gurutzatuko exekuzio bakoitzean, k hoberenaren balioa aldatzen da. Sistema orokorra izatea nahi dugunez, datuen eragina gutxitzeko modu bat bilatu behar dugu. Horretarako, kapituluaren hasieran aipatutako *clustering*-ean eragina duten bigarren arazoari egin behar diogu aurre: k parametroa edo *cluster* kopurua. Hautatu al dezakegu automatikoki balio hau? k parametroa automatikoki aukeratzea lortuko bagenu, sistema asko orokortuko genuke. Izan ere, momentuz, PAM-ekin daukagun bertsioa BTw ingurunearen datuentzako ari gara prestatzen, baina k automatikoki definitzea lortuz gero, ez luke garrantziarik izango sistema zein daturekin jartzen dugun martxan, edozein ingurunekoak direla ere, aurre-prestaketarik ez bailuke beharko.

4.5 irudian, algoritmo desberdinekin sortutako bertsioen emaitza hoberenak, k edozein dela ere, biltzen dira. *Single* bertsioa kolore desberdinez nabarmendu dugu ez delako baliozkoa; hemendik aurrera ez dugu erakutsiko.

Bertsioa	F-measure %25	F-measure %50	MF-measure %25	MF-measure %50
PAM	0.22991998	0.18031698	0.31587023	0.3044352
Average	**0.23346174**	**0.19218656**	**0.31990647**	**0.31990647**
Complete	0.23005779	0.19087751	0.31629854	0.31496984
Single	**0.31690663**	**0.27510324**	**0.4457**	**0.4457**
Ward	0.21812049	0.177389	0.3031231	0.29823348

Irudia 4.5: Sistemaren hasierako bertsioa eta algoritmo hierarkikoarekin lortutako bertsioen emaitza hoberenen konparazioa.

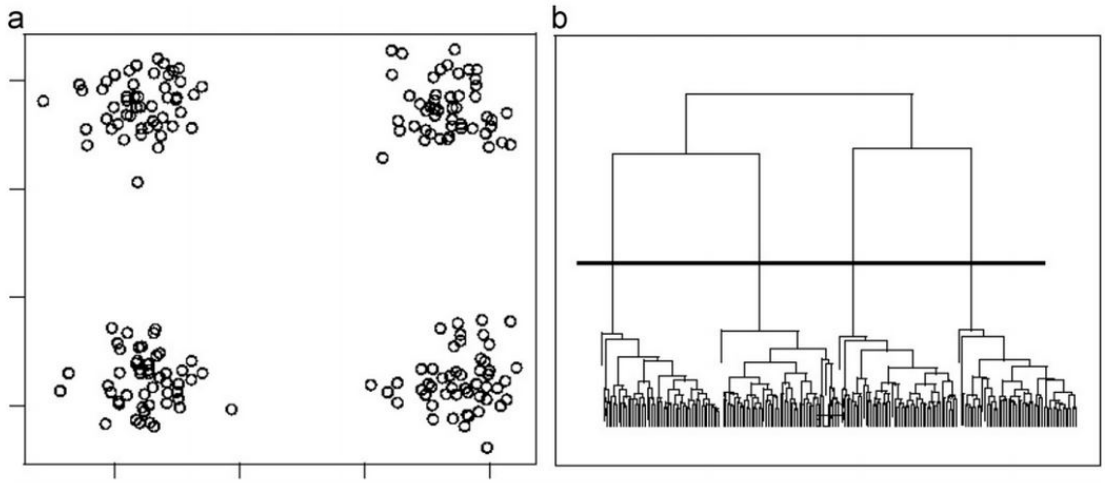
**__OHARRA:__ Taulan agertzen diren balioak algoritmo bakoitzarekin sortutako bertsioek k jakinen batentzako emandako emaitzarik __HOBERENAK__ dira eta ez

batezbestekoak. Hori dela eta, lehenengo taula honetan, *average* bertsioak emaitza hobea eman du PAM-en bertsioak baino, azken hau batezbestean bertsio hobea bada ere.

4.2 *k* parametroaren bilaketa automatikoa

Algoritmo hierarkikoek, *cluster*-en hierarkia osoa erakusten dutenez, partizio bakar batek baino informazio gehiago eskaintzen dute. Baina, aldi berean, aztertzeko konplexuagoak dira. Horregatik, oro har, zuhaitz moduko egitura daukan dendrogramaren bidez adierazten da hierarkia osoa. Partizioko *cluster*-ak dendrogramako nodoen bitartez adierazten dira. Modu berean, adarrek elkarketak adierazten dituzte.

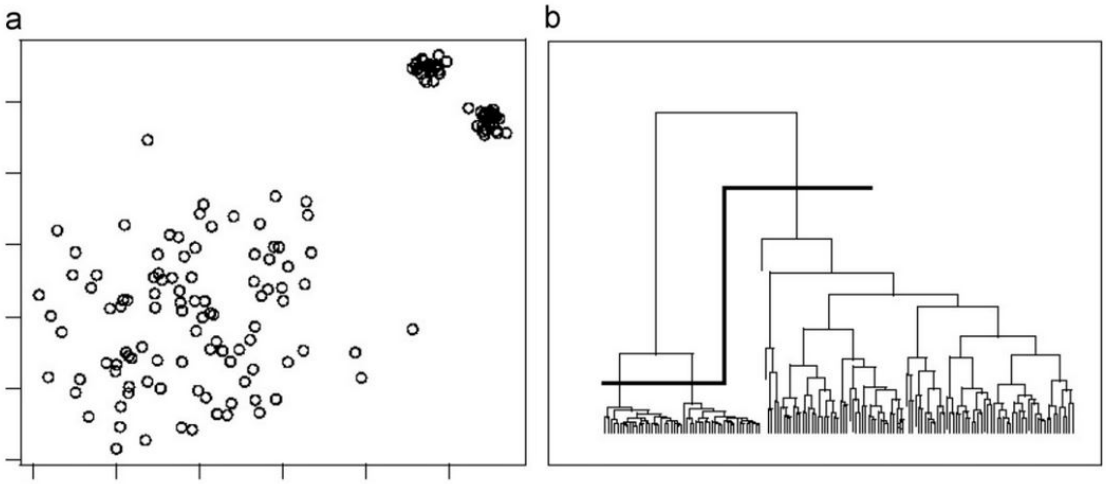
Partizioen ordena dendrogramako nodoen altueraren bitartez adierazten da. Hasierako pausoetan sortutako elkarketak dendrogramaren azpiko zatian aurki daitezke; azkeneko elkarketa, *cluster* bakarreko partizioa sortzen duena, berriz, zuhaitzaren goiko puntan dago, erro nodoan. Hierarkiako edozein partizio lortzeko nahikoa da dendrograma horizontalean ebakitzea. Zuhaitza goragotik edo beheragotik moztuz gero, *cluster* gutxiagoko edo gehiagoko partizioa lortzen da, hurrenez hurren. 4.6 irudian ikus daitezke 4 multzotan banatuta dauden datuak eta dagokien dendrograma. *b* irudian, lerro horizontal baten bidez mozten da dendrograma eta datuetan garbi ikusten diren 4 *cluster*-ak lortzen dira.



Irudia 4.6: Datuak eta dagokien dendrograma, horizontalean ebakita

Kasu batzuetan, ordea, dendrograma lerro horizontal batekin ebakitzean, ez da partizio hoberena lortzen. Hala ere, 4.7 irudian ikus dezakegun moduan, datuetan ageri diren hiru multzoak dendrograman adierazita daude, baina, lerro horizontal zuzena izan ordez, ja-

rraia ez den beste baten bitartez mozten da, partizio hoberena lortu ahal izateko. Ondorioz, hierarkiak partiziorik onena esplizituki erakusten ez badu ere, bertan adierazten da.



Irudia 4.7: Datuak eta dagokien dendrograma

Dendrograma sortzerakoan adarrak moztea proposatzen duenik badago ere, normalean hierarkia ondoren prozesatzen da, horizontalean ebakitzeko. Azaldutako arazoa saihesteko baina, SEP (*Search over the Extended Partition set*) [Gurrutxaga, 2010] algoritmoa erabili dugu, bilaketa egiteko dendrogramako partizio guztiak, hau da, dendrograma edozein modutan ebakita lor daitezkeen partizioak, kontuan hartzen dituen algoritmoa. Bilaketa-espazio horri *partizio-multzo zabaldua* deitzen diogu.

Partizio-multzo zabalduan egindako bilaketak nahitaez hierarkian egindako bilaketak ematen dituen emaitza berak edo hobeak eman behar ditu, hierarkian dauden partizio guztiak partizio-multzo zabalduan ere badaudenez gero.

Bestalde, normalean, hierarkiako partizio guztiak (edo batzuk) *cluster*-aren baliozkotasun-indizea (CVI) deituriko neurri batean oinarrituta aztertzen dira, hoberena aukeratzeko balio kuantitatiboa lortzeko asmoz. CVI-a partizioak ebaluatzen dituen indizea da, *cluster* barruko objektuen kohesioa eta *cluster* artekoen urruntasuna edo diferentzia neurtuz. Bi motatako indizeak daude: alde batetik, balio txikiek partizio hobeak adierazten duten indizeak, eta, bestetik, zenbat eta balio altuagoa eman, orduan eta partizio hobea adierazten duten partizioak. Hurrengo azpiataletan ikusiko dugunez, SEP-en berezitasunak direla eta, ezin dugu edozein CVI erabili. Horregatik, SEP-ekin batera, COP indizea proposatzen da, SEP-en beharrak asetzen dituen indizea.

4.2.1 SEP algoritmoa

Jatorrizko SEP algoritmoa azaltzen hasi baino lehen, azal ditzagun algoritmoan agertzen diren elementuak. Datu-base bat, $X = \{x_1, x_2, \ldots, x_N\} \in \Re^F$, F-espazioan bektore moduan adierazitako N objektu edo datupunturen multzoa da. *Cluster* bat, $C \subseteq X$, datu-baseko objektuen azpimultzo bat da. *Cluster* baten zentroidea, $\bar{C} = 1/|C| \sum_{x \in C} x$, *cluster*-eko bektore guztien batezbesteko bektorea da.

Datu-basearen partizio partzial bat, $P^Y = \{C_1, C_2, \ldots, C_K\}$ s.t. $\bigcup_{C_k \in P^Y} C_k = Y, C_k \cap C_l = \emptyset \quad \forall k \neq l$ and $Y \subseteq X$, datu-basearen azpimultzo baten *cluster* disjuntuen multzo bat da. Datu-basearen partizio totala, P^X, datu-base osoaren *cluster* disjuntuen multzoa da. Hemendik aurrera, edozein motatako partizioa dela ere, "partizioa" deitzen diogu. Datu-basearen partizio egokia, P^*, datu-baseko multzo naturalak adierazten dituen partizioa da, hau da, partizioko *cluster* bakoitzak datu-baseko puntuen multzo bat erakusten du.

Hierarkia bat, $H = \{P_1, P_2, \ldots, P_R\} \quad \forall P_r, P_s \quad r < s \Leftrightarrow \forall C_k \in P_r, \exists C_l \in P_s$ non $C_k \subseteq C_l$, elkarrekin lotutako partizioen multzo bat da. Bertako edozein partizio —lehenengoa izan ezik— aurreko partizioko *cluster*-en elkarketen bidez adieraz daiteke. Hierarkia bateko partizio zabaldua, $E_H = \{P : P \subseteq T, \bigcup_{C \in P} C = X$ eta $\forall C_k, C_l \in P \quad C_k \cap C_l = \emptyset\}$ non $T = \bigcup_{P \in H} \bigcup_{C \in P} C$, hierarkian aurkitu daitezkeen edozein *cluster* konbinaziorekin osatu daitekeen partizio-multzoa da.

Cluster-aren baliozkotasun indizea partizioa ebaluatzen duen funtzioa da, $\mathrm{V}(P)$. Jatorrizko lanean, besterik ezean, V-aren balio baxuagoak partizio hobea adierazten du. Partizio-multzo batean partiziorik hoberena bilatzen duen funtzioa honela definitzen dugu: $\mathrm{S}(\mathrm{V}, H) = \arg\min_{P \in H} \mathrm{V}(P)$ non V CVI bat den eta H partizio-multzo bat den.

Algoritmoak ebatzi nahi duen arazoa honakoa da: datu-base bat, X, eta hierarkia bat, H, emanik, bilatu $\hat{P} = \mathrm{S}(\mathrm{V}, E_H)$, partiziorik onena, V-an oinarrituta, H-ren partizio-multzo zabalduan.

SEP algoritmo errekurtsiboa da.

9. algoritmoari, parametro moduan V indizea eta *nodoa*, aztertzen ari garen nodoa, pasa behar dizkiogu. Datu horiek izanik, C aldagaian *nodoa*ri dagokion *cluster*-a gordetzen dugu. Jarraian, *nodoa* hostoa bada, zuzenean C *cluster*-a itzultzen dugu, aukerarik hoberena baita, besterik ez daukalako. Aldiz, *nodoa* ez bada hostoa, *elkarketa* aldagaia hutsik hasieratzen dugu. Bertan, *nodoa*ren azpitik dagoen partiziorik onena gordetzen dugu, hau da, bere umeen partiziorik onenak, elkartuta. Horretarako, *nodoa*ren *Umea* ume bakoitzeko, *elkarketa* aldagaiari *umea*ren SEP-en emaitza gehitzen diogu. *elkarketa*

Algoritmoa 9 SEP(V, *nodoa*)

```
1:  C ← nodoako clusterra
2:  if nodoa hostoa da then
3:      return {C}
4:  else
5:      elkarketa ← ∅
6:      for all umea ∈ nodoa do
7:          elkarketa ← elkarketa ∪ SEP(V, umea)
8:      end for
9:      if V({C}) < V(elkarketa) then
10:         return {C}
11:     else
12:         return elkarketa
13:     end if
14: end if
```

osatu dugunean, V indizeak C *cluster*-ari eta *elkarketa*ri ematen dizkion balioak aztertzen dira. C *cluster*-ak itzultzen duen balioa txikiagoa bada, *nodoa*ko *cluster*-a bera itzultzen dugu partiziorik onena delako, V indizearen balio txikiagoek partizio hobea adierazten baitute. *elkarketa*-ren balioa txikiagoa bada, berriz, *elkarketa* aldagaian dagoen partizioa itzultzen dugu.

Hitzez esanda, nodo bakoitzeko, hostoa bada, hostoa bera itzultzen dugu; bestela, nodoaren azpian dagoen partizio osoa aztertzen dugu, dei errekurtsiboak eginez. Partizio osoa lortu ondoren, nodoaren balioa baxuagoa bada (balio baxuagoak partizio hobea adierazten du) umeen elkarketaren balioa baino, orduan nodoa bera itzultzen dugu; bestela, elkarketa.

Nola nahi ere, lehen aipatu dugun bezala, SEP exekutatu ahal izateko indize berezi bat behar da, gehienek ez baitute partizio partzialak ebaluatzeko gaitasunik. *Cluster*-en baliozkotasuna neurtzeko indize berria COP da: *Context-independend Optimality and Partiality*. Izan ere, bi ezaugarri horiek betetzen ditu. Hori dela eta, SEP algoritmoarekin erabiltzeko egokia da, besteak ez bezala, haien bitartez ezin baita partizio partzialaren balioa neurtu, ezta momentuko partizio hoberena edozein egoeratan hoberena izango dela ziurtatu ere.

4.2.2 COP indizea

COP [Gurrutxaga, 2010] indizearen balioa *cluster* barneko bariantzaren edo intra-bariantzaren (kohesioa) eta *cluster* arteko bariantzaren edo inter-bariantzaren (urruntasuna)

estimazioan oinarritzen da. Honela definitzen da:

$$\mathrm{COP}(P^Y, X) = \frac{1}{|Y|} \sum_{C \in P^Y} |C| \frac{\mathrm{intra}_{COP}(C)}{\mathrm{inter}_{COP}(C)},$$

non

$$\mathrm{intra}_{COP}(C) = 1/|C| \sum_{x \in C} \mathrm{d}(x, \bar{C})$$

$$\mathrm{inter}_{COP}(C) = \min_{x_i \notin C} \max_{x_j \in C} \mathrm{d}(x_i, x_j)$$

eta $\mathrm{d}(x_i, x_j)$ x_i eta x_j punten arteko distantzia euklidearra den. Erroa eta hostoak ezin dituenez aztertu, 1 balioa esleitzen diogu kasu horietan. Hortaz, partizio partzial bat, P^Y, aztertzeko gai da, X datu-base osoa erabiliz, baina $X - Y$ zatiko *clustering*-ari buruz ezer jakin gabe.

Kohesioa edo intra-bariantza *cluster* bateko puntuen eta bere erdigunearen (zentroidearen) arteko distantzia euklidearraren batezbestekoa da. Urruntasuna edo inter-bariantza, aldiz, *cluster*-aren eta bere hurbileneko puntuaren arteko distantzia da, *complete linkage* bidez neurtua. *Cluster* bakoitza bere kabuz aztertzen da eta azterketa partzial guztietan bere pisuaren araberako batezbestekoa egiten da.

COPek beste ezaugarri interesgarri bat ere badauka: 0 eta 1 artean bornatuta dago. Bestalde, *cluster*-ekoa ez den puntu hurbilena *cluster*-eko zentroidean dagoenean ere balio maximoa hartzen du, baina kasu hau ez da ohikoa.

Bibliografian aurkitutakoaren arabera, *complete linkage* metrikarekin, emaitza hobeak lortzen dira aztertutako beste algoritmoekin baino. Batez ere, SEP eta COP batera erabiltzean, nahiz eta COP beste algoritmoentzako ere erabilgarria den.

4.2.3 Inplementazioa

Hasi baino lehen, aipatu behar da, izatez, SEP algoritmoa eta COP indizearen kalkuluak distantzia euklidearrarekin daudela azalduta. Gure kasuan, ordea, sekuentzien arteko distantziak (antzekotasunak) neurtzen ari garenez, ezinezkoa da distantzia euklidearra erabiltzea eta, ondorioz, Edit edo Levenshtein distantzia erabiltzen dugu. Beraz, formulak egokitu ditugu distantzia euklidearraren balioak erabili ordez, Edit distantziarenak erabiltzeko. Bestalde, *cluster*-etako zentroideak erabili ordez, medoideak erabili ditugu.

Dagoeneko kontzeptuak argi dauzkagula, SEP algoritmoa eta COP indizea inplementatzeko klase bana sortu dugu. Ikus ditzagun banaka:

COP indizea

COP indizearen balioa kalkulatzeko distantzien matrizea beharrezkoa denez, klaseak, bere baitan, distantzia-matrizea jasotzen du, beraz, eraikitzailean bertan pasatzen zaio.

Funtzio nagusiak, *computeCOP* metodoak (10 algoritmoa), *cluster*-multzo bat eta multzo horren dendrogramako gaineko nodoa (hots, *cluster* multzo hori elkartzen duen dendrogramako nodoa) pasata, *cluster* multzoaren COP balioa kalkulatu eta itzultzen du. Horretarako, aurreko atalean adierazitako formulak erabiltzen ditugu.

