EUSKAL HERRIKO UNIBERTSITATEA
Lengoaia eta Sistema Informatikoak

Doktorego-tesia

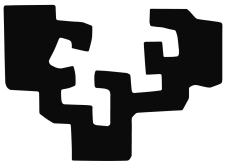
Integrazioa hizkuntzaren prozesamenduan

Anotazio-eskemak eta elkarreragingarritasuna. Testuen prozesatze masiboa, datu handien teknikak erabiliz.

Zuhaitz Beloki Leitza

Donostia, 2017

eman ta zabal zazu



EUSKAL HERRIKO UNIBERTSITATEA

Lengoaia eta Sistema Informatikoak

Integrazioa hizkuntzaren prozesamenduan

Anotazio-eskemak eta elkarreragingarritasuna. Testuen prozesatze masiboa, datu handien teknikak erabiliz.

Zuhaitz Beloki Leitzak Xabier Artola Zubillagaren eta Aitor Soroa Etxaberen zuzendaritzapean egindako tesiaren txostena, Euskal Herriko Unibertsitatean Doktore titulua eskuratzeko aurkeztua.

Donostia, 2017.

Laburpena

Tesi-lan honetan hizkuntzaren prozesamenduko tresnen integrazioa landu dugu, datu handien teknikei arreta berezia eskainiz. Tresnen integrazioa, izatez, bi mailatan landu dugu: anotazio-eskemen mailan eta prozesuen mailan.

Anotazio-eskemen mailako integrazioan tresnen arteko elkarreragingarritasuna lortzeko lehenbiziko pausoak aurkeztea izan dugu helburu. Horrekin lotuta, bi anotazio-eskema aurkeztu ditugu: Anotazio-Amaraunen Arkitektura (AWA, *Annotation Web Architecture*) eta NLP Annotation Format (NAF). AWA tesi-lan honekin hasi aurretik sortua izan zen, eta orain formalizaziolan bat egin dugu berarekin, elkarreragingarritasunari arreta berezia jarriz. NAF, bere aldetik, eskema praktikoa eta sinplea izateko helburuekin sortu dugu. Bi anotazio-eskema horietatik abiatuz, eskemarekiko independentea den eredu abstraktu bat diseinatu dugu. Abstrakzio horri esker, elkarreragingarritasunerantz jotzeko bidea zabaldu nahi izan dugu, eredu abstraktua edozein eskemarekin bateragarria dela argudiatuz.

Bestalde, tresnen prozesu mailako integrazioa ere landu dugu. Horretarako, analisi-kateak modu malguan eta deklaratiboan eraikitzeko azpiegitura bat diseinatu eta inplementatu dugu. Gainera, azpiegitura horretan oinarrituz eta datu handien teknikak aplikatuz, testu-dokumentuen bilduma erraldoiak modu banatuan eta eskalagarrian prozesatzeko arkitektura bat diseinatu eta inplementatu dugu. Sistema hori hainbat nodoz osatutako terminal talde batean ezarriz, bai analisi-kateko tresnak eta bai prozesatu beharreko dokumentuak, automatikoki, eskura dauden nodoetan zehar banatuko dira, sistema osoaren ahalmenari ahalik eta etekin handiena ateraz.

Eskerrak

Doktorego-tesia amaitzea beti izaten omen da lan nekeza. Lanak aurrera doazela dirudienean ere, urrun ikusten da aurkezpenaren ondorengo luncharen une preziatua. Bide luze horren helmugara iristea ez litzateke posible izango bidean hainbesteko laguntza eman didatenengatik izan ez balitz. Pertsona horien omenez idatzi dut orrialde hau, emandako laguntza nolabait eskertzeko asmoz.

Eskerrik beroenak bide osoan zehar gidari izan ditudan zuzendariei eman nahi dizkiet. Aitor eta Xabier, tesiaz gain ere urte mordoxka eman dut zuekin, eta esker oneko hitzak besterik ez ditut zuentzat. Zuen esperientzia eta jakinduria ezinbestekoak izan dira lan hau bide onetik eramateko, baina bereziki azpimarratu nahi dudana zuen aldetik jaso dudan tratua da. Ideiak proposatuz, inoiz ez aginduz, eta beti laguntzeko prest. Xabier, ez dut aipatu gabe utzi nahi zure zuzenketa zehatzei esker euskaraz idazten ikasi dudan guztia ere. Eskerrik asko bioi!

Horrekin batera, Ixa talde osoari eman nahi dizkiot eskerrak. Arantza, zuri ere eskerrak eman beharrean nago, lanez lepo egonda zuk ere denboraldi batean zehar zuzendariaren papera bete baituzu. Arantxa, Olatz, Josu, Iñigo, Manex, Itziar... bulegokide izan zareten guztiei, zuek bai pertsona jatorrak! Eskerrik asko bulegoan izan dugun giro paregabeagatik. Lana egin behar zenean isiltasunez, eta deskonektatzeko beharra egon denean hitz eginez edo Tourreko etapen amaierak ikusiz! Iñigo, baita jolastu ditugun pilota eta ping-pong partida guztiengatik eta bizikleta-buelta guztiengatik ere. Eskerrik asko denoi! Gainerako Ixakide guztiei ere eskertu nahi nieke urte hauetan guztietan hor egon izana, jende jator asko ezagutu baitut taldean. Kike,

zure izena bereziki aipatu nahi nuke, zerbitzari, tresna, aplikazio eta beste edozein konturekin izandako arazoen aurrean emandako laguntza ez baitago neurtzerik ere, merezita daukazu taldean duzun izen ona!