Algoritmoa 10 computeCOP($nodoZerrenda$)

```
1: medoideak ← computeMedoids(nodoZerrenda)
2: if nodoZerrenda-n erroa bakarrik dago then
3:    return 1
4: end if
5: for all nodoa ∈ nodoZerrenda do
6:    kasuKop ← kasuKop + nodoako kasu kopurua
7:    if nodoa hostoa da then
8:       return clusterCOP ← clusterCOP + 1
9:    else
10:      hostoak ← getLeafs(nodoa)
11:      for all hostoa ∈ hostoak do
12:         intraCOP ← intraCOP + distantzia(hostoa, medoideak(nodoa))
13:      end for
14:      intraCOP ← intraCOP/hosto kopurua
15:      besteHostoak ← restLeafs(nodoa)
16:      for all bhostoa ∈ besteHostoak do
17:         distantziaMax ← max(hostoak, bhostoa)
18:         if distantziaMax < interCOP then
19:            interCOP ← distantziaMax
20:         end if
21:      end for
22:      if interCOP ≠ 0 then
23:         clusterCOP ← clusterCOP + (nodoko kasu kopurua)∗(intraCOP/interCOP)
24:      end if
25:   end if
26: end for
27: COPValue ← clusterCOP/kasuKop
28: return COPValue
```

Algoritmoari parametro moduan *nodoZerrenda* pasa behar diogu. Aldagai horretan, COP balioa kalkulatzeko erabili beharreko partizioa adierazi behar da. Ondoren, *medoideak* aldagaian, *computeMedoids*() funtzioaren bitartez, *nodoZerrenda*ko *cluster*

bakoitzaren medoidea gordetzen da. Gero, kasu berezia aztertzen da: nodoen zerrendan pasatzen dugun *cluster* bakarra erroa bada, hau da, datu guztiak multzo bakar batean bildu nahi baditugu, COP balio txarrena, 1, itzulitzen dugu. Bestela, *nodoZerrenda*n dagoen *nodoa* nodo bakoitzeko, COP indizea kalkulatuko dugu, formulak dioen moduan, baina pausoz pauso. Horretarako, *kasuKop* aldagaian, partizioko hosto-kopurua jasotzen dugu, beraz, *nodoa* (*cluster*) bakoitzeko, bere hosto-kopurua gehitzen dugu. *clusterCOP* aldagaian, partizioko *cluster* guztien COP balioak batzen ditugu. Horregatik, aztertzen ari garen *nodoa* hostoa bada, *clusterCOP* aldagaiari 1 gehitzen diogu, COP indizearen baliorik txarrena, hostoen kasuan horrela zehaztuta baitago.

Bestela, *nodoa* ez bada hostoa, *hostoak* aldagaian, *nodoa*ri dagokion *cluster*-eko hostoak gordetzen ditugu, $getLeafs()$ funtzioari (ikus 12. algoritmoa) deituz. Ondoren, *hostoak* multzoan dagoen *hostoa* hosto bakoitzeko, *hostoa*ren eta *nodoa*ri dagokion *cluster*-eko medoidearen arteko distantzia kalkulatu eta *intraCOP* aldagaian batzen da, *cluster*-eko hosto guztiek medoidera duten distantziaren batura totala lortzen den arte. Jarraian, *intraCOP* (intra-bariantza) balioa *cluster*-eko hosto-kopuruarekin zatitzen da, *cluster*-eko hostoek medoidera duten distantziaren batezbestekoa lortzeko.

Ondoren, *besteHostoak* aldagaian, *nodoa cluster*-ean ez dauden hostoen multzoa lortzen da, $restLeafs()$ funtzioari (ikus 13 algoritmoa) egiten zaion deiaren bitartez. *besteHostoak* multzoan dagoen *bhostoa* hosto bakoitzeko, *nodoa*ri dagokion *cluster*-eko hosto guztien eta *bhostoa*ren arteko distantzia maximoa kalkulatzen da $max()$ funtzioari (ikus 11. algoritmoa) deituz eta *distantziaMax* aldagaian gordetzen da. *distantziaMax* *interCOP* (inter-bariantza) baino txikiagoa bada, *interCOP* aldagaia *distantziaMax* aldagaiaren balioarekin eguneratzen dugu. Modu honetan, *interCOP* aldagaian, distantzia urrunenen artean txikiena gordetzen dugu.

Gero, *interCOP* aldagaiaren balioa zeroren desberdina den begiratzen da. Baiezko kasuan, *clusterCOP* aldagaiari momentuko *cluster*-aren COP balioa gehitzen zaio, horretarako formula aplikatuz (*intraCOP* eta *interCOP* aldagaien arteko zatiketari, *nodoa*ko hosto kopurua biderkatzea).

Partizioko nodo (*nodoa*) guztiak aztertu ondoren, *COPValue* (partizioaren COP balioa gordetzen duena) aldagaian, *clusterCOP* balioa partizioko hosto-kopuruarekin (*kasuKop*) zatitzean lortzen den emaitza gordetzen da; bukaeran, balio hori itzultzen da.

Aurreko atalean erakutsitako algoritmotik gehien aldentzen den agindua azkeneko *if*-a da. Inter-bariantzaren balioa 0 bada, zatiketa egiterakoan akatsa ematen du programak. Hori dela eta, 0 ez den bitartean, algoritmoan adierazten den bezalaxe kalkulatuko dugu

COP balioa. Baina zer egin 0 denean? Kasua aztertuz gero, inter-bariantza 0 izateak bi nodoak elkarren gainean daudela adierazten du, bien arteko distantzia 0 baita. Ondorioz, bi nodo horiek oso antzekoak direnez, elkartzea nahi dugu. Beraz, COP balioa 0 izan behar du. Horregatik ez diogu ezer batzen. Egoera hau medoideak erabiltzeagatik ematen da.

10 algoritmoan ikusi dugunez, algoritmoak beste zenbait funtzio laguntzaile erabiltzen ditu. Alde batetik, *computeMedoids()* funtzioa daukagu; *cluster*-multzoa pasata, bakoitzaren medoidea kalkulatzeaz arduratzen dena. Ondoren, *array* batean itzultzen ditu medoideak, pasa zaizkion *cluster*-en hurrenkera berean; hau da, lehenengo *cluster*-aren medoidea itzultzen duen *array*-aren lehenengo posizioan dago. Funtzio hau aurretik inplementatuta zegoenez, ez dugu gehiago azaltzen.

Hurrengo hiru funtzioak, ordea, COP indizearen kalkulua egiteko inplementatu dira:

- *max*: *Cluster* bateko nodoen eta beste nodo baten arteko distantzia handiena ematen du, inter-bariantza kalkulatzeko. 11. algoritmoan ikus daiteke sasikodea.

Algoritmoa 11 $\max(hostoak, bhostoa)$

```
1: for all hostoa ∈ hostoak do
2:     distantzia ← distantzia(hostoa, bhostoa)
3:     if distantzia > max then
4:         max = distantzia
5:     end if
6: end for
7: return max
```

Funtzioari pasatzen zaizkion parametroetan, *hostoak* aldagaian, aztertu nahi dugun *cluster*-eko hostoak pasa behar zaizkio eta, *bhostoa* aldagaian, beste hostoa. Honela, *hostoak* multzoan dagoen *hostoa* hosto bakoitzeko, *hostoa*ren eta *bhostoa*ren arteko distantzia kalkulatzen dugu eta *distantzia* aldagaian gordetzen. *distantzia* momentura arte jaso dugun *max* distantzia handiena baino handiagoa bada, aldagaiak eguneratzen ditugu, *max* aldagaian *distantzia*ren balioa jasoz. *hostoak* multzoko hosto guztiak aztertzean, begiztatik atera eta *max* distantzia handiena itzultzen du.

- *getLeafs*: Nodo jakin bat emanik, horrek bere azpian hartzen dituen hostoak *array* batean bueltatzen ditu, hau da, nodo jakin horrek ordezkatzen duen partizioan dauden hostoak (sekuentziak) itzultzen ditu. 12. algoritmoan ikus daiteke sasikodea.

Algoritmoa 12 getLeafs(*dnodoa*)

```
1: nodoZerrenda ← dnodoa
2: for all nodoa ∈ nodoZerrenda do
3:    if nodoa hostoa da then
4:       hostoak ← hostoak ∪ nodoa
5:    else
6:       nodoZerrenda ← nodoZerrenda ∪ nodoaren eskuineko umea ∪ nodoaren ezker-
         reko umea
7:    end if
8: end for
9: return hostoak
```

Parametro moduan pasatzen diogun *dnodoa* nodoaren hostoak lortzeko, *nodoZerrenda* aldagaian, aztertu behar ditugun nodoak gordetzen ditugu. Hasteko, *dnodoa* bera sartzen dugu. *nodoZerrenda* multzoan dagoen *nodoa* nodo bakoitzeko, *nodoa* hostoa bada, *hostoak* multzoan gehitzen dugu, bertan gordetzen baitira *dnodoa*ren hostoak. *nodoa* ez bada hostoa, ordea, *nodoZerrenda* multzoan *nodoa*ren umeak sartzen ditugu, eskuinekoa eta ezkerrekoa. *nodoZerrenda* osorik aztertu dugunean, begiztatik ateratzen da eta *hostoak* multzoan *dnodoa*ren hostoak itzultzen ditu.

- *getRest*: Datu-base osotik, partizioan ez dauden hostoak lortzen ditu; aurreko funtzioaren aurkakoa, hain zuzen ere. 13. algoritmoan ikus daiteke sasikodea.

Algoritmoa 13 getRest(*dnodoa*)

```
1: nodoZerrenda ← dendrogramaren erroa
2: for all nodoa ∈ nodoZerrenda do
3:    if nodoa ≠ dnodoa then
4:       if nodoa hostoa da then
5:          besteHostoak ← besteHostoak ∪ nodoa
6:       else
7:          nodoZerrenda ← nodoZerrenda ∪ nodoaren eskuineko umea ∪ nodoaren
            ezkerreko umea
8:       end if
9:    end if
10: end for
11: return besteHostoak
```

dnodoa nodoari dagokion *cluster*-ekoak ez diren hostoak lortzeko, *nodoZerrenda* multzoan aztertu beharreko nodoak gordetzen ditugu. Zuhaitz osoa aztertu nahi dugunez,

dendrogramaren erroa gehitzen diogu hasieran. Ondoren, *nodoZerrenda*n dagoen *nodoa* nodo bakoitzeko, *nodoa* sarrera moduan jaso dugun *dnodoa*ren berdina den begiratzen dugu. Berdina bada, ez dugu ezer egiten. Horrela, nodo horretatik behera jaistea saihesten dugu. Esku artean dauzkagun bi nodoak desberdinak badira, *nodoa* hostoa den edo ez begiratzen dugu. Horren arabera, hostoa bada, *besteHostoak* aldagaian gehitzen dugu, bertan jasotzen baititugu pasatzen zaigun *cluster*-etik kanpo dauden hostoak. *nodoa* ez bada hostoa, *nodoZerrenda* aldagaian gehitzen ditugu bere eskuineko eta ezkerreko umea. Aztertu beharreko nodoak bukatzean, *besteHostoak* aldagaia itzultzen dugu.

SEP algoritmoa

SEP klaseko eraikitzaileari dendrograma eta distantzia-matrizea pasa behar zaizkio, azken hori COP klaseari pasatzeko, izan ere, COP indizearen objektua hemen sortuko da.

Klase honek bi metodo dauzka: alde batetik, *computeSEP()*, SEP algoritmoa exekutatzen duen funtzioa eta, bestetik, *getClustersFromNodes()*, *cluster*-ak identifikatzen dituzten nodoak emanda, *cluster* horien identifikatzaileak ematen dituen funtzioa. Azken funtzio hau, batez ere, algoritmoaren irteera sistemara egokitzeko da.

SEP algoritmoa exekutatzen duen funtzioari dagokionez (ikus 14. algoritmoa), nodo bat pasa behar zaio. Algoritmoak dioen moduan, pasatzen zaion nodoa hostoa bada, zuzenean hostoa itzultzen du; bestela, berriz, bi umeen elkarketaren COP balioa kalkulatzen du, horretarako COP klasearen instantzia erabiliz, eta nodoaren COP balioarekin konparatzen du. COP balio hoberena (baxuena) daukan partizioa itzultzen du funtzioak. Horregatik, lehenengo deian, erroa pasa behar zaio, errekurtsiboki umeak aztertuz, hostoetaraino hel dadin.

COP indizea sortu ondoren, programari parametro moduan pasatzen zaion *nodoa* hostoa bada, *nodoa* bera itzultzen dugu. Bestela, *elkarketa* aldagaian *nodoa*ren umeen partizio hoberenak gordetzen ditugu, horretarako SEP algoritmoari dei errekustsiboa eginez. *elkarketaCOP* aldagaian, *elkarketa*ren COP balioa jasotzen da, *computeCOP*() algoritmoari (ikus 10. algoritmoa) deituz. Modu berean, *nodoaCOP* aldagaian *nodoa*ren COP balioa gordetzen da. Bukatzeko, *nodoaCOP* balioa *elkarketaCOP* baino txikiagoa bada, *nodoa* bera itzuliko dugu; bestela, *elkarketa*.

Programa nagusia

Bukatzeko, SAHN algoritmoa *complete* aukerarekin erabiltzen den *main*-aren (ikus 3 algoritmoa) kopia bat egin dugu, SEP erabil dezan aldaketak sortzeko. *Complete* aukeratu

Algoritmoa 14 computeSEP(*nodoa*)

```
1: COP indizea sortu
2: if nodoa hostoa da then
3:    return nodoa
4: else
5:    elkarketa ← elkarketa ∪ computeSEP(nodoa-ren ezkerreko umea) ∪
      computeSEP(nodoa-ren eskuineko umea)
6:    elkarketaCOP ← computeCOP(elkarketa)
7:    nodoaCOP ← computeCOP(nodoa)
8:    if nodoaCOP < elkarketaCOP then
9:       return nodoa
10:   else
11:      return elkarketa
12:   end if
13: end if
```

dugu SEP algoritmoa azaltzen den artikuluan emaitzarik onenak metrika honekin ematen dituela adierazten delako [Gurrutxaga, 2010]. Gainera, *k* balioak ez dira behar dagoeneko. *Clustering*-aren atalean, dendrograma sortu ondoren, hori mozteko beharrezko funtzio berriak sortu ditugu, sistemaren egiturari jarraituz.

cutDendrogramWithSEP() funtzioak SEP klaseko aldagai bat sortzen du, dendrograma eta distantzia-matrizearekin, eta SEP algoritmoa exekutatzen duen funtzioari deitzen dio. Ondoren, SEP exekutatzen duen funtzioak itzulitako nodoen zerrenda pasatzen zaio, *getClustersFromNodes()* funtzioari deituz, *cluster*-en identifikatzaileen zerrenda lortzeko.

Azkeneko aldaketa ebaluazioaren atalean egin dugu. Bertan, *k* bakoitzerako emaitzak lortzeko begizta kendu dugu eta behin bakarrik exekuta dezan (partizio hoberenarekin) moldatu dugu.

Ikus dezagun zer gertatzen den probetan, aurreko datu-basean SEP algoritmoa erabiltzen badugu.

4.2.4 Probak

Sistemak, orain arte azaldu dugun bezala martxan jarrita, ez du espero bezala funtzionatu. Izan ere, beti bi *cluster*-etan multzokatuta itzultzen ditu sekuentziak; bat itzultzea ezinezkoa delako, ziurrenik, erroaren COP balioa bat baita.

Arazoak

Nondik etor litezke arazoak? Zergatik ez du algoritmoak behar bezala funtzionatzen? Alde batetik, distantzia euklidearra erabili ordez, *Edit* distantzia erabiltzeak eragina izan dezake, izan ere, artikuluan algoritmoa azaltzen denean, distantzia euklidearrarekin dago azalduta eta ez du beste distantziei buruz ezer esaten. Bestalde, zentroideak ere ez ditugu erabiltzen, medoideak baizik. Agian aldaketa gehiegi izan daitezke algoritmoaren funtzionamendu egokia bermatzeko.

Kodea behin eta berriz goitik behera begiratu eta, hala ere, arazorik ez antzematean, SEP algoritmo sekuentziala (ikus 15 algoritmoa) inplementatu dugu, programaren igarotzea errazago jarraitu ahal izateko.

Aldagai asko daudenez, algoritmoa ulertu ahal izateko, bakoitzak zer jasotzen duen jakitea beharrezkoa da. Jarraian nahasgarrienak izan daitezkeenak aipatzen ditugu:

- *nodoID*: nodoen zerrenda bat, aztertzen goazen heinean sortzen duguna. Zerrendaren helburua gainontzeko zerrendetako posizioak nodo batekin lotzea da. Adibidez, zerrenda honetan bostgarren lekuan dagoen nodoaren informazioa bilatzeko, gainontzeko zerrendetako bostgarren lekuan dauden datuak hartu behar ditugu.
- *partizioOnena*: nodo bakoitzari dagokion partiziorik onena gordetzen du; nodoa bera edo azpiagoko elkarketa bat jaso dezake.
- *elkarketenCOP*: nodo bakoitzarentzat, azpiko elkarketatik lortzen den COP balioa gordetzen du.
- *nodoenCOP*: nodo bakoitzarentzat, nodoak berak ematen duen COP balioa gordetzen du, hau da, azpiko hosto guztiak *cluster* bakarrean elkartzearen COP balioa.
- *nodoCOP*: momentuko nodoaren COP balioa, azpiko hosto guztiak batera biltzean.
- *elkarketaCOP*: momentuko nodoaren azpiko elkarketaren COP balioa.

Algoritmo sekuentzialari esker, balio guztiak gordeta gelditzen dira eta edozein momentutan kontsulta ditzakegu, egoera aztertu eta arazorik badagoen ikusteko. Igarotzea pausoz pauso jarraituta, honakoak ondorioztatu ditugu:

- COP balioa beti 0 eta 1 artean dago. Gainera, batzuetan nodoaren COP balioa da altuagoa eta besteetan, berriz, elkarketarena. Beraz, COP balioa zentzuzkoa da.
- SEP algoritmoa aplikatzen den hurrenkera, hau da, nodoak aztertzen diren hurrenkera, esperotakoa da, azken mailako ezkerreko hostoarekin hasten baita eta bidea behar bezala egiten baitu.

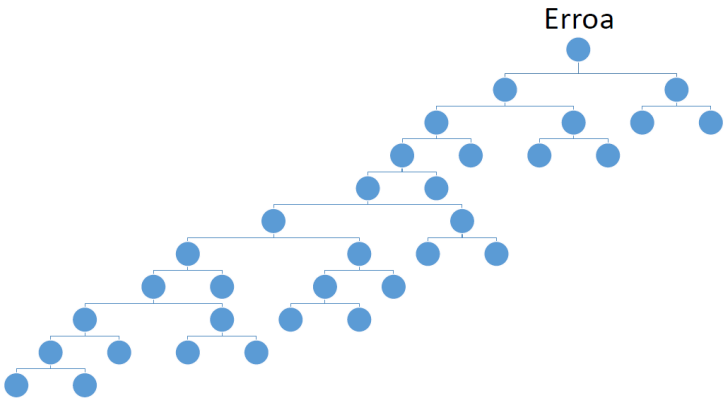
Algoritmoa 15 computeSEPSeq()

```
1: nodoZerrenda ← dendrogramarenerroa
2: for all nodoa ∈ nodoZerrenda do
3:    if nodoa ez da hostoa then
4:       nodoZerrenda ← nodoZerrenda ∪nodoaren eskuineko umea ∪ nodoaren ezkerreko umea
5:    end if
6: end for
7: COP indizea sortu
8: for all nodoa ∈ nodoZerrenda do
9:    if nodoa hostoa da then
10:      nodoID ← nodoID ∪ nodoa
11:      partizioOnena ← partizioOnena ∪ nodoa
12:      elkarketaCOP = 1
13:      nodoCOP = 1
14:      elkarketenCOP ← elkarketenCOP ∪ elkarketaCOP
15:      nodoenCOP ← nodoenCOP ∪ nodoCOP
16:   else
17:      left ← nodoa − renezkerumea
18:      indexLeft ← nodoID.lortuIndizea(left)
19:      elkarketa ← partizioOnena(indexLeft)
20:      right ← nodoa − reneskuinumea
21:      indexRight ← nodoID.lortuIndizea(right)
22:      elkarketa ← elkarketa ∪ partizioOnena(indexRight)
23:      nodoCOP ← computeCOP(nodoa)
24:      elkarketaCOP ← computeCOP(elkarketa)
25:      nodoID ← nodoID ∪ nodoa
26:      elkarketenCOP ← elkarketenCOP ∪ elkarketaCOP
27:      nodoenCOP ← nodoenCOP ∪ nodoCOP
28:      if nodoCOP < elkarketaCOP then
29:         partizioOnena ← partizioOnena ∪ elkarketa
30:      else
31:         partizioOnena ← partizioOnena ∪ nodoa
32:      end if
33:   end if
34: end for
```

- Sistemak beti bi *cluster* itzultzen dituela ikusita, elkarketak ez direla behar bezala tratatzen pentsatu dugu. Hala ere, exekuzioa pausoka ikustean, ondo igarotzen direla ikusi dugu, baina, errora hurbiltzen goazen heinean, COP balioa beti hobea da denak *cluster* bakarrean elkartuta, elkarketa mantentzea baino. Hori dela eta, momentura arteko elkarketa galtzen da eta azken bi nodoak itzultzen ditu.

Zergatik gertatzen den ezin asmatuz, dendrogramako nodoen *elkarketaCOP* eta *nodoCOP* balioak aztertu ditugu. Errora hurbiltzen goazen heinean, *nodoCOP* balioak txikitzen doaz. *elkarketaCOP*, berriz, hostoetatik igotzean, handitzen hasten da, baina erroaren inguruan gabiltzanean, txikitzen hasten da. Edonola ere, hostoen COP balioa 1 denez, hostoen gaineko nodoan gaudenean, elkarketaren COP balioa ere 1 da. Hortaz, hostoak inguruan dauden kasuetan, *elkarketaCOP* balioa handitzen da.

Horregatik, dendrogramaren egitura aztertzea pentsatu dugu, arazoa hor egon daitekeela eta. 4.8 irudian BTw datu-basearekin sortutako dendrograma baten goiko zatiaren errepresentazioa erakusten dugu, eredu moduan erabili duguna. Ikusten denez, errotik oso hurbil oraindik hostoak agertzen dira eta horrek elkarketaren COP baliogabetzen digu.



Irudia 4.8: Dendrogramaren goiko zatia.

Ondorio honetara iritsi ondoren, berriro ere artikuluari begirada bat eman diogu. Bertan egiten diren probetan *cluster* kopurua beti 10 edo txikiagoa dela ikusi dugu. Hortaz, proba horiekin ezin dugu bermatu gure kasuan ongi lan egingo duenik, gure datu-basean ziurrenik *cluster* kopuru altuagoa baitaukagu.

Hurrengo atalean arazoari nola aurre egin diogun azaltzen dugu.

Egokitzapena

Arrazoi hori aurkitu dugula, algoritmoa zertxobait egokitzea pentsatu dugu arazoa saihesteko. 16 algoritmoan ikus daitekeen moduan, oraingoan, elkarketaren COP balioa artikuluan [Gurrutxaga, 2010] adierazten den bigarren moduan kalkultu dugu, bertako hosto-kopurua lortu ahal izateko. Gainera, elkarketaren COP balioari biderkatzaile bat

jarri diogu, zenbat eta hosto gehiago egon elkarketan (hots, dendrograman zenbat eta gorago egon), orduan eta txikiagoa da biderkatzailea, eta, ondorioz, orduan eta txikiagoa elkarketaren COP balioa. Izan ere, proportzioa kalkulatzen dugu, elkarketako hosto-kopurua dendrogramako hosto-kopuruarekin zatituz. Elkarketako hosto-kopurua zenbat eta altuagoa izan, ordea, proportzioa orduan eta altuagoa izango da eta guk, biderkatzailea txikiagoa izatea nahi dugu. Beraz, biderkatzailea $1 - proportzioa$ eginez lortu dugu.

Orain bai, ikus ditzagun aldaera hau exekutatuta lortzen ditugun emaitzak.

Emaitzak

Atal honetan, BTw-ko datu-basearekin SEP algoritmoa martxan jarrita lortu ditugun emaitzak erakusten ditugu.

Sistemak, batezbestean, 443 *cluster* sortzen ditu exekuzio bakoitzean. 4.9 irudian, balidazioko 10 exekuzioetan lortutako emaitzen batezbestekoak adierazten dira (*cluster*-kopurua ere batezbestekoa da).

SEP					
Cluster kopurua	Proposamen kopurua	F-measure %25	F-measure %50	MF-measure %25	MF-measure %50
443	4	0.20709915	0.17871068	0.29066983	0.29962885

Irudia 4.9: SEP algoritmoa eta COP indizea konbinatuz lortutako emaitzak.

Argi ikus daiteke emaitzak nahiko onak direla, nahiz eta PAM-enak baino zertxobait kaxkarragoak diren. Proiektuarekin lanean jarraitu ahala, emaitza horiek hobetzen saiatu gara eta, SEP-en abantailak aprobetxatuz, beste ingurune batean hobea den edo ez probatu dugu.

4.2.5 Ondorioak

SEP algoritmoak ez digu artikuluan definitzen den bezala balio, baina, egindako aldaketa txiki horrekin, zentzuzko emaitzak lortu ditugu. Distantzia euklidearraren ordez *Edit* distantzia eta zentroideen ordez medoideak erabiltzeak espero genuena baino eragin handiago izan du, eta, hori konpontzeko, algoritmoan biderkatzaile bat sartzea beharrezkoa izan da.

Ikusi dugunez, oraingoan ere, emaitzak ez dira PAM-enak bezain onak, baina oso gutxirengatik. Kontuan izan behar da, ordea, SEP algoritmoak, abantaila handiak es-

Algoritmoa 16 computeSEPSeq()

```
1: nodoZerrenda ← dendrogramarenerroa
2: for all nodoa ∈ nodoZerrenda do
3:    if nodoa ez da hostoa then
4:       nodoZerrenda ← nodoZerrenda ∪ nodoa − reneskuinekoumea ∪ nodoa −
         renezkerrekoumea
5:    end if
6: end for
7: COP indizea sortu
8: for all nodoa ∈ nodoZerrenda do
9:    if nodoa hostoa da then
10:      nodoID ← nodoID ∪ nodoa
11:      partizioOnena ← partizioOnena ∪ nodoa
12:      elkarketenCOP ← elkarketenCOP ∪ 1
13:      nodoenCOP ← nodoenCOP ∪ 1
14:   else
15:      left ← nodoa − renezkerumea
16:      indexLeft ← nodoID.lortuIndizea(left)
17:      elkarketa ← partizioOnena(indexLeft)
18:      right ← nodoa − reneskuinumea
19:      indexRight ← nodoID.lortuIndizea(right)
20:      elkarketa ← elkarketa ∪ partizioOnena(indexRight)
21:      for all cluster ∈ elkarketa do
22:         indexCluster ← nodoID.lortuIndizea(cluster)
23:         elkarketaCOP ← elkarketaCOP + hostoKopurua(cluster) ∗
            nodoCOP(indexCluster)
24:         kasuak ← kasuak + hostoKopurua(cluster)
25:      end for
26:      elkarketaCOP ← elkarketaCOP/kasuak
27:      proportzioa ← kasuak/hostoKopurua(dendrogramarenerroa)
28:      elkarketaCOP ← elkarketaCOP ∗ (1 − proportzioa)
29:      nodoCOP ← computeCOP(nodoa)
30:      nodoID ← nodoID ∪ nodoa
31:      elkarketenCOP ← elkarketenCOP ∪ elkarketaCOP
32:      nodoenCOP ← nodoenCOP ∪ nodoCOP
33:      if nodoCOP < elkarketaCOP then
34:         partizioOnena ← partizioOnena ∪ elkarketa
35:      else
36:         partizioOnena ← partizioOnena ∪ nodoa
37:      end if
38:   end if
39: end for
```

kaintzen dizkigula, izan ere, ez dugu k parametroa doitzeko beharrik izan eta, hala ere, lortutako emaitzak oso parekoak dira.

Hori dela eta, PAM-ekin garatutako sistema baino aukera hobea izan daiteke sistema beste ingurune batzuetara aplikatzen dugunean.

Jarraian agertzen den 4.10 irudian, PAM algoritmoarekin eta SAHN algoritmoaren bertsioekin lortutako emaitza hoberenak SEP algoritmoarekin lortutakoekin konparatzen dira. PAM eta SAHN algoritmoekin inplementatutako bertsioen kasuan, balidazioko emaitzak bakarrik hartu ditugunez kontuan, SEP-ekin balidazioan daukagun emaitza bakarra erakusten dugu. Hemendik aurrera ere, irizpide berari jarraitu diogu konparazioa egitean.

Bertsioa	F-measure %25	F-measure %50	MF-measure %25	MF-measure %50
PAM	0.22991998	0.18031698	0.31587023	0.3044352
Average	**0.23346174**	**0.19218656**	**0.31990647**	**0.31990647**
Complete	0.23005779	0.19087751	0.31629854	0.31496984
Ward	0.21812049	0.177389	0.3031231	0.29823348
SEP	0.20709915	0.17871068	0.29066983	0.29962885

Irudia 4.10: Orain arteko sistemaren bertsio bakoitzarekin lortutako emaitza hoberenen konparazioa.

Emaitza txarrenak SEP-ek lortzen ditu *Ward* bertsioarekin batera, baina kontuan izatekoa da horiek emaitza hoberenak direla, eta ez batezbestekoak. Ikus dezagun zer gertatzen den proiektuan aurrera egin ahala.

5. kapitulua

Profil antzekoena bilatzeko atalean aldaketak

Aurretik aipatu dugun bezala, aztertzen ari garen sekuentziaren profil antzekoena bilatzeko K-NN algoritmoa erabiltzen da. *Clustering*-a egin ondoren, lortzen dugun multzo bakoitzeko medoidea kalkulatzen da, nolabait esateko, erdigunea dena; hau da, beste guztietatik distantzia txikiena duen sekuentzia. Distantzia esaten dugunean, antzekotasuna esan nahi dugu. Medoidea dagoen *cluster* jakin horretako ordezkaria izango da. Hortaz, erabiltzaile berri baten sekuentzia aztertzen hasten garenean, *cluster* bakoitzeko medoidearekin konparatzen da, hurbilen daukana (antzekoena) zein den kalkulatzeko eta, ondoren, *cluster* horretako proposamenak egiten zaizkio. Prozesu hau guztia aztertzen jarrita, honako hobekuntzarako aukerak aurkitu ditugu:

- Aztertzen ari garen sekuentziaren hurbileneko *cluster*-a bilatzen ari garenean, medoidearekin egiten den konparazioa ez da oso egokia, aztertzen ari garen sekuentziaren %25 edo %50 bakarrik hartzen baitugu, medoide osoarekin konparatzeko. Beraz, tamainan daukaten diferentzia da bien arteko distantzia minimoa, *Edit* distantzia erabiltzen dugunez, egin beharreko aldaketa bakoitzak unitateko kostua baitauka.
- Bi sekuentzien arteko distantziak aztertzean, sistemak distantzia absolutuekin egiten duela lan konturatu gara. *Cluster* hurbilena distantzia absolutuarekin bilatzea, ordea, ez da oso zuzena, batez ere sekuentzia medoide osoarekin konparatzen badugu. Izan ere, zenbat eta luzeagoa izan medoidea, orduan eta handiagoa izango da distantzia, nahiz eta hasierako zatia bietan bera izan. Nola nahi ere, aldaketa honen garrantzia txikiagoa da aurretik medoidearen luzera mozten badugu.
- Proposamenak egiteko, printzipioz, *cluster* bakarra erabiltzen da. Multzo horretan

4 proposamen baino gutxiago badaude, orduan hurrengo hurbileneko *cluster*-era jotzen du, harik eta 4 proposamen desberdin lortzen dituen arte edo *cluster* guztiak aztertzen dituen arte. Hala ere, medoide hurbilena izateak ez du esan nahi multzo horretako URLak direnik egokienak proposatzeko, bigarren hurbileneko *cluster*-ean *support* hobea daukan URL-ren bat egon baitaiteke. Hau da, gerta daiteke erabiltzaile berriak profil bat baino gehiagorekin izatea antzekotasuna eta, ondorioz, egokiena haien irteera konbinatzea izatea.

5.1 Medoidearen zati batekin konparatu

Lehenengo arazoa dela eta, konparazioa egiteko garaian, aztertu beharreko sekuentzia medoidearen hasierako zatiarekin bakarrik konparatzea pentsatu dugu. Aukera bat bi sekuentzietatik tamaina bera hartzea litzateke; hau da, medoidearen hasierako zatia hartzean, aztertzen ari garen sekuentziatik hartu dugun zatiaren tamaina kontutan hartzea. Hala ere, oso zorrotza dela iruditzen zaigunez, medoidetik aztertzen ari garen sekuentziaren URL kopurua baino bi gehiago hartzea erabaki dugu; horrela, URL berak topatzeko malgutasuna ematen diogulakoan.

5.1.1 Inplementazioa

Medoidea eta aztertzen ari garen sekuentzia aztertzen dituen jatorrizko funtzioaren sasi-kodea 17 algoritmoan ikus daiteke.

Algoritmoa 17 KnnSim()

```
1: for all medoidea ∈ medoideak do
2:     distantzia ← getScore(egindakoZatia, medoidea)
3:     distantziak ← distantziak ∪ distantzia
4: end for
5: distOrdenatuak ← orderSim(distantziak)
6: emaitza ← ordenatuMedoideak(distantziak, distOrdenatuak)
7: return emaitza
```

Algoritmoan, *medoideak* multzoan aztertzen ari garen partizioko *cluster*-etako medoideak dauzkagu gordeta. *medoideak* multzoan dagoen *medoidea* osagai bakoitzeko, *getScore*() funtzioari deituz, orain arte sekuentziatik hartu dugun zatiaren (*egindakoZatia*) eta *medoidea*ren arteko distantzia gordetzen dugu *distantzia* aldagaian. Jarraian, *distantziak* multzoan, *distantzia* gordetzen dugu. Horrela, begizta bukatzean, *distantziak* aldagaian sekuentziatik aztertu dugun zatiaren eta partizioko

medoide guztien arteko distantziak dauzkagu gordeta. Ondoren, *distOrdenatuak* aldagaian, *distantziak* aldagaiko balioak ordenatzen ditugu *orderSim*() funtzioari esker. Distantziak txikienetik handienera ordenatzen dira. Bukatzeko, *emaitza* aldagaia itzuli aurretik, bertan medoideak eta distantziak ordenatuta sartzen ditugu *distOrdenatuak* aldagaiko balioen arabera, *ordenatuMedoideak*() funtzioari esker.

Aldaketa txiki hau inplementatzeko nahikoa izan da kode lerro batzuk gehitzea. Medoide bat eskura izanik, beste aldagai batean interesatzen zaigun zatia gordetzen dugu. Tamaina erabakitzeko aztertu beharreko sekuentziaren luzera begiratu besterik ez da egin behar. Testeko sekuentziaren luzera (Lt) medoidearen luzera (Lm) baino handiagoa bada, medoide osoa pasatzen diogu distantzia kalkulatzen duen funtzioari. Lm Lt baino bat edo bi handiagoa bada, modu berean, medoide osoa pasatzen diogu. Aldiz, Lm oraindik luzeagoa bada, Lt+2 tamainako zatia bakarrik pasatzen diogu distantziak kalkulatzen dituen funtzioari.

Proposamenak lortzen dituen funtziotik KnnSim() funtzioari deitzen zaionean, bertsio berria (18. algoritmoa) jarri dugu.

Algoritmoa 18 KnnSimShort()

```
1:  for all medoidea ∈ medoideak do
2:      ml ← medoidearenluzera
3:      sl ← sekuentziarenluzera
4:      if ml ≤ sl then
5:          medoideMotza ← medoidea
6:      else
7:          if ml = sl + 1 then
8:              medoideMotza ← medoidea
9:          else
10:             medoideMotza ← medoideZatia(medoidea, sl + 2)
11:         end if
12:     end if
13:     distantzia ← getScore(egindakoZatia, medoideMotza)
14:     distantziak ← distantziak ∪ distantzia
15: end for
16: distOrdenatuak ← orderSim(distantziak)
17: emaitza ← ordenatuMedoideak(distantziak, distOrdenatuak)
18: return emaitza
```

Aurretik azaldu dugun bezala, *ml* aldagaian medoidearen luzera gordetzen dugu; *sl* aldagaian, berriz, aztertzen ari garen sekuentzia zatiaren luzera. Medoidearen luzera txikiagoa bada, *medoideMotza* aldagaian medoide osoa sartzen dugu, baita medoidea uni-

tate bakarrean luzeagoa bada ere. Bestela, hau da, medoidearen luzera bi unitate edo gehiago luzeagoa bada, *medoideMotza* aldagaian *medoidea* aldagaiaren zati bat bakarrik sartzen dugu, sekuentziaren luzera baino bi unitate handiagoko zatia, *medoideZatia*() funtzioak itzultzen diguna, medoidea eta hartu nahi dugun luzera adieraziz. Ondoren, algoritmoak berdin funtzionatzen du, *distantzia* kalkulatzeko *medoidea* erabili beharrean *medoideMotza* aldagaia erabiltzen dugun desberdintasunarekin.

Finean, *medoideMotza* aldagaian, medoidearen interesatzen zaigun zatia gordetzen dugu, nahiz eta batzuetan medoide osoa den, eta, ondoren, horrekin konparatzen dugu aztertzen ari garen sekuentziaren zatia (*egindakoZatia* aldagaia).

5.1.2 Probak

Hurrengo taulan, sistemak *clustering*-erako PAM algoritmoarekin, medoide motzagoak erabiliz lortutako emaitzak agertzen dira. Balidazio gurutzatuko exekuzio bakoitzaren emaitzak erakustea alferrikakoa iruditu zaigu, hortaz, k bakoitzarentzat lortutako batezbestekoak bakarrik adierazi ditugu.

PAM					
Cluster kopurua	Proposamen kopurua	F-measure %25	F-measure %50	MF-measure %25	MF-measure %50
50	4.0	**0.27934477**	**0.24831896**	0.41473898	0.48843074
100	4.0	0.27385104	0.2468636	0.41267672	0.49876204
150	4.0	0.27327055	0.24752752	0.4171067	0.50726473
200	4.0	0.273642	0.2474021	0.4173048	0.5076387
300	4.0	0.2723604	0.24806139	0.42076984	0.51658136
400	4.0	0.27129188	0.24780934	0.42073217	0.5177012
500	4.0	0.26982212	0.24817479	**0.4211672**	**0.52136505**

Irudia 5.1: PAM algoritmoarekin lortutako emaitzak, konparazioan medoide motzagoa erabilita.

3.4 atalean ikusitako emaitzen aldean, hobekuntza dezente lortu da.

Orain, *SAHN* algoritmoa *average* eta *complete* bertsioekin inplementatuta lortutako emaitzen taulak ageri dira, 5.2 eta 5.3, hurrenez hurren.

Average					
Cluster kopurua	Proposamen kopurua	F-measure %25	F-measure %50	MF-measure %25	MF-measure %50
50	4.0	0.18150736	0.1677392	0.2485631	0.27970502
100	4.0	0.21321996	0.18957081	0.30004284	0.31999046
150	4.0	0.219256	0.18814717	0.2996555	0.31124133
200	3.8	0.21439087	0.19308503	0.30016667	0.3224442
300	4.0	0.1946098	0.17497513	0.27544075	0.29813293
400	4.0	0.21969926	0.1961689	0.3053303	0.32863227
500	4.0	**0.22604239**	**0.20782855**	**0.3121498**	**0.34997678**

Irudia 5.2: *Average* bertsioarekin lortutako emaitzak, konparazioan medoide motzagoa erabilita.