Elhuyar Fundazioa ere ez nuke aipatu gabe utzi nahi. Josuri eta gainontzeko guztiei, tesiak iraun duen bitartean ehunetik ehun eman ezingo nuela jakinda ere nirekin kontatzeagatik, eta lanerako eman dizkidazuen erosotasun guztiengatik, mila esker!

Azkenerako utzi bazaituztet ere, garrantzitsuenak, oraingo honetan ere, etxekoak izan zarete. Aita, ama, tesia idatzi izanak eman didan aukera hau aprobetxatu nahi nuke zuei eskerrak emateko. Tesiaren azken txanpan ia nik baino gehiago *sufritu* duzue, bizitzaren beste arlo guztietan bezalaxe. Nonbait guraso eredugarririk bada, horixe zarete zuek. Aiuri, gauza bera zuri ere, familian egin ditugun bazkari-afari horiei guztiei esker errazagoa izan baita tesiarekin aurrera egiteko indarrak ateratzea. Eta nola ez, egunero nire ondoan dagoen Naiarari, egun osoa lanean pasa dudanetan ere etxera iritsi eta zu ikusteak eguna alaitu izan baitit. Eskerrik asko guztioi!

Esker instituzionalak

Eskerrak Euskal Herriko Unibertsitateko Euskararen arloko Errektoreordetzari, tesi-lan hau euskaraz egiteko emandako bekarengatik.

Aurkibidea

I Sarrera

1 Sarrera **3**
1.1 Anotazio-eskemen mailako integrazioa: anotazio-eskemak eta elkarreragingarritasuna 6
1.2 Prozesu mailako integrazioa: testuen prozesaketarako arkitektura banatua 9
1.3 Tesiaren kokapena eta ekarpen nagusiak 12
1.4 Tesiaren egitura 13
1.5 Argitalpenak 13

II Tresnen anotazio-eskemen mailako integrazioa

2 Informazio linguistikoaren adierazpen-ereduak: arloaren egungo egoera **17**
2.1 Sarrera eta kokapena 17
2.2 Anotazio-eredu abstraktuak 19
2.3 Datu estekatuak eta informazio linguistikoa 26
2.4 Elkarreragingarritasuna 28

3 Anotazio-eskemak: Anotazio Amaraunen Arkitektura eta NAF **31**
3.1 Anotazio Amaraunen Arkitektura 31

3.1.1 AWAren datu-eredua 32
3.1.2 AWAren anotazio-eskema 46
3.2 NLP Annotation Format (NAF) 58
3.2.1 Anotazioen arteko erreferentziak eta aingurak 61
3.2.2 Anotazio txertatuak 63
3.2.3 Anotazioen identifikadoreak 65
3.2.4 NAF eta datu estekatuak 65
3.2.5 NAFen ekosistema 66

4 Anotazio linguistikoen arteko elkarreragingarritasuna 69
4.1 Elkarreragingarritasunaren arazoa 69
4.2 Elkarreragingarritasunaren bila 72
4.2.1 RDF eta OWL 75
4.2.2 Anotazio-eskemen abstrakzioa 78
4.2.3 AWA eta NAF anotazio-eskemak eredu abstraktuaren arabera mapatzen 81

III Tresnen prozesu mailako integrazioa

5 Datu handien teknikak hizkuntzaren prozesamenduan: arloaren egungo egoera 91
5.1 Prozesaketa-ereduak 91
5.2 MongoDB datu-baseak 93
5.3 Prozesaketa banaturako teknologiak 99
5.3.1 MapReduce 99
5.3.2 Apache Hadoop 102
5.3.3 Apache Apex 104
5.3.4 Apache Twill 104
5.3.5 Facebook Corona 105
5.3.6 Apache Spark 107
5.3.7 S4: stream-konputazio banaturako plataforma 108
5.3.8 Apache Storm 111
5.3.9 Apache Ignite 114

5.3.10 Apache Flink 115
5.3.11 Google Cloud Dataflow 116
5.4 Hizkuntzaren prozesamendua ingurune banatuetan 117

6 Hizkuntzaren prozesamendu masiborako arkitektura bat 123
6.1 Sarrera eta motibazioa 123
6.2 Hizkuntzaren prozesamendurako analisi-kateak 125
6.3 Sistemaren arkitektura 126
6.3.1 Nodo nagusia 128
6.3.2 Nodo langileak 129
6.3.3 Datuen fluxua nodoetan zehar 131
6.3.4 Datu-basearekiko integrazioa 135
6.3.5 Topologien definizioa 138
6.3.6 Sistema hutsetik ezartzen 140
6.3.7 Esperimentuak eta emaitzak 141

IV Ondorioak

7 Ondorioak eta etorkizuneko lanak 159
7.1 Ekarpen nagusiak 159
7.2 Ondorioak 160
7.3 Etorkizuneko lanak 165

Irudien zerrenda

1.1 Analisi-kate sinple baten irudia. Testua lau maila linguistikotan prozesatzen duten HPek osatzen dute katea. Bakoitzaren emaitza hurrengo HPak jaso eta erabiltzen du. 4

1.2 Analisi-kateko prozesatzailek itzulitako informazio linguistikoa adierazteko beharrezkoa da anotazio-eskema bat jarraitzea. Irudiko eskema adibide gisa besterik ez dugu erabili, ez da tesian zehar garatu ditugunetako bat. 7

2.1 Anotazio linguistiko baten adibidea. 18

3.1 AWAren datu-ereduaren diagrama. 33

3.2 Anotazio linguistikoen itxura orokorra esaldi oso baten gainean (anotazioak gorriz, aingurak berdez eta anotazioen informazio linguistikoa urdinez). 36

3.3 Bi dependentzia-anotazioren irudikapen grafikoa. Bi anotazioen burua *gainditu* da, eta modifikatzaileak *Mikelek* eta *azterketa*. 37

3.4 Dependentzia-anotazioen adibidea. 37

3.5 TEIk proposatutako ezaugarri-egiturak erabiliz, horrela adierazten dira ezaugarri linguistikoak AWAn. 39

3.6 *txoriak* tokenaren bi interpretazio morfosintaktikoak. Horrela adierazten da AWAn anbiguotasuna, aingura berari hainbat analisi esleituz. 42

3.7 *publikoak* hitzaren interpretazio posibleak. Erabilitako notazioa: *Aingura::Anotazioa(esteka)::Informazio linguistikoa*. . . 44

3.8 Lematizazioaren irteera *Mendizorrotzako ekitaldi publikoak ikusi ditu* esaldia analizatuta. 45

3.9 Lematizatzaileak sortutako interpretazioak multzokatzen. . . 45
3.10 Interpretazio multzoen sekuentzien bidez, hurrengo pausoetan erabiliko diren interpretazio-aingura konplexuak osa daitezke. 46
3.11 Anotazioen aingurek informazio linguistikoa zein elementuri buruz ematen den zehazten dute. 48
3.12 Datu-ereduaren eskema orokorra, ainguretan zentratuz. 48
3.13 AWAren ainguren eskema, aingura konplexuak, eta zehazki TextRef aingurak, gehituta. 49
3.14 AWAren ainguren eskema, spanak gehituta. 50
3.15 AWAren ainguren eskema, erlazio-aingurak gehituta. 50
3.16 Testu-dokumentu mailako serializazioaren eskema. Inplementazio hau XML dokumentuetan oinarritzen da. 52
3.17 Corpus mailako serializazioaren eskema. Inplementazio hau datu-baseetan oinarritzen da. 57
3.18 NAF dokumentu baten adibidea. 60
3.19 Dependentzia-anotazioek terminoei egiten diete erreferentzia, identifikadoreak erabiliz. 61
3.20 `span` elementuekin anotazio zerrenda bati egiten zaio erreferentzia. Adibide honetan, Paul Newman entitateak Paul eta Newman terminoei, biei, `span` bakarrarekin egiten die erreferentzia. 63
3.21 Tokenek (`wf`) offsetak erabiliz egiten diete erreferentzia dagozkien testu zatiei. 64
3.22 Batzuetan, anotazioek elkarren erreferentzia gorde ordez, bata bestearen barruan definitzen dira, haien arteko lotura estua den kasuetan. Horrela gertatzen da, irudian ikusten den bezala, rol-anotazioekin. 64
3.23 NAFen erraza da anotazio bat kanpoko ezagutza-base batekin lotzea. Irudiko adibidean, *Paul Newman* entitatea DBPedia Spotlighteko Paul Newman aktorearekin eta Paul Newman musika taldearekin lotu da. Konfiantza balioaren bidez, testuinguru jakin horretan aktoreari dagokiola adierazi da. . . . 66
3.24 Hainbat `externalRef` elementurekin, anotazio bat ezagutza-base bat baino gehiagorekin lotu daiteke. 67