Complete					
Cluster kopurua	Proposamen kopurua	F-measure %25	F-measure %50	MF-measure %25	MF-measure %50
50	4.0	0.17800745	0.17722598	0.24536243	0.29970208
100	4.0	0.21515389	0.19254532	0.29847884	0.32335705
150	4.0	0.21974	0.1960207	0.30963424	0.3341135
200	4.0	0.21809335	0.1984373	0.3059081	0.33864814
300	4.0	0.22056898	0.20139761	0.3124647	0.34446853
400	4.0	0.22588484	0.20123944	0.3153134	0.33966392
500	4.0	**0.22731355**	**0.2089489**	**0.31916255**	**0.35824072**

Irudia 5.3: *Complete* bertsioarekin lortutako emaitzak, konparazioan medoide motzagoa erabilita.

Ikus dezakegunez, bi kasu horietan, emaitzak kaxkartu egin dira.

Jarraian, *clustering*-ean *SAHN* algoritmoaren *ward* metrikadun bertsioa erabiliz lortutako emaitzak ageri dira 5.4 irudian; hasiera batean, txarrenak zirenak.

Ward					
Cluster kopurua	Proposamen kopurua	F-measure %25	F-measure %50	MF-measure %25	MF-measure %50
50	4.0	0.26196486	0.23436368	0.38117725	0.43886834
100	4.0	0.25844422	0.23911825	0.38230738	0.45737752
150	4.0	0.25555953	0.23702839	0.38069254	0.46225405
200	4.0	0.25169858	0.2383879	0.3759684	0.4716359
300	4.0	0.25718743	0.23918077	0.3943262	0.48363551
400	4.0	0.2547887	0.24022207	0.39010552	0.49048358
500	4.0	**0.26216775**	**0.24160476**	**0.40477234**	**0.4977159**

Irudia 5.4: *Ward* bertsioarekin lortutako emaitzak, konparazioan medoide motzagoa erabilita.

Oraingoan, aldiz, hobekuntza nabarmena izan da, batez ere, jatorrizko sistemarekin *ward* bertsioak emaitza txarrenak ematen zituela kontuan hartuta. Nolanahi ere, emaitzak ez dira PAM-enak bezain onak izatera iristen.

Azkenik, SEP algoritmoarekin lortutako emaitzak ikus ditzakegu 5.5 irudian:

SEP					
Cluster kopurua	Proposamen kopurua	F-measure %25	F-measure %50	MF-measure %25	MF-measure %50
433	4.0	0.26362854	0.23712198	0.40095013	0.4776237

Irudia 5.5: SEP algoritmoarekin lortutako emaitzak, konparazioan medoide motzagoa erabilita.

Kasu honetan, SEP algoritmoak balidaziorako eta testerako emaitza bakarra (batezbesteko bakarra) itzultzen duenez eta ez k bakoitzeko bat, ebaluazio-neurri bakoitzeko balio bakarra erakus dezakegu. Emaitzei dagokienez, PAM-enak baino kaxkarxeagoak izaten jarraitzen dute, baina SAHN-en edozein bertsio baino hobeak dira.

5.1.3 Ondorioak

Proben emaitzetan ikusi ahal izan dugun moduan, aldaketa txiki honekin hobekuntza nabarmenak lortu ditugu. Medoide osoa testeko sekuentziaren zati txiki batekin konparatzean, distantziak asko handitzen dira, eta, ondorioz, hurbileneko *cluster*-aren bilaketan eragina dauka.

Lehenengo, sistemaren bertsio bakoitzak, medoideak moztuta, lortu dituen emaitza hoberenak konparatu ditugu 5.6 irudian.

Bertsioa	F-measure %25	F-measure %50	MF-measure %25	MF-measure %50
PAM	**0.27934477**	**0.24831896**	**0.4211672**	**0.52136505**
Average	0.22604239	0.20782855	0.3121498	0.34997678
Complete	0.22731355	0.2089489	0.31916255	0.35824072
Ward	0.26216775	0.24160476	0.40477234	0.4977159
SEP	0.26362854	0.23712198	0.40095013	0.4776237

Irudia 5.6: Sistemaren bertsio bakoitzarekin lortutako emaitza hoberenak, medoideak moztuta.

Ondoren, sistemaren bi bertsio hoberenek (PAM eta SEP) emandako emaitzak aldaketarik egin gabe lortutako emaitzekin konparatu ditugu 5.7 irudiko taulan. Emaitzak hobeto ulertzeko, ezker zutabean adierazten diren zenbakiek zenbatgarren probaren emaitzak

diren adierazten dute; hau da, "1" lerroan dauden emaitzak orain arte lortutako lehen emaitzak dira, beraz, kasu honetan, sistemari aldaketarik egin gabe lortutakoak. "2" lerroko emaitzak, berriz, medoidea moztuta lortutako emaitzak dira.

Bertsioa	F-measure %25	F-measure %50	MF-measure %25	MF-measure %50
PAM				
1	0.22991998	0.18031698	0.31587023	0.3044352
2	**0.27934477**	**0.24831896**	**0.4211672**	**0.52136505**
SEP				
1	0.20709915	0.17871068	0.29066983	0.29962885
2	**0.26362854**	**0.23712198**	**0.40095013**	**0.4776237**

Irudia 5.7: PAM eta SEP algoritmoekin inplementatutako bertsioen orain arteko emaitzen konparazioa.

Oraindik, emaitza hoberenak PAM algoritmoarekin lortzen dira, baina SEP-ek hobekuntza handia egin du; bigarren bertsio onena izatera iritsi da.

Segidan, distantzia absolutuaren arazoa konpontzeari ekin diogu.

5.2 Distantzia erlatiboa

Bigarren arazoari dagokionez, distantzia absolutuaren eragina aztertuta, distantzia erlatiboa inplementatzea erabaki dugu. Modu honetan, medoidearen luzerak ez du eraginik *cluster* hurbilena bilatzerakoan; lortutako distantzia absolutua medoidearen luzerarekin edo aztertu beharreko sekuentziaren luzerarekin (bietatik luzeena denarekin) zatitzen baita. Horrela, distantziaren balioa 0 eta 1 artean mugatuta dago beti eta, horrela bai, balio txikiena duen *cluster*-a da sekuentziatik hurbilen dagoena.

5.2.1 Inplementazioa

Aztertzen ari garen sekuentziaren eta medoidearen zatiaren arteko distantzia kalkulatu ondoren, distantzien bektorean gorde baino lehen, bi sekuentzietako luzeenaren luzerarekin zatitzen da. Gainontzeko guztia (distantzien bektorea ordenatzea, hurbilena aukeratzea...) lehen bezalaxe egiten da, orain, balioak 0-1 artean mugatuta dauden diferentziarekin.

Jarraian (19. algoritmoa), medoide motzarekin kalkuluak egiteko sortutako funtzioa egokitu dugu, distantzia normaliza dezan.

Algoritmoa 19 KnnSimShortNorm()

```
for all medoidea ∈ medoideak do
  ml ← medoidearenluzera
  sl ← sekuentziarenluzera
  if ml ≤ sl then
    medoideMotza ← medoidea
    max ← sl
  else
    if ml = sl + 1 then
      medoideMotza ← medoidea
      max ← ml
    else
      medoideMotza ← medoideZatia(medoidea, sl + 2)
      max ← sl + 2
    end if
  end if
  distantzia ← getScore(egindakoZatia, medoideMotza)
  distantziak ← distantziak ∪ distantzia/max
end for
distOrdenatuak ← orderSim(distantziak)
emaitza ← ordenatuMedoideak(distantziak, distOrdenatuak)
return emaitza
```

Kasu honetan aldatzen den gauza bakarrak *max* aldagaiaren agerpena da. Hasieran, *medoideMotza* aukeratzen dugun bitartean, kasuaren arabera, *max* aldagaiari balio bat edo beste ematen diogu, hots, sekuentzia luzeagoa den kasuan, *max* aldagaian sekuentziaren luzera sartzen dugu; medoidea sekuentzia baino unitate bat luzeagoa bada, medoidearen luzera, eta, bestela, sekuentziaren luzera bi unitate gehituta, beste modu batera esanda, medoidetik hartzen dugun zatiaren luzera. Amaitzeko, *distantzia* aldagaia eguneratu ondoren, *distantziak* multzoan sartu baino lehenago, *max* aldagaiarekin zatitzen dugu. Horrela, aztertzen ari garen sekuentziaren eta medoidearen arteko distantzia erlatiboa lortzen dugu, gehienezko distantziak eraginik izan ez dezan.

5.2.2 Probak

Ikus ditzagun bada, medoide motzagoak erabiltzen dituen sistemaren bertsioaren gainean distantzia erlatiboa inplementatuta sistemak ematen dizkigun emaitzak. Lehenik PAM algoritmoarekin lortutako emaitzak ikus ditzakegu 5.8 taulan.

Aldaketa honekin ere, emaitzak hobetzea lortu dugu, nahiz eta ez den medoideak

PAM					
Cluster kopurua	Proposamen kopurua	F-measure %25	F-measure %50	MF-measure %25	MF-measure %50
50	4.0	0.2919012	0.25981882	0.43804637	0.51930237
100	4.0	**0.29426676**	**0.26344365**	**0.44960847**	0.54135144
150	4.0	0.28951985	0.26127294	0.44742268	0.54398805
200	4.0	0.29046026	0.26188314	0.44877306	0.54531133
300	4.0	0.28927073	0.26230747	0.45035997	0.55199754
400	4.0	0.28676552	0.26084223	0.44840297	0.55150783
500	4.0	0.28513256	0.2604717	0.4491748	**0.55514926**

Irudia 5.8: PAM algoritmoarekin lortutako emaitzak, konparazioan distantzia erlatiboa erabilita.

moztean adinakoa izan. Ondoren, *SAHN*-en *average* eta complete bertsioen emaitzak ageri dira, 5.9 eta 5.10 tauletan, ondoz ondo.

Average					
Cluster kopurua	Proposamen kopurua	F-measure %25	F-measure %50	MF-measure %25	MF-measure %50
50	4.0	0.17648801	0.1573833	0.24273017	0.2658602
100	4.0	0.21105818	0.18629923	0.29792035	0.31621233
150	4.0	0.21839392	0.18612121	0.29873854	0.3093356
200	3.8	0.21325123	0.19102019	0.29907447	0.3206232
300	4.0	0.19179381	0.1713475	0.27282077	0.29497272
400	4.0	0.21384123	0.19036609	0.29970318	0.32239515
500	4.0	**0.21865813**	**0.19978055**	**0.3051327**	**0.34154588**

Irudia 5.9: *Average* bertsioarekin lortutako emaitzak, konparazioan distantzia erlatiboa erabilita.

Complete					
Cluster kopurua	Proposamen kopurua	F-measure %25	F-measure %50	MF-measure %25	MF-measure %50
50	4.0	0.17333977	0.16753878	0.23981734	0.28666544
100	4.0	0.21241322	0.1872013	0.29535353	0.31694454
150	4.0	0.2178128	0.19267723	0.30759948	0.3307286
200	4.0	0.21661945	0.19571619	0.30459428	0.33716062
300	4.0	0.21687488	0.1969347	0.3090829	0.34108433
400	4.0	0.22059724	0.19590677	0.3106003	0.33498693
500	4.0	**0.22215554**	**0.20293991**	**0.31479535**	**0.3534506**

Irudia 5.10: *Complete* bertsioarekin lortutako emaitzak, konparazioan distantzia erlatiboa erabilita.

Bi bertsio horietan, medoidea moztean emaitzak nabarmen kaxkartu direla ikusi dugu.

Baina oraindik ere, distantzia erlatiboa jartzean, pixka bat gehiago okertu dira. Hurrengo 5.11 taulan, berriz, *ward* bertsioaren emaitzak ikus daitezke.

Ward					
Cluster kopurua	Proposamen kopurua	F-measure %25	F-measure %50	MF-measure %25	MF-measure %50
50	4.0	0.2710802	0.24240232	0.39561045	0.4555181
100	4.0	0.26757303	0.24670903	0.3962221	0.4740199
150	4.0	0.26927572	0.2476083	0.40181375	0.4852096
200	4.0	0.25940558	0.24527316	0.38867354	0.48611918
300	4.0	0.26876718	0.24976543	0.41313583	0.505761
400	4.0	0.26256892	0.2486701	0.40333638	0.50662225
500	4.0	**0.27303997**	**0.2517683**	**0.42472714**	**0.5199812**

Irudia 5.11: *Ward* bertsioarekin lortutako emaitzak, konparazioan distantzia erlatiboa erabilita.

PAM-en kasuan gertatu den bezala, bertsio honekin ere hobekuntza nabaritzen da emaitzetan.

Azkenik, ikus ditzagun SEP algoritmoarekin lortutako emaitzak 5.12 taulan.

SEP					
Cluster kopurua	Proposamen kopurua	F-measure %25	F-measure %50	MF-measure %25	MF-measure %50
433	4.0	0.27973968	0.25164273	0.4339884	0.5186118

Irudia 5.12: SEP algoritmoarekin lortutako emaitzak, konparazioan distantzia erlatiboa erabilita.

Azken bertsio honen emaitzak ere hobetu direnez, berriz, SEP-en emaitzak PAM-en eta SAHN-en *ward* bertsioarekin lortutakoen artean kokatzen dira, beraz, bertsio hoberenetakoa izaten jarraitzen du.

5.2.3 Ondorioak

Profilak hobeto multzokatzeko egin ditugun bi aldaketekin, PAM algortimoaren bertsioan emaitzak %6an hobetzea lortu dugu, sekuentziaren laurdena ezagutuz, aurrikuspena egitean, eta, algoritmo hierarkikoaren *ward* metrikaren bertsioan eta SEP algoritmoarekin, berriz, %7. *Average* eta *complete* bertsioek, aldiz, emaitza txarragoak eman dituzte, alde handiarekin.

5.13 irudian ikusten den taulan, aldaketa honekin sistemaren bertsio bakoitzak eman dituen emaitza hoberenak konparatzen dira. 5.14 irudiko taulan, ordea, PAM eta SEP al-

goritmoek, guztien artetik emaitza hoberenak ematen dituztenek, orain arte egindako probetan emandako emaitza hoberenak konparatzen dira. "1" eta "2" lerroak aurreko atalean ikusi ditugun berak dira, baina "3" lerroa gehitu dugu, distantzia erlatiboaren aldaketa egin ostean lortutako emaitzak adierazteko. Ikus dezakegunez, emaitzak hobetzea lortu dugu, balio altuenak azken lerroan baitaude (letra lodiz markatuak).

Bertsioa	F-measure %25	F-measure %50	MF-measure %25	MF-measure %50
PAM	**0.29426676**	**0.26344365**	**0.44960847**	**0.55514926**
Average	0.21865813	0.19978055	0.3051327	0.34154588
Complete	0.22215554	0.20293991	0.31479535	0.3534506
Ward	0.27303997	0.2517683	0.42472714	0.5199812
SEP	0.27973968	0.25164273	0.4339884	0.5186118

Irudia 5.13: Sistemaren bertsio bakoitzarekin lortutako emaitza hoberenak, distantzia erlatiboa jarrita.

Bertsioa	F-measure %25	F-measure %50	MF-measure %25	MF-measure %50
PAM				
1	0.22991998	0.18031698	0.31587023	0.3044352
2	0.27934477	0.24831896	0.4211672	0.52136505
3	**0.29426676**	**0.26344365**	**0.44960847**	**0.55514926**
SEP				
1	0.20709915	0.17871068	0.29066983	0.29962885
2	0.26362854	0.23712198	0.40095013	0.4776237
3	**0.27973968**	**0.25164273**	**0.4339884**	**0.5186118**

Irudia 5.14: PAM eta SEP algoritmoekin inplementatutako bertsioen orain arteko emaitzen konparazioa.

Bukatzeko, ikus dezagun emaitzak gehiago hobetzea lortzen dugun kapitulu honen hasieran azaldutako hirugarren puntuarekin: estekak proposatzeko garaian, *cluster* edo profil desberdinak konbinatuz.

5.3 *Cluster* bat baino gehiago konbinatzea

Lehenago azaldu dugun bezala, sekuentzia batentzako proposamen egokienak ez dute zertan hurbileneko *cluster*-ean egon. Izan ere, baliteke hurbileneko *cluster*-eko lehenengo proposamena egokia izatea, baina bigarrenaren *support*-a oso baxua izatea. Aldi berean,

baliteke hurrengo *cluster* hurbileneko lehenengo edo bigarren URLa, dagoeneko proposamentzat hartu dugunaren desberdina izateaz gain, egokiagoa izatea, *support* handiagoa daukalako, hots, maizago klikatzen den esteka delako.

Hori dela eta, proposamenak *cluster*-era dagoen distantziaren arabera bakarrik aukeratu ordez, hartaraino dagoen distantziaren eta proposatzen ari garen estekaren *support*-aren arteko oreka bilatzen saiatu gara, horretarako zenbait konbinazio desberdin probatuz.

Proposamen egokiena lortzeko, distantziak ahalik eta txikiena izan behar du, eta, *support*-ak, aldiz, ahalik eta handiena. Hori dela eta, ezin dira zuzenean konbinatu, ez baikenuke jakingo zein balio den hoberena. Horregatik, distantzia erlatiboa daukagunez dagoeneko, bere zati osagarria bilatzen dugu 1 balioari distantzia kenduz, eta lortzen duguna *support*-arekin konbinatzen dugu. Horrela bai, badakigu balio maximoa bilatu behar dugula proposamen egokiena aurkitzeko.

Hurbileneko bi *cluster*-ak kontuan hartuz, hauetatik ateratzen zaigun proposamen bakoitzarekin konbinatzeko eragiketa kalkulatzen dugu eta, horietatik, emaitza altuena lortzen duten lau proposamenak hartzen ditugu kontuan. Proposamen bat errepikatzen bada, baztertu eta hurrengora pasatzen gara. Ez baditugu lau proposamen lortzen, aurreko metodo bera erabili dugu; hau da, hurbileneko hirugarren *cluster*-etik aurrera, distantzia bakarrik kontutan hartuz, falta diren proposamenak bilatzen ditugu.

5.3.1 Inplementazioa

Bi profil konbina ditzan lortzeko, proposamenak egiten dituen funtzioa aldatu behar dugu. Hasteko, ikus dezagun jatorrizko funtzioaren sasikodea:

Aurretik ikusi dugun *knnSimShortNorm*() (ikus 19. algoritmoa) funtzioari esker, *medOrdenatuak* eta *distOrdenatuak* aldagaietan, distantziaren arabera (txikienetik handienera) ordenatutako medoideak eta distantziak gordetzen ditugu, hurrenez hurren. Ondoren, *medOrdenatuak*-en dagoen *medoidea* osagai bakoitzeko, hau da, hurbileneko medoidetik hasi eta urrutien dagoeneraino, medoide horri dagokion *cluster*-eko proposamenak eta *support*-ak jasotzen ditugu *proposamenCl* eta *supportCl* aldagaietan. Horretarako, *recosInEachCluster*() funtzioa erabiltzen dugu; medoide jakin bat pasata, medoide horren *cluster*-eko proposamenak eta proposamen hauen *support*-ak itzultzen ditu. Proposamenak lortzean, *proposamenCl* multzoan dagoen *prop* proposamen bakoitzeko, nahikoa proposamen dauzkagun begiratzen dugu. Horretarako, *proposamenak* (proposamen orokorrak, hau da, *cluster* bat baino gehiago konbinatuz lortzen ditugun proposamenak gordetzen dituen aldagaia) aldagaiaren tamaina

Algoritmoa 20 getNextPossibleSteps()

```
1:  medOrdenatuak, distOrdenatuak ← knnSimShortNorm()
2:  for all medoidea ∈ medOrdenatuak do
3:      proposamenCl, supportCl ← recosInEachCluster(medoidea)
4:      for all prop ∈ proposamenCl do
5:          if proposamenak ≤ proposamenKopurua then
6:              if prop ∉ proposamenak then
7:                  proposamenak ← proposamenak ∪ prop
8:                  supportak ← supportak ∪ support(prop)
9:              end if
10:         end if
11:     end for
12: end for
13: return proposamenak
```

proposamenKopurua baino txikiagoa den begiratzen dugu. Baiezko kasuan, esku artean daukagun proposamena (*prop*) dagoeneko *proposamenak* aldagaian dagoen aztertzen dugu eta, ez badago, bertan sartzen dugu. Horretaz gain, *supportak* aldagaia ere eguneratzen dugu, *prop* proposamenari dagokion *support*-a gehituz. Aldagai horretan *proposamenak* aldagaiko proposamen orokorren *support*-ak gordetzen ditugu.