4.1 Elkarreragingarritasun estrukturalaren arazoa ilustratzeko adibidea. Informazio morfologikoa adierazteko modua oso desberdina da NAFen eta AWAn. NAFen oinarrizko informazio morfologikoa ematen da: lema (`lemma`), kategoria (`pos`) eta bestelakoa (`morphofeat`). AWAren kasuan, aldiz, hitzaren informazio morfologiko aberatsaz gain, oinaren eta morfemena ere ematen da. 73
4.2 Irudiko bi anotazioek, informazio berdina ematen duten arren, itxura desberdina dute, bai egitura aldetik, bai kontzeptuen izendatze aldetik, eta bai erabilitako adierazpide fisikoaren aldetik (XML eta JSON). 74
4.3 RDF hirukote bat. 75
4.4 RDF hirukotez osatutako ontologia oso sinplea. Ontologia horren arabera, hitz batek kategoriak eta morfemak izan ditzake. Domeinua kontuan hartuta, badakigu hitzek kategoria bakarra izaten dutela, baina RDF hutsez ezin da horrelako kardinalitaterik zehaztu. 76
4.5 Eredu abstraktuaren irudikapena. Anotazioek aingura bat eta informazio linguistikoa daukate lotuta, eta anotazio bakoitzak, era berean, aingura izaera ere badauka, hurrengo mailako anotazioen aingura izan baitaiteke. 79
4.6 Eredu abstraktua, osorik. 80
4.7 Anotazio linguistikoen itxura orokorra esaldi oso baten gainean (anotazioak gorriz, aingurak berdez eta anotazioen informazio linguistikoa urdinez). 81
4.8 AWA anotazio-eskema eredu abstraktura egokituta 84
4.9 NAF anotazio-eskema eredu abstraktura egokituta 86

5.1 BSON dokumentu baten adibidea. XMLn bezala, egitura hierarkikoak eraiki daitezke. 95
5.2 MongoDBko dokumentu bat erreferentzia bidez adierazia. Izatez, lau dokumentutan gorde da, morfema bakoitzeko bana eta hitz osoarentzat bestea. Ondoren, hitzaren dokumentu nagusian, morfemen erreferentziak gorde dira, identifikadoreen bidez. 96

5.3 Hainbat dokumentu jasota, hitz bakoitzaren agerpen kopurua kontatzen duen MapReduce programa. Irudiko bi dokumentuen artean hiru lerro osatzen dituztenez, *map* funtzioaren hiru exekuzio abiaraziko dira. Adibidean 7 hitz desberdin daudenez, *reduce* funtzioa hainbeste aldiz exekutatuko da. 101

5.4 Hadoopen arkitektura. Nodo nagusiak, MapReduce funtzioen exekuzioak kudeatzeaz gain (ataza kud.), exekuzioak ere egiten ditu (ataza exek.). HDFSri dagokionez, izenen nodoa, datuen nodo bakoitzean zein datu aurkitzen diren adierazten duena, nodo nagusian aurkitzen da. 103

5.5 Hadoopek ataza-kudeatzaile bakarra ezartzen du terminal taldeko ataza guztiak banatzen eta ataza-exekutatzaile guztiak gainbegiratzen. 106

5.6 Coronan, ataza mota bakoitzak bere kudeatzaile propioa dauka. Horrela, terminal taldearen kudeatzailearen lana oso arina da, atazen oso azaleko kudeaketa besterik ez baitu egin behar. 107

5.7 Sparkek ere streaming-prozesaketa egin dezake, Spark Streaming izeneko hedapenari esker. 108

5.8 S4 programa baten adibidea. Programak, datu-korronte mugagabe batetik, testu-dokumentuak jasotzen ditu, eta dokumentu guztien artean agerpen gehien izan dituzten hitzen zerrenda osatzen du denbora errealean. 110

5.9 Storm programen bi topologia posible. Lehena lineala da eta bigarrena ez-lineala. 111

5.10 Gehien agertu diren hitzak ordenatuta zerrendatzeko adibidearen Storm topologia lineala. 113

5.11 Dokumentuen prozesaketa lau HPko analisi-kate batekin egiten duen topologia ez-lineala. 114

6.1 Sistemaren arkitektura adierazten duen irudia. Sistema hainbat nodoz osatzen da, nodo nagusi bat eta hainbat nodo langile. Nodo nagusia sistemaren kudeaketaz arduratzen da, eta nodo langileak prozesaketa gauzatzeaz. 128

6.2 Datuen fluxua horrelakoa litzateke datu-baserik erabiliko ez bagenu. HP bakoitzak, berak sortutako geruzaz gain, jasotakoak ere bidaliko lizkieke hurrengoei, bestela bidean galduko lirateke eta. 132
6.3 Datuen fluxua, MongoDB datu-basea gehitu ondoren. HP batetik bestera bidaltzen den bakarra NAF dokumentuaren identifikadorea da. Horrekin, HPek datu-basetik eskuratzen dituzte beharrezko dituzten NAF geruzak. HP bakoitzak zein geruza behar dituen kanpoko fitxategi batetik (HP konf.) irakurtzen du. 134
6.4 Topologia zati baten definizioaren adibidea. 138
6.5 Exekuzio paraleloa dela eta, prozesamendu-denbora eta igarotako denbora desberdinak dira. Irudiko exekuzioaren prozesamendu-denbora 8 minutukoa da, dokumentu bakoitza prozesatzen emandako denborak metatu egiten baitira. Igarotako denbora, berriz, 4 minutukoa da, pasatutako denbora erreala baita horretarako kontuan hartzen dena. 145
6.6 Bost dokumenturen bi prozesaketa posible denboran zehar nola exekutatu diren adierazten duten bi eskema. 147
6.7 Esperimentuetan erabilitako ingeleserako topologia ez-lineala. 153
6.8 Dokumentu bat topologia ez-linealarekin prozesatuta, prozesaketa-denboraren banaketa HPen artean nolakoa izan den adierazten duen eskema. 154
6.9 Egindako azterketa teorikoaren arabera, grafika honetan ikus daitezkeen hobekuntzak lortuko genituzke PUZ-nukleo kopuru mugagabearekin, sarrerako testuak esalditan zatituko bagenitu. Grafikan ikus daiteke prozesatzen igarotako denbora nola hobetzen den dokumentuen batezbesteko esaldi kopuruak gora egin ahala. 155

Taulen zerrenda

3.1 AWAren XML serializazioaren arabera sortzen diren XML dokumentuak. Anotazioen osagai bakoitza zein fitxategitan aurkitzen den ikus daiteke. 53

6.1 Esperimentuetan erabilitako HPen zerrenda, bakoitzaren sarrera eta irteerako anotazio-geruzak ere adieraziz. † ikurra daukaten HPek bezero-zerbitzari eredua jarraitzen dute. . . . 143
6.2 Esperimentuetan erabilitako ingelesezko eta gaztelaniazko dokumentu multzoak. Dokumentu multzo bakoitzaren hitz eta esaldi kopurua, batezbestekoa (μ) eta desbideratze estandarra (σ) adierazi ditugu. 144
6.3 en100 dokumentu multzoa prozesatzen emandako denborak 6 nodo langilez osatutako terminal talde batean. Batch ereduari jarraituz egin dira prozesaketa guztiak. Hainbat ezarpen probatu ditugu batch-prozesaketarako portaerarik onena zeinek daukan ikusteko. Denborak minututan neurtuta daude. 150
6.4 Batch eredua jarraituz egindako prozesaketen estatistikak. Denborak minututan neurtuta daude. 150
6.5 Streaming-prozesaketako esperimentuen prozesaketa-denbora eta latentziak, topologia lineal eta ez-linealekin. Denborak minututan neurtuta daude. 152

I Atala

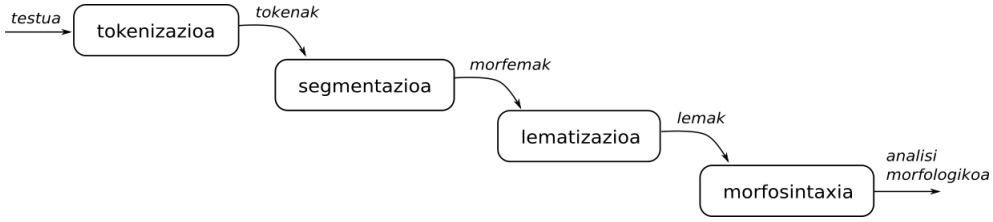
SARRERA

1. KAPITULUA

Sarrera

Aginduak interpretatzen dituzten edota hitz egiten duten robotak, webeko edukien bilatzaileak, itzultzaile automatikoak, produktu baten inguruko erosleen iritzietatik automatikoki erauzitako balorazio-txostenak, galdera-erantzuneko sistemak... denak baliatzen dira hizkuntzaren prozesamendua delako arloaren ekarpen zabalaz. Hizkuntzaren prozesamenduari esker, makinei hizkuntza erabiltzen irakasten zaie, hizkuntzarekin erlazionatutako aplikazioak garatu eta gizakioi mota askotako lanak errazteko (Chowdhury, 2003). Aplikazio batzuen kasuan, makinek testuak ulertzea nahi dugu (Manning, 2016), edo ahotsa bera ulertzea (Rabiner eta Juang, 1993). Horretaz baliatuz, testu bilduma erraldoietan aurkitzen den informazioa oso azkar prozesatu eta interesatzen zaigun informazioa erauztea lor dezakegu (Agerri *et al.*, 2014a; Banko *et al.*, 2007). Beste batzuetan, makinek hitz egitea lortzea da helburua (Dutoit, 1997). Askotan, makinek testu idatziak sortzea nahi dugu, galdera bat idatzi eta erantzuna eman behar digutenean (Ferrucci *et al.*, 2010), adibidez; eta, beste askotan, testuak landu eta eraldatzea izaten da gure nahia, adibidez, hizkuntza batean idatzitako testua beste hizkuntza batera automatikoki itzultzea nahi dugunean, edo testu konplexu bat automatikoki laburtu eta idazkera sinpleagoan jaso nahi dugunean (Aranzabe *et al.*, 2012). Hizkuntzaren prozesamenduaren aplikazioak dira horiek denak.

Makinen prozesagailuek ez dute giza garunaren moduan funtzionatzen. Hori horrela, teknika konplexuak erabiltzen dira makinei hizkuntzak erabiltzen irakasteko. Gehien erabilitako tekniketako bat estatistikan oinarritutakoa



1.1 irudia: Analisi-kate sinple baten irudia. Testua lau maila linguistikotan prozesatzen duten HPek osatzen dute katea. Bakoitzaren emaitza hurrengo HPak jaso eta erabiltzen du.

da (Manning *et al.*, 1999; Koehn *et al.*, 2007). Kasu horretan, corpusak ezinbesteko tresna dira ikertzaileentzat, corpusetan testu bilduma erraldoiak biltzen baitira. Corpusa egokia bada, metodo matematikoz baliatuz, makina gai izan daiteke testu horietatik hizkuntza *ikasi* eta, ondoren, erabiltzeko.