Orain, 21 algoritmoan, funtzio horretan egindako aldaketak erakusten ditugu.

Funtzioari hasieran txertatu diogun zatiari esker, lehenengo bi *cluster*-etako datuak lortzen ditugu (proposamenak eta *support*-ak, *propX* eta *supX* izeneko aldagaietan, hurrenez hurren, bai lehenengo eta bai bigarren *cluster*-entzat). Bien datuak *propLag* eta *supLag* aldagaietan elkartu ondoren, proposamen bakoitzarentzat, *balioak* aldagaian formularen balioa gehitzen dugu (formula aldakorra da). Ondoren, aurrerago azaltzen dugun *sort()* (ikus 22 algoritmoa) funtzioaren bitartez, aldagai guztiak *balioak* aldagaiaren arabera ordenatzen ditugu, balio handiena dutenak hasieran jarriz eta *propLag*2 eta *supLag*2 aldagaietan gordetzen ditugu. Bukatzeko, balio hoberena ematen duten lau proposamenak gehitzen ditugu *proposamenak* multzo nagusian, horretarako *propLag*2 aldagaiko proposamenak aztertuz, eta, nahikoak ez badaude, jatorrizko funtzioan egiten den bezala, distantziaren arabera aukeratzen dira hurrengo proposamenak, hirugarren *cluster*-etik aurrera aztertzen hasita.

Ikus dezagun sakonago *sort()* funtzioak egiten duena 22 algoritmoan. Hiru *array* emanik, azkenekoaren arabera ordenatzen ditu denak; hau da, *balioak* zerrendan balio altuena duen proposamenaren datuak jartzen ditu hiru *array*-etan lehenengo posizioan.

Funtzioak *balioak* zerrendako balio bakoitza zerrendan aurrerago dauden balioekin

Algoritmoa 21 getNextPossibleSteps_Comb()

1: $medOrdenatuak, distOrdenatuak \leftarrow knnSimShortNorm()$
2: $prop1, sup1 \leftarrow recosInEachCluster(medOrdenatuak(1))$
3: $prop2, sup2 \leftarrow recosInEachCluster(medOrdenatuak(2))$
4: $propLag \leftarrow prop1 \cup prop2$
5: $supLag \leftarrow sup2 \cup sup2$
6: **for all** $prop \in propLag$ **do**
7: $balioak \leftarrow balioak \cup formula(supLag(prop), distantzia(prop))$
8: **end for**
9: $propLag2, supLag2 \leftarrow sort(propLag, supLag, balioak)$
10: **for all** $prop \in propLag2$ **do**
11: **if** $proposamenak \leq proposamenKopurua$ **then**
12: **if** $prop \notin proposamenak$ **then**
13: $proposamenak \leftarrow proposamenak \cup prop$
14: $supportak \leftarrow supportak \cup supLag2(prop)$
15: **end if**
16: **end if**
17: **end for**
18: 20 algoritmoko 2. lerroko for-aren berdina, baina hirugarren *cluster*-etik aurrera aztertuz
19: **return** proposamenak

Algoritmoa 22 sort($propLag, supLag, balioak$)

1: **for** $i = 0$**to***luzera*($balioak$ **do**
2: **for** $k = i \in luzera(balioak)$ **do**
3: **if** $balioak(i) < balioak(k)$ **then**
4: $permute(balioak, i, k)$
5: $permute(propLag, i, k)$
6: $permute(supLag, i, k)$
7: **end if**
8: **end for**
9: **end for**
10: $emaitza \leftarrow propLag, supLag$
11: **return** emaitza

konparatzen du; posizio txikienean dagoena (i) posizio handienekoa (k) baino txikiagoa bada, lekuz trukatzen ditu zerrenda guztietan (hori da $permute()$ funtzioak egiten duena bektore bat eta bi posizio pasata). Horrela, hiru zerrendak formulako balio handiena duenetik txikiena duenera ordenatzen ditu, horrela, proposamen egokienak zerrendaren hasieran uzten ditu.

5.3.2 Probak

Proba bakoitzerako, 21 algoritmoko 7. lerroan adierazi dugun $formula()$ funtzioa aldatu dugu, distantziari eta *support*-ari eragiketa desberdina aplika diezaion.

Lehenengo kasuan, distantziaren osagarria *support*-arekin konbinatzeko biderketa egin dugu, hau da, $(1 - distantzia) * support$. Ikus ditzagun sailkatzaile bakoitzarekin lortzen diren emaitzak:

PAM					
Cluster kopurua	Proposamen kopurua	F-measure %25	F-measure %50	MF-measure %25	MF-measure %50
50	4.0	0.29445523	0.26260287	0.4434511	0.5256766
100	4.0	**0.2954814**	**0.26558524**	0.45248914	0.54521054
150	4.0	0.2925873	0.264347	0.45223507	0.54988325
200	4.0	0.29298478	0.26375276	0.45393363	0.5507059
300	4.0	0.2882538	0.26450402	0.4486553	0.5567922
400	4.0	0.29127616	0.26265234	**0.45562798**	0.5564059
500	4.0	0.289582	0.26173353	0.45559758	**0.559509**

Irudia 5.15: PAM algoritmoarekin lortutako emaitzak, 2 profil konbinatuz.

5.15 irudian, PAMekin lortutako emaitzak ageri dira. Profilak konbinatuz eta bai medoidearen eta bai distantziaren aldaketak mantenduz, emaitzak beste pixka bat hobetzea lortu dugu, gutxi bada ere.

Average					
Cluster kopurua	Proposamen kopurua	F-measure %25	F-measure %50	MF-measure %25	MF-measure %50
50	4	0.17600378	0.15608937	0.24192464	0.26464766
100	4	0.21114269	0.18670802	0.2982393	0.31674197
150	4	**0.21876793**	0.18826728	0.29971987	0.31153515
200	3.8	0.21188442	0.18929626	0.29790702	0.3182867
300	4	0.19138554	0.17042181	0.2729789	0.29520693
400	4	0.21325298	0.1904742	0.29936165	0.32288092
500	4	0.21653977	**0.19704494**	**0.3037082**	**0.33955842**

Irudia 5.16: *Average* bertsioarekin lortutako emaitzak, 2 profil konbinatuz.

5.16 eta 5.17 tauletan, SAHN algoritmoaren *average* eta *complete* bertsioekin lortutako emaitzak ikus ditzakegu eta ez dira ia aldatzen. 5.18 irudian, *ward* metrikarekin inplementatutako bertsioaren emaitzak daude; horiek ere, pixka bat hobetu dira, PAM algoritmoaren bertsioarekin batera.

Complete					
Cluster kopurua	Proposamen kopurua	F-measure %25	F-measure %50	MF-measure %25	MF-measure %50
50	4.0	0.17286712	0.16885723	0.24014708	0.28855425
100	4.0	0.21374913	0.18545555	0.29682168	0.31674522
150	4.0	0.21803431	0.19415484	0.30781817	0.33081362
200	4.0	0.216239	0.19662279	0.30427736	0.33779624
300	4.0	0.21421058	0.19142139	0.30642644	0.33733374
400	4.0	0.21844761	0.19266205	0.30905762	0.331374
500	4.0	**0.2215822**	**0.20189288**	**0.3144154**	**0.35240564**

Irudia 5.17: *Complete* bertsioarekin lortutako emaitzak, 2 profil konbinatuz.

Amaitzeko, 5.19 irudiko emaitzetan ere, SEP algoritmoarekin emaitzak pixka bat hobetzea lortu dugula ikus dezakegu.

Ward					
Cluster kopurua	Proposamen kopurua	F-measure %25	F-measure %50	MF-measure %25	MF-measure %50
50	4.0	0.27677625	0.24614927	0.40191323	0.45935088
100	4.0	0.27204227	0.25038153	0.39973158	0.47843057
150	4.0	**0.277951**	0.25067478	0.4148627	0.49021083
200	4.0	0.26628032	0.24918011	0.39855364	0.49103403
300	4.0	0.27123684	0.25283104	0.41595984	0.50812864
400	4.0	0.26560295	0.2513059	0.40716895	0.5094493
500	4.0	0.2743855	**0.2541092**	**0.4265646**	**0.52284753**

Irudia 5.18: *Ward* bertsioarekin lortutako emaitzak, 2 profil konbinatuz.

SEP					
Cluster kopurua	Proposamen kopurua	F-measure %25	F-measure %50	MF-measure %25	MF-measure %50
433	4.0	0.2845369	0.25517926	0.4393715	0.52318054

Irudia 5.19: SEP algoritmoarekin lortutako emaitzak, 2 profil konbinatuz.

Ondoren, emaitza hoberenak eman dituzten bi bertsioekin, beste formula bat probatu dugu, biderketaren bitartez izan ordez, baturarekin konbinatuz. Bi kasuetan, emaitzak pixka bat okertu dira. Horregatik, distantziak indar gehiago behar duela pentsatu dugu, beraz, formulan, distantziaren osagarriari 1.5 biderkatu diogu: $1.5 * (1 - distantzia) + support$.

PAM algoritmoarekin lortuako emaitzak begiratzean, lortu berri ditugun emaitzak biderkagairik gabeko baturaren emaitzen berdinak direla konturatu gara. Hau da, berdin da zenbateko indarra ematen diogun distantziari, emaitzak ez dira aldatzen. Beraz, aukeraketa *support*-aren menpe dagoela ondorioztatu dugu.

Kodea aztertzean, distantziarekin gertatu den moduan, *support*-a ere normalizatu gabe dagoela konturatu gara. Hortaz, hurrengo proba moduan, formula bera aplikatzea erabaki dugu, baina *support*-a *cluster*-eko datu kopuruarekin zatituz.

Oraingoan ere ez ditugu datuak erakutsi, izan ere, emaitzek pixka bat okerrera egin dute. Ondorioz, lehen *support*-ak zeuzkan balio altuak kontuan hartuta, horrek indar gehiago behar duela pentsatu dugu. Hori dela eta, beste proba bat egitea bururatu zaigu, *support* erlatiboari biderkatzaile bat jarriz, indarra emateko.

Emaitzak ez direnez askorik aldatu, emaitza hobeak ematekotan, *support*-ak askoz indar gehiago behar du. Hortik, egin dugun lehen proba da emaitza hoberenak lortzen dituena; biderketa egiten badugu ere, lehenengo bi *cluster*-etako proposamenen aukeraketa *support*-aren balio absolutuaren eta distantziaren osagarriaren arteko biderketaren arabera egiten duena.

5.3.3 Ondorioak

Atal honen amaieran ere, sistema zertxobait hobetzea lortu dugula esan dezakegu. Zenbait aukera desberdin aztertu ondoren, sistemaren bertsiorik onena honakoa izan da: *support* absolutuarekin biderketaren formula erabili duguna, hau da, $(1 - distantzia) * support$.

Gainera, aurreko aldaketekin irabazitakoaz gain, beste %2-ko hobekuntza izatea lortu dugu sekuentziaren laurdena ezagutuz egindako aurreikustepenean; hasierako bertsiotik, %7-8 inguru.

Edonola ere, SAHN algoritmoarekin inplementatutako *average* eta *complete* aukeren bertsioen kasuan, emaitzek behera egin dute aldaketak egin ahala. Aurreneko proba egitean zerbait gaizki egin dugula pentsatu badugu ere, probekin aurrera egin dugun heinean, jarrera bera jarraitu dutela ikusi dugunez eta probak behin eta berriz egin ditugunez, ez dugu portaera horren arrazoirik topatu. Beraz, etorkizunerako lan moduan uzten dugu kasu hauek hobeto aztertzea.

Bestalde, aipatu beharrekoa da SEP algoritmoarekin inplementatutako sistemak ere, PAM-en nahiko parean, emaitzak onak eman dizkigula. Hain onak ez badira ere, SEP

algoritmoarekin dagoen sistema askoz orokorragoa da eta beste edozein daturentzat balio du, inongo moldaketarik egin behar izan gabe, batez ere, *k* parametroaren azterketa saihestuz.

Laburpen moduan, konbinazio hoberenarekin, hau da, *support* absolutuarekin biderketa bidez konbinatzean, bertsio bakoitzarentzat lortutako emaitza hoberenak 5.20 taulan ikus daitezke.

Bertsioa	F-measure %25	F-measure %50	MF-measure %25	MF-measure %50
PAM	**0.2954814**	**0.26558524**	**0.45562798**	**0.559509**
Average	0.21876793	0.19704494	0.3037082	0.33955842
Complete	0.2215822	0.20189288	0.3144154	0.35240564
Ward	0.277951	0.2541092	0.4265646	0.52284753
SEP	0.2845369	0.25517926	0.4393715	0.52318054

Irudia 5.20: Sistemaren bertsio bakoitzarekin lortutako emaitza hoberenak, *support* absolutua distantziarekin biderketa eginez konbinatuta.

Bi bertsio hoberenen hobekuntza laburtzeko, proiektuan zehar PAM eta SEP algoritmoekin inplementatutako bertsioekin orain arte lortutako emaitza guztiak agertzen dira 5.21 irudiko taulan. Lehenengo hiru lerroak azaldu ditugu dagoeneko. Laugarren lerroa, *support* absolutuarekin, konbinaketa biderketa bidez egiteari dagokio.

Bertsioa	F-measure %25	F-measure %50	MF-measure %25	MF-measure %50
PAM				
1	0.22991998	0.18031698	0.31587023	0.3044352
2	0.27934477	0.24831896	0.4211672	0.52136505
3	0.29426676	0.26344365	0.44960847	0.55514926
4	**0.2954814**	**0.26558524**	**0.45562798**	**0.559509**
SEP				
1	0.20709915	0.17871068	0.29066983	0.29962885
2	0.26362854	0.23712198	0.40095013	0.4776237
3	0.27973968	0.25164273	0.4339884	0.5186118
4	**0.2845369**	**0.25517926**	**0.4393715**	**0.52318054**

Irudia 5.21: PAM eta SEP algoritmoekin inplementatutako bertsioen orain arteko emaitzen konparazioa.

Bukatzeko, proiektu honetan zehar egin beharreko aldaketekin lor genezakeen sistema

hoberena esku artean daukagunez, ikus dezagun zein emaitza ematen dituen test-eko kasuekin. 5.22 irudian, bi algoritmo hoberenekin lortutako emaitza hoberenak ageri dira.

	F-measure %25	F-measure %50	MF-measure %25	MF-measure %50
PAM	**0.2978075**	**0.26589862**	**0.45815358**	**0.55778354**
SEP	0.2851257	0.25531024	0.4402358	0.52196157

Irudia 5.22: PAM eta SEP algoritmoekin inplementatutako bertsioen testeko emaitza hoberenak.

Ikus dezakegunez, bi kasuetan emaitzak pixka bat hobetzen dira, eredu osoa kontuan hartuz eta sekuentziaren %50a ezagututa ateratzen ditugun kalkuluetan izan ezik, azken horiek pixka bat okertzen baitira (ikus 5.21 irudiko taula).

6. kapitulua

Sistema beste testuinguru batean aplikatzea

6.1 Discapnet.net ingurunea

Discapnet webgunea ezintasunak dituzten pertsonei zuzendutako webgunea da. ONCE fundazioak eta Technositek ordaindutako ekimena da, ezinduak gizartean eta lanean integratzen laguntzeko. Bi lerro nagusi dauzka: alde batetik, informazio-zerbitzua, elkarte, profesional, ezindu eta familientzat; bestetik, pertsona ezinduek parte hartzeko ekintza ekonomikoak eta kultur- eta gizarte-ekintzak garatzeko plataforma bat.

Technositek 10 hilabeteetan zehar bi zerbitzarietan jasotako mugimendu osoa *log* formatuan jaso du eta hortik, eskaera edo egoera akastunak kendu eta saioak identifikatuz, sekuentzien datu-basea lortu da.

Hala ere, webgunea pixka bat aztertuz gero, informazioaren zatirik handiena dinamikoa dela ikus dezakegu. Webguneak hiru zati nagusi dauzka: *Áreas Temáticas*, *Comunidad* eta *Actualidad*. Horien zati batzuk oso aldakorrak dira eta ezin dira kontuan hartu.

Guk proiektuan aztertu dugun zatia *Áreas Temáticas* delakoa izan da. Bere barnean, honako guneak dauzka: *Accesibilidad*, *Derechos*, *Educación*, *Empleo*, *Medioambiente*, *Ocio y cultura*, *Tecnología* eta *Canal Senior*. Azpiatal horien artean, ordea, diferentzia handiak daude. Webgunea begiratzen badugu, *Derechos* atala oso zabala dela kontura gaitezke, adibidez; ondorioz, bertako estekak asmatzea zaila izan daiteke, aukera ugari baitago eta denei desberdina interesa baitakieke.

Irudia 6.1: Discapnet webgunearen sarrerako orria.

6.1.1 Probak

Kapitulu honen helburua proiektuan zehar lortutako sistema hoberena ingurune berrira aplikatzea da. Hori dela eta, sistemaren azken bi bertsio hoberenak (PAM eta SEP algoritmoekin inplementatu ditugun sistemak) *Discapnet* inguruneko datuekin jarri ditugu martxan. Datu-base horrek, atal guztiak kontuan hartuta, 48.000 kasu pasa dauzka. Horregatik, sistema datu-base horrekin martxan jartzean, *ALDAPA* taldeak laborategian utzitako ordenagailua ez da gai exekuzio osoa bukatzeko.

Memoria arazoa denez eta aipatutako ordenagailuak 8GB-ko memoria duenez, Korta eraikinean dauden ordenagailuekin proba egin dugu; horiek 16GB-eko memoria baitaukate. Oraingoan, hasiera batean, memoriak ez ditu arazoak ematen, baina denbora asko behar du exekutatzen eta, azkenean, memoria faltagatik, exekuzioa bukatu gabe gelditzen da. Izan ere, k bakoitzerako hasierako medoideak aukeratzeko, batezbestean, 7 ordu behar ditu eta, ondoren, truke bakoitza egiteko, ordu eta erdi. Hainbeste kasu direla kontuan hartuz, truke asko egin behar dira eta, hortaz, denbora asko behar du k bakar baten PAM egiteko. Eta gainera, balidazio gurutzatua egiteko eskatzen badiogu, asteak behar ditugu sistema PAM-ekin exekutatzeko.

Hori guztia kontuan hartuta, arazo horiek saihesteko, beste aukera batzuk hausnartu ditugu. Alde batetik, datu-base osoa aztertzeak makinari eragiten dion lan-karga txikiagotzeko, balidazio gurutzatua egin ordez, *holdout* teknika erabiltzea pentsatu dugu.