Hala ere, aplikazio baten baitan testu hutsa bere horretan erabiltzea ez da oso erabilgarri suertatzen. Informatikaren esparruan hain ohiko den metodoa erabiltzea ezinbestekoa izaten da hizkuntzaren prozesamenduan ere, alegia, lan konplexu bat ataza txikiagoetan zatitzea (Agerri *et al.*, 2014b). Horrela, corpusekin eta beste edozein testu zatirekin egiten den lehenbiziko gauza analisi-kate batekin prozesatzea izan ohi da (ikus 1.1 irudia). Analisi-kateak askotarikoak izan daitezke, baina, funtsean, testuak hainbat maila linguistikotan prozesatzea izaten da haien zeregina, morfologia edo sintaxi mailako analisia eginez (Aduriz *et al.*, 1999), adibidez. Testu bat prozesatu ondoren, testu horren gaineko informazio linguistiko zabalagoa izango dugu eskura: hitz bakoitzaren informazio morfologikoa, esaldien sintaxiaren gaineko informazioa, terminoen semantika edo esanahiarekin lotutakoa etab. Egoera horretan, hizkuntzaren prozesamenduko edozein aplikazio inplementatzeko zeregina askoz ere errazagoa izaten da.

Hala ere, analisi-kate baten aplikazioan hainbat konplikazio agertzen dira. Maila linguistiko bakoitzaren barruko xehetasunak alde batera utzi eta edozein analisi-kateri modu orokorrean eragiten dioten bi arazo identifikatu ditugu, tesi-lan honetan modu sakonean lantzeko. Bi arazoak integrazioaren kontzeptuarekin lotuta daude, analisi-kateak oso heterogeneoak izan baitaitezke, bai anotazio-eskemen mailan eta bai prozesu mailan (Zajac *et al.*, 1997;

Ferrucci eta Lally, 2004). Azter ditzagun bi integrazio maila horiek sakonago, eta ikus ditzagun zein diren tesi-lan honen helburuak integrazio-arazo horien aurrean:

- Tresnen anotazio-eskemen mailako integrazioa, analisi-kateak anotazio-eskema desberdinak erabiltzen dituzten moduluz osatzen baitira. Modulu horiei hizkuntza-prozesatzaile (HP) deritzegu. HP bakoitzak testua maila linguistiko jakin batean prozesatu eta anotazio linguistikoz aberasten du. Anotazioek informazio konplexua izan dezakete barruan, eta, ondorioz, informazio horren adierazpidea arazo-iturri bilakatzen da. Izan ere, analisi-kateko HP bakoitzak, beste HP baten edo batzuen irteerak jaso eta interpretatzeko gai izan behar du. Denek anotazio-eskema berbera erabiliko balute ez legoke arazorik, baina iturri desberdinetatik jasotako HPak elkarlanean jartzean, bateragarritasun-arazoak agertzen dira. Horren aurrean, alde batetik, anotazio-eskema egokien diseinua landu nahi izan dugu, eta bestetik, eskemen arteko elkarreragingarritasunaren arazoa landu eta horren inguruko proposamen batzuk aurkeztu.

- Tresnen prozesu mailako integrazioa. HPen arteko fluxua zurruna denean, HPen arteko konbinazio berriak sortuz analisi-kateak eraikitzea lan nekeza bilakatzen da. Horretarako malgutasuna eskaintzea oso garrantzitsua dela uste dugu, ohikoa baita, proiektuaren edo atazaren arabera, analisi-kateak moldatu eta HP batzuk kendu eta berriak gehitzea. Hori konpontzeko, HPak konbinatuz analisi-kateak modu deklaratiboan definitzeko arkitektura bateratu eta moldagarri bat eraiki dugu. Gainera, gure arkitekturak ingurune banatuetara egokitzeko gaitasuna eskaintzen du, dokumentu bilduma handiak modu eskalagarrian prozesatu ahal izateko. Analisi-kateetako HPen arteko fluxua ere moldagarria da, lortu nahi den optimizazioaren arabera. Dokumentu multzo baten prozesaketa osoa optimizatu nahi bada, dokumentuak paralelizatzen dira, bakoitza makina edo PUZ batean prozesatuz. Aldiz, dokumentu solteen edo multzo txikien prozesaketa bada optimizatu nahi dena, prozesuak ere paralelizatu egiten dira, elkarren artean dependentziarik ez duten prozesuak paraleloan exekutatuz.

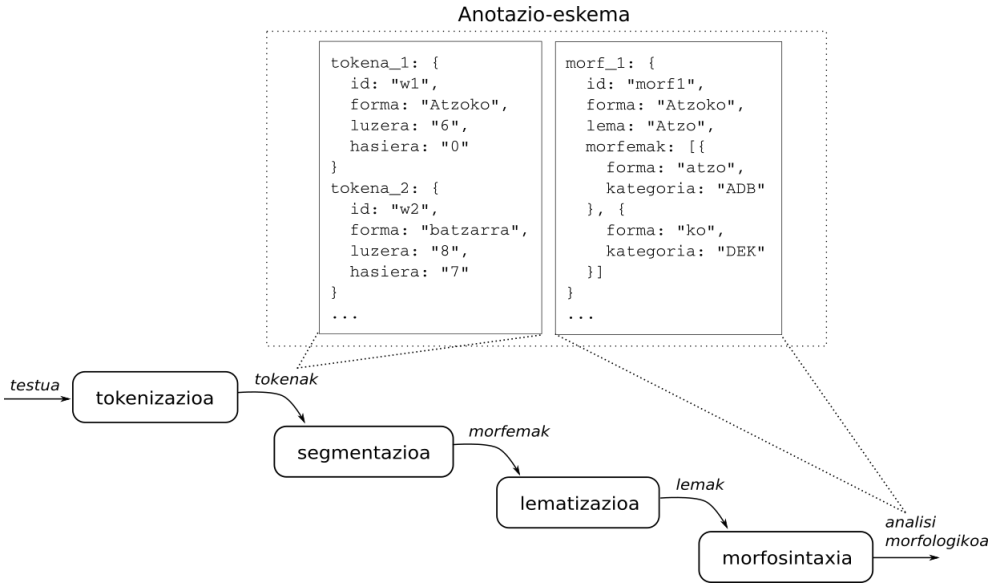
Bi integrazio mailek lotura estua dute elkarren artean. Izan ere, testuen prozesaketa egiteko ingurune ideal batean, analisi-kateak edozein HPz eraikitzeko aukera eskaini behar litzateke. Hala ere, aipatu dugun bezala, anotazio-eskemen mailako integrazioa lortzea ezinbestekoa izango da helburu hori lortzekotan, bestela ezinezkoa izango baita eskema desberdinak erabiltzen dituzten bi HP analisi-kate berean integratzea (Chiarcos, 2012b).

Jarraian, integrazio maila bakoitzak eskaintzen duen problema sakonago aztertuko dugu, irakurlea tesi-lan honetan hobeto koka dadin eta ondo uler dezan, hasieratik, ondorengo kapituluetan aurkeztuko duguna zertan datzan.

1.1. Anotazio-eskemen mailako integrazioa: anotazio-eskemak eta elkarreragingarritasuna

Hizkuntzaren prozesamenduaren hastapenetatik aztergai izan den kontzeptua da hizkuntza-anotazioen adierazpenarena (Ide eta Romary, 2006). Edozein informazio linguistiko prozesatu nahi dela ere, tresnak informazio hori ulertu edo interpretatzen jakin behar du, eta horretarako ezinbestekoa da informazio-unitate bakoitza adierazteko egitura bat finkatzea.

Adibidez, analizatzaile morfologiko batek hitzen egitura morfologikoa erauzi eta anotazio linguistikoen bidez adierazi behar luke (Aduriz *et al.*, 1999). Jakina da euskara morfologia aldetik hizkuntza aberatsa dela, aurrizki eta atzizkiak konbinatuz hitz berriak sortzen baititugu (Alegria *et al.*, 1996). Adibidez, “azpian” hitzaren analisi morfologikoa egiten badugu, lema “azpi” dela eta “an” atzizkia itsatsita duela ikusiko dugu. Haratago joanda, lemaren eta atzizkiaren informazio zabalagoa ere eman dezakegu. Adibidez, “azpi” lema izen arrunta dela, eta “an” atzizkia deklinatzailea dela, zehazki, kasu inesiboa adierazten duen deklinatzailea. Informazio gehiago ere atera daiteke hitz horretatik, haren numeroa singularra dela etab. Hori dena modu txukun eta koherente batean adierazi behar da, HPek anotazioak erosotasunez eta arintasunez interpreta edo sor ditzaten. Mota horretako informazioa egituratzeko XML lengoaia oso ohikoa da, baina XML egiturak ere diseinatu behar dira.



1.2 irudia: Analisi-kateko prozesatzailek itzulitako informazio linguistikoa adierazteko beharrezkoa da anotazio-eskema bat jarraitzea. Irudiko eskema adibide gisa besterik ez dugu erabili, ez da tesian zehar garatu ditugunetako bat.

1.2 irudian ikus daitekeen bezala, analisi-kateko prozesatzaileek itzulitako informazio linguistikoa adierazteko beharrezkoa da anotazio-eskema bat jarraitzea.

Bestalde, hitzaren forma berak interpretazio posible bat baino gehiago izan ditzake maila linguistiko bakoitzean (Aduriz eta Díaz de Ilarraza, 2013). Adibidez, "txoriak" hitzak bi interpretazio morfosintaktiko izan ditzake: nor galderari erantzuten dion absolutibo plurala, edo nork galderari erantzuten dion ergatibo singularra. Lehenbiziko interpretazioa hurrengo esaldian aurki daiteke: "txoriak txioka dabiltza". Bigarren interpretazioa, berriz, esaldi honetan aurki daiteke: "txoriak txio egin du". Testuinguruaren arabera, bata edo bestea izango da zuzena. Hala ere, aplikazio askotan erabilgarria izaten da hitz bakoitzaren interpretazio anbiguo posible guztiak adieraztea, eta zuzena dena modu berezian markatzea. Hortaz, anotazio-eskemak gai izan behar du hitz bakoitzaren interpretazio bat baino gehiago biltzeko ere.