Bestetik, proiektuan zehar aurkitu dugun k hoberena BTw ingurunearentzat bakarrik baliagarria denez, ezin dugu datu-base honetan emaitza hoberenak k berak ematen dituela ziurtatu, baina, edonola ere, *ALDAPA* taldeak aurretik egin dituen ikerketak aztertuta [O. Arbelaitz, 2014], *Discapnet* ingurunean emaitza hoberenak ematen dituen k 130 dela badakigu. Beraz, gainera, k guztiak probatu ordez, 130-ekin bakarrik probatzea daukagu aukeren artean.

Balidazioa egitea, hortaz, alferrikakoa da, zuzenean test-eko kasuak aztertzen baititugu eta, gainera, k hoberena finkatuta baitaukagu dagoeneko. Horrek egokitzapen txiki bat egitea eskatzen digu: oraingoan, ikasteko, lehen bezala, datuen %70 erabiltzen dugu, baina, gainontzekoa, %30a, test-erako erabiltzen dugu, balidazioa hutsik utziz.

Hala ere, datu-base osoa aztertu nahi izateak, nahiz eta *holdout* egin, sekuentzien datu guztiak memorian kargatzea dakar. Hori dela eta, oraindik ere arazoak ematen jarraitzen du.

Ondorioz, buruan geneukan bigarren aukerari ekin diogu. Aurretik aipatu dugun bezala, *Discapnet* ingurunea gai desberdinen inguruko informazioa jasotzen duen zatietan egituratuta dago, eta, hortaz, datu-basean ere sekuentziak gaika multzokatuta dauzkagu. Horrela, datu-base txikiagoak lortzen ditugunez, sistemak exekuzioak arrakastaz buka ditzake eta balidazio gurutzatua erabil dezakegu.

Lehenengo eta behin, datu-basearen gaikako banaketa zein den ikus dezakegu 6.1 taulan.

Ident.	**Gaia**	**Saio-kopurua**
0	*Accesibilidad*	10259
1	*Derechos*	22561
2	*Educación*	1773
3	*Empleo*	4720
4	*Medioambiente*	852
5	*Ocio y cultura*	3603
6	*Tecnología*	3954
7	*Canal Senior*	338

Taula 6.1: *Discapnet* datu-basearen banaketa gaika.

Lehenengo zutabean, datu-basearen zati bakoitzaren identifikatzailea ageri da, aurrerago agertzen diren proben emaitzetan adierazten duguna; jarraian, identifikatzaile bakoitza zein gairen sekuentziak dituen datu-baseari dagokion erakusten da, eta, azken zutabean, berriz, datu-base bakoitzak zenbat saio jasotzen dituen.

Datu-base guztien tamaina desberdina denez, balidazio gurutzatua egitean sortzen den

Datu-basea	Ikasteko	Testerako
0	7182	3077
1	15792	6768
2	1239	531
3	3304	1416
4	595	255
5	2520	1080
6	2765	1185
7	238	100

Taula 6.2: *Discapnet* ingurunearen balidazio gurutzatuaren banaketa gaika.

banaketa ere desberdina da guztietan. 6.2 taulan, gai bakoitzean ikasteko eta azterketa egiteko erabiltzen diren saio-kopurua adierazten da, izan ere, balidazioa hutsik uztea erabaki dugu, proba hauen helburua ez baita k hoberena bilatzea. Gainera, *ALDAPA* taldeak egindako aurreko ikerketei esker [O. Arbelaitz, 2014], badakigu datu-base bakoitzarentzat emaitza hoberenak ematen dituen *cluster* kopurua zein den sistema PAM algoritmoarekin daukagunerako. Probetan lortutako emaitzen taulan bertan adierazten dugu balio zehatza.

Web meatzaritzako sistemarako proposatu ditugun hobekuntzak edozein ingurunetan eragiten duten errendimenduaren hobekuntza ikusteko, sistemaren azken bertsioak martxan jarri aurretik, jatorrizko sistemak ematen dituen emaitzak aztertzea komeni da, ondoren, konparatzeko erreferentziak izateko. 6.2 eta 6.3 irudietako tauletan, jatorrizko sistemak PAM eta SEP algoritmoekin inplementatuta ematen dituen emaitzak ikus daitezke, hurrenez hurren. Lehenengo zutabean, lerro horretan aztertzen den datu-basearen identifikatzailea adierazten da. Identifikatzailea zein gairi dagokion jakiteko, jo 6.1 taulara. Bigarren zutabean, datu-base jakin bakoitzean sortutako *cluster*-kopurua adierazten da; PAM-en kasuan, *ALDAPA* taldeak idatzitako artikuluan egokitzat adierazitakoa [O. Arbelaitz, 2014], eta, SEP-en kasuan, algoritmoak berak erabakitakoa. Hurrengo zutabeetan, orain arteko probetan aztertu ditugun datu berak adierazten dira: proposamen-kopurua eta ebaluazio neurriak, sekuentziaren zati ezberdinak ezagutzen direnerako. Amaitzeko, beheko lerro grisean, batezbestekoak erakusten ditugu, zutabe bakoitzarentzat.

PAM algoritmoaren kasuan (6.2 irudia), emaitza hoberenak 3 datu-basearekin (*Empleo*) lortzen dira. Txarrenak, aldiz, 5 datu-basearekin (*Ocio y cultura*). Ikus dezakegunez, bi datu-baseen arteko aldea ikaragarria da. Izan ere, lehen esan bezala, datu-base guztiak oso desberdinak dira; bai eskaintzen duten informazioagatik eta bai tamainagatik. Hori dela eta, kasu batzuetan besteetan baino zailagoa izan daiteke estekak behar bezala proposatzea. Horregatik, gutxigorabeherako erreferentzia bat izateko, ebaluazio-neurri

PAM						
Datu-basea	Cluster kopurua	Proposamen kopurua	F-measure %25	F-measure %50	MF-measure %25	MF-measure %50
0	90	4.0	0.3127653	0.01602162	0.05942478	0.04209957
1	80	4.0	0.08503111	0.05847179	0.09962768	0.09962768
2	27	4.0	0.21045063	0.14276740	0.23187721	0.23187721
3	60	4.0	**0.46586832**	**0.29722300**	**0.55637085**	**0.55637085**
4	20	4.0	0.12038895	0.07800423	0.15200943	0.15200943
5	50	4.0	0.04023491	0.02279714	0.057640497	0.058106527
6	50	4.0	0.28467345	0.24798425	0.2988338	0.2988338
7	13	4.0	0.14529061	0.05907292	0.23600143	0.2133591
Batezbeste	48,75	4.0	0.172901814	0.1152927925	0.2114732096	0.2065355209

Irudia 6.2: Jatorrizko sistemaren PAM algoritmoarekin inplementatutako bertsioak *Discapnet* inguruneko datuekin emandako emaitzak.

bakoitzaren batezbestekoa erabil dezakegu.

SEP						
Datu-basea	Cluster kopurua	Proposamen kopurua	F-measure %25	F-measure %50	MF-measure %25	MF-measure %50
0	762	4.0	0.02142377	0.01102932	0.05087318	0.029686172
1	2150	4.0	0.01495660	0.01211331	0.016548835	0.016548835
2	120	4.0	0.19089568	0.12007334	0.32837203	0.32837203
3	78	4.0	**0.45009166**	**0.29830346**	**0.5519427**	**0.5519427**
4	111	4.0	0.10602778	0.06779141	0.13339	0.13339
5	357	4.0	0.08236446	0.04546076	0.09812513	0.09812513
6	121	4.0	0.16570106	0.12779250	0.22263567	0.21564041
7	54	4.0	0.10660926	0.06418780	0.20068055	0.1947575
Batezbeste	469,125	4.0	0.1604448086	0.1049483696	0.2003210119	0.1960578471

Irudia 6.3: Jatorrizko sistemaren SEP algoritmoarekin inplementatutako bertsioak *Discapnet* inguruneko datuekin emandako emaitzak.

6.3 irudian ikus dezakegunez, SEP-en jokaera ere antzekoa da. Emaitza hoberenak, oraingoan ere, 3 datu-basearekin lortzen dira. Kaxkarrenak, berriz, 1 datu-basearekin (*Derechos*), sarreran aipatu dugun bezala, datu-base zaila baita, eskaintzen duen informazio zabala dela eta. 5 datu-basea, PAM-ekin txarrena zena, SEP-ekin lortutako emaitzetan ere txarrenen artean kokatzen da.

Bi tauletako batezbestekoak aztertzen baditugu, bai PAM algoritmoarekin eta bai SEP algoritmoarekin lortutako emaitzak oso antzekoak direla ikus dezakegu, nahiz ta PAM-enak zertxobait hobeak diren. Hala ere, kontuan hartzekoa da PAM-ekin emaitza horiek

lortzeko ordu askotako azterketa behar izan dela, ikertzen, probak egiten eta emaitzak konparatzen, datu-base bakoitzarentzako *k* hoberena zein den ondorioztatu ahal izateko. SEP-ekin, ordea, exekuzio bakarra martxan jartzea nahikoa izan da maila bereko emaitzak lortzeko eta horrek abantaila ikaragarria dakar.

Hasierako emaitzak ikusi ondoren, sistemaren azken bertsioarekin hobekuntzarik lortu dugun ikusteko garaia heldu da. 6.4 eta 6.5 tauletan, sistemaren azken bertsioak PAM eta SEP algoritmoekin lortzen dituen emaitzak ikus daitezke, ondoz ondo.

PAM						
Datu-basea	Cluster kopurua	Proposamen kopurua	F-measure %25	F-measure %50	MF-measure %25	MF-measure %50
0	90	4.0	0.18416995	0.14832027	0.31666696	0.4154882
1	80	4.0	0.13924727	0.09730611	0.2246805	0.30358532
2	27	4.0	0.27663568	0.15372911	0.42169797	0.48358977
3	60	4.0	**0.47584632**	**0.30225280**	**0.579004**	**0.6016059**
4	20	4.0	0.17971404	0.14281365	0.28711307	0.38250038
5	50	4.0	0.17945665	0.13799335	0.29330742	0.34867957
6	50	4.0	0.34755057	0.31065527	0.417347	0.48397645
7	13	4.0	0.20354803	0.08259472	0.23406662	0.27804828
Batezbeste	48,75	4.0	0.2482710638	0.17195816	0.3467354425	0.4121842338

Irudia 6.4: PAM algoritmoarekin inplementatutako azken bertsioak *Discapnet* inguruneko datuekin emandako emaitzak.

SEP						
Datu-basea	Cluster kopurua	Proposamen kopurua	F-measure %25	F-measure %50	MF-measure %25	MF-measure %50
0	762	4.0	0.16843140	0.15080996	0.27244553	0.4117051
1	2150	4.0	0.11398599	0.09852812	0.18930517	0.30498153
2	120	4.0	0.20366348	0.14855707	0.3563846	0.4773628
3	78	4.0	**0.44935253**	**0.30789420**	**0.55740696**	**0.5958842**
4	111	4.0	0.21961310	0.13713540	0.33132258	0.36013344
5	357	4.0	0.16882475	0.11173730	0.28354344	0.31831643
6	121	4.0	0.26299545	0.21849585	0.3382769	0.3844057
7	54	4.0	0.17226057	0.12228982	0.3138028	0.35884407
Batezbeste	469,125	4.0	0.2350201829	0.1709885137	0.3303109975	0.4014541588

Irudia 6.5: SEP algoritmoarekin inplementatutako azken bertsioak *Discapnet* inguruneko datuekin emandako emaitzak.

Jatorrizko sistemarekin konparatuz, emaitzak zentzuzkoak direla ikus dezakegu, hau da, lehen emaitza hoberenak enpleguaren datu-baseak ematen zituen, eta orain, ere bai;

berdin datu-base txarrenekin. Edonola ere, lorpenak askoz hobeak dira orain, ebaluazio--neurriek oro har ematen dituzten emaitzak askoz altuagoak baitira, nahiz eta datu-base hoberenaren emaitzak ez diren hainbeste hobetu (SEP-en kasuan pixka bat kaxkartu da).

Hori gutxi balitz, berriz ere PAM-en eta SEP-en emaitzak nahikoa parekatuta daude. Horrek SEP-ekin inplementatutako sistema egonkorra eta hobea dela baieztatzen digu, izan ere, ez diogu minutu bakar bat ere eskaini beharrik izan emaitza on horiek lortzeko, PAM-ekin ez bezala. Hortaz, PAM-ek lor ditzakeen emaitza hoberenen maila bereko emaitzak lor ditzakegu, *k*-ren azterketa guztia egin gabe. Ondorioz, sistemak emaitza ahalik eta hoberenak lehen exekuzioan lortzea ahalbidetzen digu, edozein ingurunetako datuak aplikatzen dizkiogula ere.

6.2 Ondorioak

Atal honetan, *Discapnet* ingurunearekin egindako lana berrikusten da. 6.6 irudiko taulan, aztertutako bi bertsioek (hasierakoa eta azkena) algoritmo bakoitzarekin ebaluazio-neurri bakoitzarentzat eman dituzten emaitzen batezbestekoak adierazten dira, hau da, datu-base guztien emaitzen batezbestekoak. Lehenengo zutabean sistemak erabiltzen duen algoritmoa ageri da eta, bigarrenean, algoritmo bakoitzeko, sistemaren zein bertsio den.

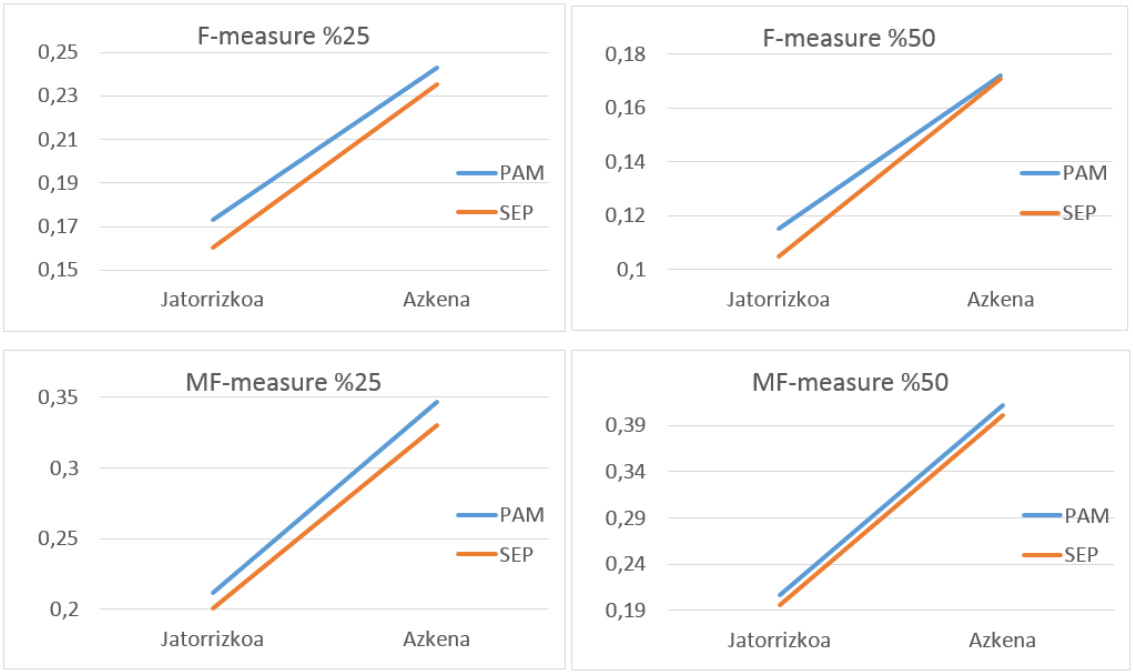
Algoritmoa	Bertsioa	F-measure %25	F-measure %50	MF-measure %25	MF-measure %50
PAM	Jatorrizkoa	0.172901814	0.1152927925	0.2114732096	0.2065355209
	Azkena	**0.2482710638**	**0.17195816**	**0.3467354425**	**0.4121842338**
SEP	Jatorrizkoa	0.1604448086	0.1049483696	0.2003210119	0.1960578471
	Azkena	0.2350201829	0.1709885137	0.3303109975	0.4014541588

Irudia 6.6: Hasierako eta azken bertsioekin lortutako emaitzen batezbestekoen konparazioa, algoritmoaren arabera.

Ikus dezakegunez, aurreikuspeneko kasuetan, bi algoritmoekin %7-ko hobekuntza lortzen da sekuentziaren laurdena edo erdia ezagutzen bada. Aldiz, eredu osoa kontuan hartuta ebaluatzean, emaitzak gehiago hobetzen dira: %13 sekuentziaren laurdena ezagutzen den kasuan eta %21, sekuentziaren erdia ezagutzen dugunean.

Jarraian, 6.7 irudiko grafikoetan, goiko grafikoetan aurreikuspenerako lortu ditugun hobekuntzak ageri dira; beheko grafikoetan, berriz, eredu osoa kontuan hartuz lortutako emaitzak. Bestalde, ezkerreko grafikoetan, sekuentziaren laurdena ezagutzen dugun kasuak daude; eskuinekoan, aldiz, sekuentziaren erdia ezagutzen dugun kasuak. Hala ere,

grafiko bakoitzak aztertzen duen ebaluazio-neurria zein den izenburuak adierazten du. Grafiko bakoitzean, lerro urdinak PAM algoritmoaren emaitzak adierazten ditu; lerro laranjak, ordea, SEP algoritmoarenak. Lehenengo puntuko balioak jatorrizko sistemaren emaitza adierazten du, eta bigarren puntuak, sistemaren azken bertsioarena. Bukatzeko, grafikoaren ezkerreko balioek (bertikalean adieraziek) emaitzen balioak adierazten dituzte.



Irudia 6.7: Probetako emaitzen hobekuntzak.

Oro har, bi algoritmoekin egiten den hobekuntza parekoa dela ikus dezakegu eta hasieran SEPen emaitzak baxuagoak direnez, bukaeran ere, pixka bat txarragoak izaten jarraitzen dute. Hala ere, aurreikuspenean sekuentziaren %50 ezagutzen dugunean, bien emaitzek bat egitea lortu dugu.

Emaitzak BTw-ko datu-basearekin lortutakoak bezain onak ez badira ere, lortutako hobekuntza nabaria da.

Hurrengo orrietan, proiektuaren jarraipena eta ondorio orokorrak irakur daitezke.

7. kapitulua

Jarraipena eta kontrola

Kapitulu honetan, proiektuaren jarraipena egiten dugu, baita komunikazio-, kalitate- eta kontingentzia-planen kontrola ere, plangintzan adierazitakoarekin alderatuz. Bukaeran, ebaluazio pertsonala agertzen da.

7.1 Proiektuaren garapena

Atal honetan, zeregin bakoitzean behar izandako eta bakoitzarentzat aurreikusitako orduen arteko konparazioa egin dugu. Osotasunean, proiektua bukatzeko 430 ordu aurreikusita daude plangintzan. Edonola ere, jarraian ikus daitekeen bezala, 10'5 ordu gehiago behar izan ditugu.

Hurrengo 7.1 irudiak zeregin bakoitzeko aurreikusitako eta behar izandako orduak erakusten ditu eta bakoitzaren desbiderapena ondoan adierazten da, bai ordutan eta bai atazari esleitutako ordu-kopuruaren araberako ehunekoan ere. Berdez, aurreikusitakoa baino ordu gutxiago behar izan ditugun atazak markatu ditugu; gorriz, berriz, denbora gehiago behar izan dutenak. Bestalde, intentsitateak desbiderapenaren neurria adierazten du, zenbat eta handiagoa izan, orduan eta ilunagoa da kolorea.

Ez da proiektuaren osatzea arriskuan jarri duen atzerapen edo arazorik egon. Hala ere, plangintzan aurreikusitako orduekin konparatuz, desbideraketak gertatu dira; batez ere, jatorrizko sistema ezagutzean eta memoria idaztean (gorri ilunena). Bien larritasuna, ordea, ez da berdina. Izan ere, jatorrizko sistema ezagutzeko atazari 13 ordu besterik ez dizkiogu esleitu plangintzan eta, ondorioz, ordu gutxi batzuek desbiderapenaren ehunekoa handia

		Estimazioa	Erreala	Desbiderapena	
Kodea	Ataza	Denbora	Denbora	Denbora	Ehunekoa
-	-	430:00	440:30	10:30	2%
1	KUDEAKETA	43:00	43:05	0:05	0%
1.a	Plangintza	18:00	19:40	1:40	9%
1.b	Kontrola	10:00	9:20	-0:40	-7%
1.c	Jarraipena	15:00	14:05	-0:55	-6%
2	IKERKETA	43:00	45:40	2:40	6%
2.a	Sistema ezagutzea	13:00	16:00	3:00	23%
2.b	Algoritmoak aztertzea	30:00	29:40	-0:20	-1%
3	PROBAK	172:00	179:15	7:15	4%
3.a	Prestaketa	120:00	129:40	9:40	8%
3.b	Emaitzak aztertzea	52:00	49:35	-2:25	-5%
5	DATUAK APLIKATZEA	43:00	38:10	-4:50	-11%
6	ITXIERA	129:00	134:20	5:20	4%
6.a	Memoria idaztea	110:00	134:20	24:20	22%
6.b	Defentsa prestatzea	19:00			
	GUZTIRA:	430:00:00	440:30:00	10:30:00	2%

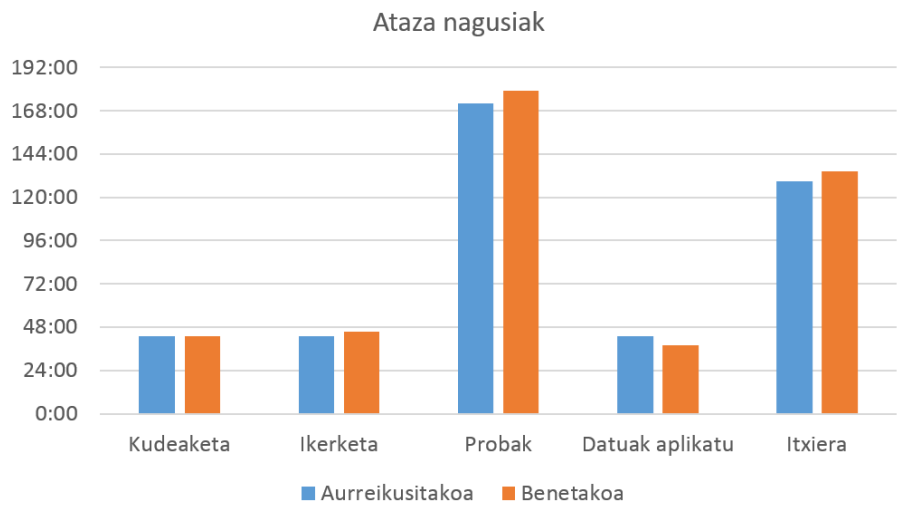
Irudia 7.1: Jarraipen-taula. Aurreikusitako-denborak eta denbora-erreala adierazten dira, desbiderapenekin batera.

izatea eragiten dute. Bestalde, memoria idaztean gertatutako desbiderapena handiagoa izan da, atazarentzat aurreikusitako ordu kopurua ere handiagoa baita.

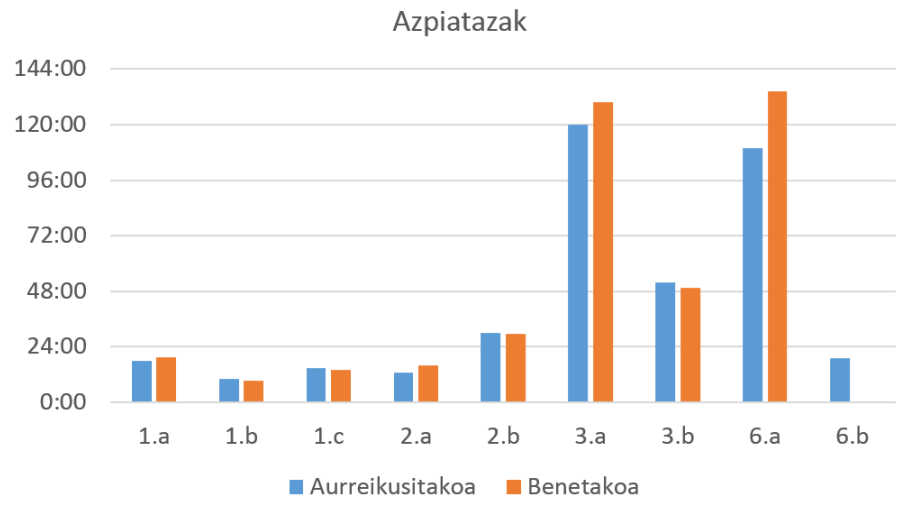
Azken lerro urdin argian ikus dezakegunez, oraindik defentsari dagozkion orduak ez ditugu bete, baina bai aurreikusitako guztiak. Beraz, horretarako koltxoi moduan erreserbatutako orduak erabil ditzakegu, izatez proiektua 450 ordukoa baita, aurretik esan bezala, Hezkuntza Ministeritzak emandako lankidetza-beka dela eta.

7.2 irudian, ataza nagusi bakoitzerako aurreikusitako orduen grafikoa (urdina) eta benetan behar izandakoena (laranja) alderatzen dira. Zutabe bikote bakoitzaren azpian zein atazari dagozkion adierazten da. Ezkerreko ardatzean orduak adierazten dira. Modu berean, 7.3 irudian, azpiataza bakoitzarekin egin dugu konparaketa. Kasu honetan, zutabe bikote bakoitzaren azpian azpiatazaren izena sartzen ez denez, kodea jarri dugu. Kode bakoitza zein azpiatazari dagokion aurreko jarraipen-taulan agertzen da (ikus 7.1 irudia).

Oro har, aurreikuspena nahiko egokia izan da. Desbiderapen handienak 3.*a* eta 6.*a* atazetan gertatu dira, hau da, probak prestatzean eta memoria idaztean. Probetan izandako desbiderapenaren arrazoia SEP algoritmoarekin izandako arazoak dira, izan ere, denbora asko pasa dugu arazoak aurkitzeko probak egiten eta konponbideak bilatzen. Memoria idaztean izandako desbideraketari dagokionez, esperientzia falta izan da arrazoi nagusia,



Irudia 7.2: Ataza nagusientzak aurreikusitako eta benetan erabilitako orduen konparazioa.



Irudia 7.3: Azpiatazentzat aurreikusitako eta benetan erabilitako orduen konparazioa.

inoiz ez baikara halako proiektu batean sartuta egon, eta, ondorioz, ez dugu hain dokumentu luzerik idatzi. Gainera, dokumentua *LaTeX*-ez idatzi izanak zeregina zaildu du, batez ere, bukaeran, formatu aldaketa batzuk egitean.

Bestalde, hasiera batean, plangintzan eguneroko iterazio-metodo bati jarraitzea proposatu dugu (ikus 2.2.3 atala eta 2.1 irudia). Proiektuari ekitean, metodo horri jarraitu diogu zeregin berarekin ez aspertzeko asmoz, baina, gerora, zeregin garrantzitsuenarekin edo presa gehiena duenarekin zentratzea egokiagoa dela ikusi dugu, beste zeregin batzuek itxaron baitezakete, proiektuaren aurrerapena oztopatu gabe.

7.2 Komunikazioak

Plangintzan sortutako komunikazio-planean, hiru komunikazio bide nagusi adierazi baditugu ere, horietako bi bakarrik erabili ditugu: posta elektronikoa eta aurrez aurreko komunikazioa, izan ere, Skype teknologia erabiltzeko beharrik ez dugu izan, edozein momentutan kideen laborategira edo zuzendariaren bulegora joateko aukera izan baitugu.

Laborategiko kideekin, oro har, aurrez aurre komunikatu gara, kasu bereziren batean izan ezik, posta elektronikoaren beharra izan baitugu, batez ere, exekuzioen inguruko berriren bat emateko.

Olatz Arbelaitzekin, berriz, biak erabili ditugu. Alde batetik, posta elektronikoa kontsulta edo galdera motzak egiteko erabili dugu, baita proiektuaren berri labur-labur emateko edo bilerak adosteko ere. Beste aldetik, arazo handiagoak izan ditugunean edo proiektuaren jarraipena sakonago egiteko, bilerak adostu eta bildu egin gara.

7.3 Kalitatea

2.7 atalean aipatutako eraginkortasun adierazleak aztertzen ditugu jarraian, proiektuan lortutako kalitatea zein den ikusteko, hasieran zehaztutako irismenaren arabera.

Adierazle kuantitatiboak:

- Ordutegia mantentzea: proiektua plangintzan aurreikusitakoa baino lehenago bukatu dugu. Izan ere, oro har, atazak bakoitzarentzat jarritako muga baino lehenago amaitu ditugunez, hurrengo atazari ekin diogu eta horrek guztia aurreratzea eragin du.
  Hala ere, salbuespen bat izan dugu: SEP algoritmoaren inplementazioa. Kasu honetan, zenbait arazo izan ditugu eta, arazoa bilatzeko zailtasunen ondorioz, egun batzuetan zehar alde batera utzi dugu ataza, berriro hastean beste ikuspuntu bat izateko. Edonola ere, ataza batekin atzeratu arren, beste guztiekin aurreratu dugunez, proiektua adierazitako epeetan arrakastaz bukatu dugu.
- Lan egindako orduak: proiektua osatzeko 450 orduetatik, 430 aurreikusi ditugu plangintzan atazak egiteko eta 20 ordu koltxoi moduan gorde ditugu. Aurreko orrietan ikusi dugunez, aurreikusitako orduetatik zertxobait pasa gara, baina, hala ere, oraindik 9'5 ordu gelditzen zaizkigu defentsa prestatzeko. Ziurrenik, ordu gehiago beharko baditugu ere, %10eko desbiderapena egiteko beste 55 ordu sartu beharko

genituzke eta ez dugu hainbeste beharko, beraz, adierazle hau ere arrakastaz bete dugu.

- Proba-kopurua: hasteko, aipatu beharrekoa da proba arrakastatsu bat lortu ahal izateko, proba asko egin behar izan ditugula: emaitzak aztertzeko, egokia den edo ez jakiteko, moldaketak egiteko... Adierazle honetan, proba arrakastatsuak bakarrik zenbatu ditugu, besteak kontuan eramatea ezinezkoa baita. Honakoak izan dira, bada, egindako probak:

  1. Algoritmo partizionalaren lekuan, algoritmo hierarkikoa ordezkatzea
  2. *k*ren bilaketa automatikoa egiteko, SEP algoritmoa inplementatzea
  3. Profil antzekoenaren bilaketan, sekuentzien konparazioan medoidea moztea
  4. Profil antzekoenaren bilaketan, distantzia erlatiboa erabiltzea
  5. Profil antzekoenaren bilaketan, *cluster*-ak konbinatzea
  6. Sistemaren azken bertsioa *Discapnet* inguruneko datuekin probatzea

  Plangintzan, arrakasta izateko hiru eta zortzi proba artean egitea finkatu dugu, nahiz eta lehenengo probak egitean aldatzeko aukera irekita utzi dugun. Hala ere, ikus dezakegunez, sei proba egin ditugunez, arrakasta lortu dugu.
- Sistemaren eraginkortasuna: sistemaren azken bertsioak emaitzak behar bezala ematen ditu eta ez dauka exekutatzeko inongo arazorik. Hala ere, ikusi dugunez, aplikatzen ditugun datuen arabera, ordenagailu ahaltsuago baten beharra izan dezake sistemak.

Adierazle kualitatiboak:

- Kodearen argitasuna: azken bertsioan erabilitako aldagaien izenak jatorrizko sisteman dauden aldagaien izenen logika berari jarraituz sortu ditugu eta bakoitzak zer jasotzen duen adierazten du. Gainera, funtzioen izenak esanguratsuak dira; hortaz, bakoitzak zer egiten duen adierazten du. Bestalde, kodean iruzkinak jarri ditugu, baina hobe daiteke.
- Lortutako emaitzekin gogobetetzea: azken bertsioan hobekuntza esanguratsuak lortu ditugu. Ondorioak 8 kapituluan ikus daitezke garbiago. Edonola ere, sistemaren emaitzekin asetuta gelditu garela esan dezakegu.
- Lortutako esperientzia: proiektuan zehar, web meatzaritzari eta ereduen bilaketari buruz asko ikasi dugu, baita programazioari buruz ere. Gainera, ikasketa pertsonala egiten eta proiektu handi bat dokumentatzen ere ikasi dugu. Amaitzeko, ikerketa-talde baten parte izatea esperientzia ezinhobea izan da.

Adierazle garrantzitsuak:

- Zuzendariaren oniritzia: zuzendariak bere oniritzia eman du proiektuaren defentsa prestatzeko, beraz, proiektua amaitutzat eman dezakegu.

Oro har, proiektua arrakastatsua izan da; kalitatea neurtzeko adierazle gehienak erabat bete ditugu. Gainera, plangintzan adierazitako helburu nagusi guztiak bete ditugu, baita gehigarriak ere. Tamalez, ordea, ezin izan ditugu hautazko helburuak bete.

7.4 Arriskuak

Plangintzaren 2.7.2 atalean, batez ere, honako arrisku hauek aipatu ditugu:

- Hardware arazoak (sistema martxan dagoen makinan edo ordenagailu pertsonalean)
- Lana galtzea (kodea, memoria edota hori idazteko beharrezko materiala)
- Betebeharrak, helburuak edo irismena eguneratzea
- Ezusteko programazio arazoak
- Esperotako hobekuntzak ez lortzea
- Ingurune berrietako datuak ez lortzea

Hardware aldetik ez dugu arazorik izan, laborategiko ordenagailuak eta ordenagailu pertsonalak ez baitute inongo arazorik eman. Bestalde, lana ere ez dugu galdu, horretarako neurri asko hartu baititugu, segurtasun-kopiak maiz eta gailu desberdinetan eginez. Gainera, plangintzan ez dugu eguneraketa handirik egin, hortaz, helburuak betetzeko aukera izan dugu.

Hala ere, programazio arazo ugari izan ditugu, batez ere, SEP algoritmoa inplementatzean, aurretik aipatu dugun bezala. Zuzendariari eta laborategiko kideei laguntza eskatu arren, arazoaren iturria zein den identifikatzea asko kosta zaigu eta tartean denbora asko pasa dugu proba lagungarriak egiten. Arazoa egun batzuetan zehar alde batera uzteak eta artikulutik egindako aldaketak sakonago aztertzeak asko lagundu digu konponbide bat bilatzeko garaian.

Proba batzuetan, espero genituen hobekuntzak ez lortzea ere gertatu zaigu, adibidez, algoritmo hierarkikoak inplementatu ditugunean. Kasu honetan, beste hierarkiko batzuk probatzeko denborarik ez dugunez izan, proba nagusi honetan ez dugu hobekuntzarik lortu. Hala ere, ez da alferrikako lana izan: alde batetik, SAHN-ekin ez dela hobekuntzarik lortzen ikusteko balio izan digu, eta, bestetik, aurrerago SEP algoritmoa inplementatu ahal izateko.

Tarteko proba askotan ere gertatu zaigu emaitzetan aldaketarik ez lortzea, baina hurrengo proba arrakastatsua prestatzeko asko bideratu gaituzte; beraz, proba guztiak izan dira baliagarriak.

Bukatzeko, beste inguruneetako datuen arriskua daukagu zerrendatuta. *Discapnet* inguruneko datuekin lan egiteko aukera izan badugu ere, tamalez, ez ditugu *gipuzkoa.eus* ingurunearen datuak lortu. Hirugarren pertsonen lana denez, ezin izan dugu ezer egin. Plangintzan sarean bilatutako beste datu batzuk aplikatzeko aukera hausnartzea aipatu genuen. Horrek plangintza eguneratzea eta lan-ordu gehiago eskatzen dituenez, hautazko helburu hori bertan behera uztea erabaki dugu, dagoeneko proiektuak arrakasta lortu duela eta.

7.5 Ebaluazioa pertsonala

Atal honetan, proiektuaren egikaritzapenari eta egindako lanari buruzko ebaluazio pertsonala egin dugu.

- Egindako lanarekin gustora gaude. Proiektu luze bat aurrera eramateko gai izan gara, arazo handiegirik gabe, eta ateratako emaitzak positiboak izan dira, gainera, *ALDAPA* taldearentzat onuragarriak izan daitezke. Hala ere, oraindik hobekuntza asko daude egiteke sisteman, nahiz eta gure proiektutik kanpo gelditzen den.
- Egindako aldaketek probetarako ematen duten abaniko zabala kontuan hartuta, ezin izan ditugu nahi adina proba egin. Bestalde, SEP algoritmoaren eta COP indizearen inplementazioarekin aurreikusitakoa baino denbora gehiago pasa dugu, datuak zirela eta, algoritmoa ez delako datu-basera behar bezala egokitzen. Bukaeran lortutako emaitzekin, ordea, asetuta gelditu gara.
- Arazo larririk izan ez badugu ere, aurreko atalean ikusi dugun bezala, plangintzan aurreikusitako zenbait arrisku gertatu zaizkigu proiektuan. Hala ere, horientzako kontingentzia-plana prest zegoenez, oro har nahikoa izan da bertan azaldutakoari jarraitzea. Aurreikusi ez ditugun arazoen kasuan, zuzendariarekin bildu eta azkar eta arrakastaz jokatzen jakin dugu.

8. kapitulua

Ondorioak

Kapitulu honetan, proiektuan lortutako emaitzen berrikuspena egiten dugu. Gainera, etorkizunean egin litezkeen lana eta hobekuntzak aipatzen ditugu. Bestalde, ikasitako lezioak ere bertan aurki daitezke.

8.1 Lortutako emaitzak

Proiektuan zehar ateratako emaitzak ikusita, honakoak ondorioztatu ditugu:

SEP algoritmoa COP indizearekin erabiltzearen eragina

Lehenik, algoritmo erabat desberdina erabiltzeak hainbat onura ekarri dizkigu. Alde batetik, BTw-ko datuekin, emaitza hoberenetakoak lortu dituen algoritmoa izan da. Bestetik, *Discapnet* ingurunera oso ondo egokitu da. Hurrengo lerroetan ikus ditzakegu emaitzen alderaketak.

Clustering algoritmoaren eragina

Sistema, hasiera batean, PAM algoritmoarekin bakarrik zegoen inplementatuta. PAM algoritmo partizionala izanik, sistema algoritmo hierarkikoekin probatzea ideia ona iruditu zaigu, aldaketarik badagoen aztertzeko. 4 kapituluan ikusi ahal izan dugun moduan, emaitzak antzekoak izan dira, nahiz eta PAM-enak zertxobait hobeak diren. Hala ere, algoritmo hierarkikoek informazio gehiago ematen digute dendrogramaren bitartez, baita aurrerago SEP algoritmoa inplementatzeko aukera ere.

Nolanahi ere, guztiek dute hutsune bera: *k*-ren balioa doitu beharra.

Medoidearen luzeraren eragina

Medoide osoa sekuentziaren zati txiki batekin konparatzean, *cluster*-aren eta sekuentziaren arteko distantzia asko handitzen da, batez ere, medoidea luzea denean. Hau da, nahiz eta sekuentziaren zatia eta medoidearen hasierako zatia berdinak izan, medoidea oso luzea bada, distantzia asko handitzen da, unitate bat soberan daukan URL bakoitzeko. Aldiz, sekuentzia eta medoidea erabat desberdinak izanda ere, biak motzak badira, distantzia askoz txikiagoa da, eta hori aukeratzen da *cluster* hurbilenekotzat.

Hori dela eta, medoide osoarekin egindako proben emaitzak eta medoidea laburtuta lortutako emaitzak konparatzen baditugu, argi ikusi dugu hobekuntza nabarmena lortzen dela, proiektu osoan egindako nabarmenena. 8.1 irudian, ebaluazio-neurri bakoitzerako algoritmo bakoitzak lortzen duen hobekuntzaren ehunekoa adierazten da.

Bertsioa	F-measure %25	F-measure %50	MF-measure %25	MF-measure %50
PAM	%5.0	**%6.8**	%10.6	**%21.7**
SEP	**%5.6**	%5.9	**%11.0**	%17.8

Irudia 8.1: Medoidea moztean lortutako hobekuntzen ehunekoa.

Taulan letra lodiz markatuta dauden datuak begiratzen baditugu, sekuentziaren laurdena tratatzen dugunean SEP algoritmoak hobekuntza handiagoak lortu dituela ohar gaitezke, nahiz eta diferentzia ez den oso handia. Aldiz, sekuentziaren erdiarekin lan egitean, PAM algoritmoak lortu ditu emaitza hobeak, eta, oraingoan bai, aldea handiagoa da.

Distantzia erlatiboa erabiltzearen eragina

Medoide osoa sekuentziaren zati batekin alderatzen bada, distantzia absolutua erabiliz gero, aurreko atalean aipatu bezala, arazoak izan ditzakegu aukera egokiena egiteko. Medoidearen luzera moztu ondoren, ordea, distantzia erlatiboa erabiltzeak ez du hainbesteko eragina sisteman. Hala ere, emaitzak zertxobait hobetu dira, gutxi bada ere. Aldaketa oso txikia denez, ez ditugu emaitzen hobekuntzak ehunekotan erakutsi; azken bertsioarenak erakusten dira aurrerago.

KNN algoritmoan aukeratzeko irizpidearen eragina

Profil desberdinen konbinazioa egitean, aukeraketan, distantzia bakarrik hartzen du kontuan jatorrizko sistemak. Hurbileneko *cluster*-etik lehenengo lau proposamenak hartzen ditu, baina ez dute zertan egokienak izan. Ikusi dugun moduan, lehenengo bi *cluster*-etan dauden proposamenen *support*-a kontuan hartzeak hobekuntzak eragin ditu emaitzetan, hurbileneko bi *cluster*-etatik proposamen erabilienak hartzen ari baikara.

Bertsioa	F-measure %25	F-measure %50	MF-measure %25	MF-measure %50
PAM	**%16.1**	**%8.2**	%13.5	**%24.6**
SEP	%7.5	%7.5	**%14.6**	%22.1

Irudia 8.2: *Cluster*-en konbinazioa aldatzean lortutako hobekuntzen ehunekoa.

8.2 irudian, jatorrizko sistemarekin konparatuz, azken bertsioak, hau da, aldaketa guztiak dituen bertsioak, ebaluazio-neurri bakoitzean lortzen dituen hobekuntzen ehunekoa erakusten da, algoritmo bakoitzerako. 8.1 irudiko taulako emaitzekin alderatzen baditugu, SEP-en hobekuntza ez dela hain nabarmena ikus dezakegu. Atentzioa benetan deitzen duen kasu bakarra dago, PAM algoritmoak sekuentziaren laurdena ezagutuz aurreikuspena egiten duenekoa; izan ere, %11 inguru hobetzen da, orokorrean %3 inguru hobetzen badira ere.

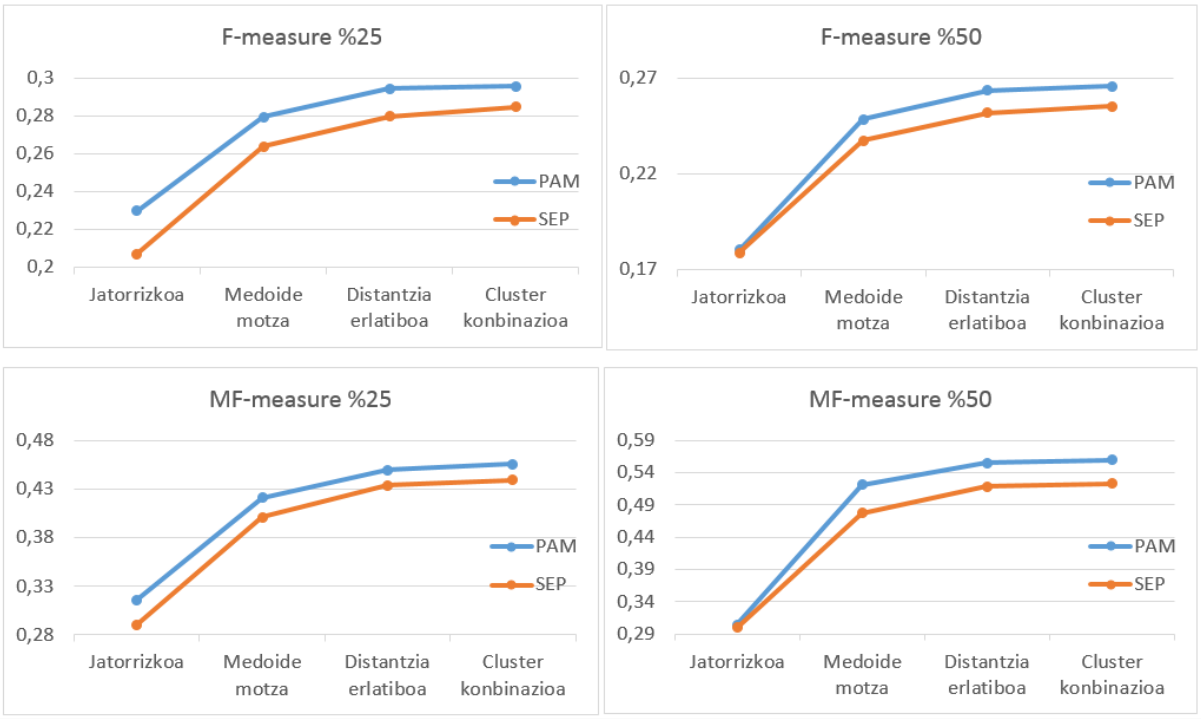
Bai jatorrizko sistemari eta bai azken bertsioari ingurune berriko datuak aplikatuta, batez ere, bi ondorio atera ditugu: alde batetik, datu berriekin lortutako emaitzak ez dira BTw-ko datuekin lortutakoak bezain onak izan, ez PAM eta ez SEP algoritmoarekin, datu horien ezaugarriak direla eta; bestetik, SEP-en emaitzak ia-ia PAM-en pare daudela ikusi dugu, inolako prestakuntzarik egin gabe. 8.3 irudian, sistemaren azken bertsioak ingurune bakoitzarekin testean lortutako emaitza hoberenak agertzen dira (SEP edo PAM-enak izan daitezke).

Ingurunea	F-measure %25	F-measure %50	MF-measure %25	MF-measure %50
BTw	0.2978075	0.26589862	0.45815358	0.55778354
Discapnet	0.2482710638	0.17195816	0.3467354425	0.4121842338

Irudia 8.3: BTw eta *Discapnet* inguruneetako datuekin lortutako emaitza hoberenen konparazioa.

Egindako aldaketa guztien ondoren lortutako hobekuntzak ikusteko grafiko batzuk prestatu ditugu. Lerro urdinak PAM algoritmoarekin inplementatutako bertsioarekin lortutako emaitzak adierazten ditu; lerro laranjak, berriz, SEP algoritmoarekin lortutako sistemarenak. Bestalde, zutabe bakoitzeko puntuek proba jakin horretan lortutako emaitza adierazten dute eta, noski, probak ordenatuta daude; beraz, lehenengo probatik azkenekora egin dugun hobekuntza adierazten dute lerroek. Grafikoaren ezkerreko ardatzean, lortutako emaitzen balioen eskala adierazten da. Amaitzeko, grafikoaren izenburuak zein ebaluazio-neurriri dagokion hobekuntza erakusten den jakinarazten du.

Hortaz, 8.4 irudian, goiko grafikoetan, aurreikuspenerako lortutako emaitzak ageri dira; behekoetan, eredu osoa kontuan hartuz lortutako emaitzak. Ezker aldeko grafikoek, sekuentziaren laurdena ezagutuz lortutako emaitzak erakusten dituzte; eskuinekoek, berriz, sekuentziaren laurdena ezagutuz lortutakoak. Edonola ere, grafiko bakoitzaren izenburuaren bitartez jakin dezakegu bakoitzak zein ebaluazio-neurri adierazten duen. Grafikoen adierazpena 6.7 irudikoen berdina da (ikus azalpenak 6.2 atalean).



Irudia 8.4: Proiektuan zehar, ebaluazio-neurrietan lortutako hobekuntzak.

Grafiko guztiek garbi erakusten dute hobekuntza esanguratsuena lehenengoa izan dela, hau da, aztertzen ari garen sekuentzia zatia medoide osoarekin konparatu ordez, medoidearen zati batekin konparatzea. Hurrengo probekin pixkanaka emaitzak hobetu

dira, baina aldaketa txikiagoa izan da batetik bestera.

8.2 Etorkizunerako lana

Proiektuaren iraupenean zehar ahal beste egin badugu ere, sistema hobetzeko aukera ugari daude. Beraz, atal honetan egiteke gelditzen den lana azaltzen dugu:

Distantziaren neurketa aldatzea

Jatorrizko sistemak *Edit* distantzia erabiltzen duenez, proiektu osoan zehar distantzia berarekin jarraitu dugu. Hala ere, etorkizunean probatzeko moduko aukera bat da distantzia globalak erabili ordez, distantzia lokalak erabiltzea (adibidez, *Smith-Waterman*). Distantzia lokalak erabiliz, antzekotasun-maila txikia duten eremuetako zailtasunari aurre egiten zaio. Hau da, distantzia lokalek eremu hauek saihesten dituzte eta erraztasuna duten eremuetan zentratzen dira. Ondoren, antzekotasuna batezbestekoa eginez kalkulatzen da.

SAHN algoritmoaren bertsioen jokabidea aztertzea

SAHN algoritmoarekin probak egin ditugunean aipatu dugun bezala, *average* eta *complete* bertsioek portaera berezia izan dute esplotazio atalean aldaketak egin ditugunean. Gainontzeko inplementazio guztiek hobekuntzak erakutsi badituzte ere, bi hauen kasuan emaitzak kaxkartu egin dira. Horregatik, kasu horiekin zer gertatzen den aztertzeke gelditzen da.

Doiketa hobetzea

Proiektuan zehar ikusi dugunez, k-ren bilaketa automatikoa egiteko inplementatu dugun SEP algoritmoa oso ondo egokitu da sistemara. Nolanahi ere, jatorrizko algoritmoan zenbait aldaketa egin behar izan ditugu. Egin ditugun probekin zentzuzko emaitzak lortu baditugu ere, ez dugu PAM algoritmoarekin lortutako emaitzak parekatzea edo gainditzea lortu. Hortaz, etorkizunean egin daitezken lanen aukera bat izan daiteke moldaketa hori gehiago aztertzea, emaitzak hobetzeko bidea izan baitaiteke.

SPADE ordezkatzea

Proiektuaren hasieran adierazi dugun bezala, profilak sortzeko prozesua ez dakigu erabat egokia ote den. Horretarako, sisteman SPADE algoritmoa erabiltzen da. Hala ere,

interesgarria izan daiteke prozesua pausoz pauso aztertu eta, sistemaren ezaugarriak kontuan hartuta, algoritmo egokiagorik badagoen ikertzea. Kasua balitz, inplementatu eta probak egin genitzake, ondorioak ateratze aldera.

KNN atala

Sistemak sekuentzia baten profil antzekoena bilatzeko KNN sailkatzailea erabiltzen du. Atal hau aztertu ondoren, zenbait ahulezia aurkitu ditugu eta horiek indartzeko aldaketa batzuk ere egin ditugu. Ez dira, ordea, egin daitezkeen aldaketa guztiak, ahulezia gehiago bila baititzakegu ziurrenik, eta, hori gutxi balitz, gainbegiratutako sailkapeneko beste algoritmo batekin sailkatzea ere proba dezakegu.

Probak behar bezala egitea

6 atalean ikusi dugunez, proben egokitasuna ez da erabatekoa izan, edo, behintzat, ezin izan ditugu nahi bezala egin. Hori dela eta, egin beharreko lan moduan gelditzen da baliabide hobeak izanik (ordenagailu ahaltsuagoak eta denbora gehiago, batez ere), *Discapnet* inguruneko datu-basearekin egindako probak errepikatzea, baina, kasu honetan, balidazio gurutzatua behar bezala eginez, datu-base osoarekin.

Horretaz gainera, jatorrizko programa azter dezakegu, memoriaren erabilera optimizatzeko asmoz, agian baliabide hobeen beharrik ere ez baitago problema handiagoak exekutatu ahal izateko.

8.3 Ikasitako Lezioak

Proiektuaren bizi-zikloan zehar ikasitako lezioak bi multzotan bana ditzakegu: orokorrak, informatikako proiektuetarako baliagarriak direnak, eta zehatzak, gure proiektuaren edukiei buruz ikasitakoak.

Orokorrak

Atal honetan, informatikako proiektuekin izan ditzakegun arazo orokorretatik ikasitako lezioak zerrendatzen ditugu.

- Informatikako proiektu guztietan bezala, programazioaren punturen batean blokeatuta gelditzeko arriskua dago. Konpontzeko modu bakarra denbora eskaintzea eta

kideei laguntza eskatzea da. Gure kasuan izan den bezala, ordea, baliteke kideek ere ezer ez ikustea eta, ondorioz, berriz leku berean gelditzea.
Hori dela eta, arazoa denbora labur batez alde batera uztea eta ondoren berriro hartzea lagungarria da, baita arazotzat identifikatzen ditugun atal txikiak berriro hutsetik inplementatzea ere. Esan beharrik ez dago proba desberdinak egiteak asko laguntzen duela, nahiz eta jakin proba horiek ez direla emaitza egokiak ematen dituztenak, arazoaren iturria bilatzen laguntzen baitute.

- Memoria idazteko garaian, nahiz eta guztia dokumentatuta izan, testu guztia egituratzea kostatzen da. Horregatik, prozesu osoa zein den badakigunez, memoriari ekin baino lehen, esan beharrekoaren egitura zehaztea komeni da. Modu honetan, atal bakoitzean zer sartu behar dugun jakinda, askoz errazagoa da erredakzioa egitea, gauza bakoitza behin eta berriz lekuz aldatzen ibiltzea baino.
- Hilabeteetan zehar osatzen den proiektu bat denez, hasieratik eguneroko bat eramatea lagungarria izan da, batez ere, memoria idaztean sortutako zalantzak argitzeko. Hala ere, egunerokoa nahiko baldar idatzi dugu eta, ondorioz, zerbait zehatza bilatzeko momentuan zailtasunak izan ditugu. Horregatik, egunerokoa modu ordenatuago batean idaztea komeni da, edo, behintzat, egun bakoitzeko informazioa pilatzeko estandar bat finkatzea (izenburua, data, hitz gakoak eta informazioa, adibidez).
- Ezagutzen ez dugun sistema berri batekin lanean hasi behar dugunean, beharrezkoa da lehenik sistemaren ideia orokor bat egitea. Gainera, sistemak erabiltzen dituen algoritmoak lantzea nahitaezkoa da hasierako probak jarraitu eta funtzionamendua ulertu ahal izateko. Edonola ere, oinarri txiki bat hartu ondoren, sistema jakin baten barneko prozesua ikasteko modurik onena datu-base txiki batekin probak egitea eta pausoz pauso *debugger*-arekin jarraitzea da, momentu bakoitzean aldagaien balioak aztertuz eta agindu bakoitzean zer aldatzen den ikusiz.

Zehatzak

Hurrengo lerroetan, proiektu honetako edukiei buruz ikasitakoak aipatzen dira.

- Proiektu osoa kontuan hartuta, datu-meatzaritzako (web meatzaritzako) problema oso bati aurre egiten ikasi dugu, datuen biltzea eta aurreprozesaketa kenduta.
- Proba desberdinak egiteko, *clustering* algoritmo desberdinen ezaugarriak aztertu ditugu. Algoritmo guztiak elkarren oso desberdinak izan dira, batzuk partizionalak eta besteak hierarkikoak zirenez gero. Gainera, guztien funtzionamendua ezberdinak denez, banaka-banaka landu ditugu. Jarraian, sistemarako egokiak diren edo

ez eta arazoak non sor daitezkeen ikertu behar izan dugu, zenbait kasutan, baita moldaketak egin ere.

- *Clustering* atala lantzerakoan, algoritmoez gain, beste bi gauza ere ikasi ditugu: alde batetik, k egokiena bilatzeak zer nolako zailtasuna duen konturatu gara, eta, bestetik, datuek sisteman zenbateko eragina daukaten ikusi dugu. k egokiena bilatzeko. k-ren bilaketari aurre egitea lortu dugu proiektuan, baina, datuen eraginari dagokionez, ez dago ezer egiterik.
- Balidazioari dagokionez, CVI-ak zer diren eta nola funtzionatzen duten aztertu dugu. Horretarako beharrezkoa izan da CVI desberdinen inguruko artikuluak irakurtzea eta bakoitzaren lan egiteko modua ulertzea.
- Web meatzaritzako sistemak ebaluazioa egiteko erabil ditzakeen bi metodologiak, balidazio gurutzatua eta *holdout* teknika, sakonago aztertu ditugu, bakoitzaren ezaugarriak, berezitasunak, abantailak eta desabantailak ikusiz eta bi metodologiak elkarrekin konparatuz.

Bibliografia

[Arbelaitz, 2013] Arbelaitz, O. (2013). Web usage and content mining to extract knowledge for modelling the users of the bidasoa turismo website and to adapt it. *Expert Systems with Applications*, 40(18?):7478–7491.

[Chordia, 2011] Chordia, B., A. K. (2011). Grouping web access sequences using sequence alignment method. *Indian Journal of Computer Science and Engineering (IJCSE)*, 2:308–314.

[Dasarathy, 1991] Dasarathy, S. (1991). *Nearest neighbor (NN) norms: NN pattern classification techniques*. IEEE Computer Society Press, New York, NY, USA.

[Gurrutxaga, 2010] Gurrutxaga, I. (2010). SEP/COP: An efficient method to find the best partition in hierarchical clustering based on a new cluster validity index. *Pattern Recognition*, 43(10?):3364–3373.

[Gusfield, 1997] Gusfield, D. (1997). *Algorithms on Strings, Trees, and Sequences - Computer Science and Computational Biology*. Cambridge University Press, New York, NY, USA.

[Kaufman, 1990] Kaufman, L., R. P. J. (1990). *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley-Interscience.

[O. Arbelaitz, 2014] O. Arbelaitz, A. Lojo, J. M. I. P. (2014). Global versus link prediction approach for discapnet: website focused to visually impaired people. *Preprints of the Federated Conference on Computer Science and Information Systems*, pages 51–58.

[William H. E. Day, 1984] William H. E. Day, H. E. (1984). Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of Classification*, 1(1?):7–24.

[Zaki, 2001] Zaki, M. J. (2001). Spade: An efficient algorithm for mining frequent sequences. *Machine Learning*, 42:31–60.