Tesi-lan honen helburuetako bat euskarazko testuen anotazio linguistikoak behar bezala adierazteko gai den Anotazio-Amaraunen Arkitektura, edo AWA (ikus 3.1 atala), izeneko anotazio-eskemaren baitan aurrerapausoak ematea izan da, diseinua berrituz eta formalizazio-ariketa bat eginez. Euskararentzat ez-ezik, edozein hizkuntzarentzat baliagarri izan dadin izan dugu helburu, eta, gainera, informazioaren adierazpidea ahalik eta modurik orokorrenean diseinatzen ahalegindu gara, edozein kasutarako egokia izan dadin.

Bestalde, tesi-lan honek lotura estua izan du NewsReader proiektuarekin[1]. Proiektuaren helburua dozenaka mila albisteren testuak denbora-tarte mugatuan prozesatu eta haietatik gertakarien denbora-lerro bat erauztea izan da. Horretarako, anotazio linguistikoak adierazteko eskema bat eraiki behar izan dugu (ikus 3.2 atala), KAF eskeman (Bosma *et al.*, 2009) oinarrituta. Anotazio-eskema berriari NAF deitu diogu, eta ezaugarri nagusiak bere izaera orokorra eta sinpletasuna dira.

Aurrez esan bezala, anotazio-eskema desberdinak erabiltzen dituzten aplikazioen arteko elkarreragingarritasunaren arazoa ere aztertu nahi izan dugu. Horri konponbidea jartzea konplikatua dela ikusi da urteetan zehar, hasiera-hasieratik landu den arazoa izan arren, gaur egun ere oraindik buruhausteak ematen jarraitzen baitu (Hellmann *et al.*, 2013; Chiarcos, 2012b). Arazo nagusia anotazio-eskema estandar baten gabeziak dakar. Saiakerak egin diren arren (Ide eta Romary, 2004a), praktikan, anotazioen inguruko estandar baten ezarpena ezinezkoa da, hizkuntza ulertzeko teoria desberdinak baitaude, eta bakoitzaren arabera informazio linguistikoaren izaera aldatu egin baitaiteke. Proiektu edo aplikazioaren izaerak ere anotazioen izaera desberdina izatea eska dezake. Gainera, hizkuntza batetik bestera ere hizkuntzaren oinarrizko egitura aldatu egiten da. Ingelesa morfologiari dagokionez sinpleagoa den bezala, euskara edo finlandiera, esaterako, oso konplexuak dira. Hori kontuan hartuta, erraz uler daiteke hizkuntza guztien ezaugarriak bilduko dituen eskema estandar baten diseinua eta ezarpena zein nekeza izan daitekeen.

Aldi berean, ordea, eskema estandar bat ez egoteak begi-bistako konplikazioak dakartza. Tresna bakoitzak informazio linguistikoa bere eskema pro-

[1] `http://www.newsreader-project.eu` (kontsulta: 2017-05-08)

pioa jarraituz egituratzen badu, eta corpus bakoitzak ere bere adierazpidea erabiltzen badu, baliabideak elkarrekin konbinatuta erabiltzea ezinezko bihurtzen da. Arazo hori oso nabaria da hizkuntzaren prozesamenduan, azpiataza asko dituen arloa izanik, ohikoa baita ataza bakoitza tresna edo modulu batek gauzatzea, eta prozesaketa maila linguistiko desberdinetan egin nahi denean, hainbat tresna elkarrekin erabiltzeko beharra askotan agertzen baita, bakoitzak sarrera gisa besteek sortutako irteera hartuz (ikus 1.1 irudia). Horregatik da hain garrantzitsua, nolabait, tresnak anotazio-eskemarekiko independente izatea. Tesi-lan honetan (ikus 4. kapitulua), arazo horri konponbidea ematen, edo, hobeto esanda, konponbidearen lehenbiziko zatia lantzen saiatu gara. Izatez, anotazio-eskemen arteko bihurketak egiteko lehenbiziko urrats batzuk definitu ditugu. Horretarako, anotazio-eredu abstraktu bat diseinatu dugu, edozein eskemak esplizituki edo inplizituki izan ditzakeen elementuak identifikatuz. Horrela, bi eskemaren arteko bihurketa egiteko, lehenbizi eskemak eredu abstraktuaren arabera berrantolatu eta, ondoren, mapaketak modu automatizatuan egitea proposatzen dugu, horretarako OWL (Bechhofer, 2009) teknologiaz baliatuz.

Pauso horiek jarraituz, tresnen eta anotazioen artean geruza berri bat kokatuko litzateke, anotazioen abstrakzio bat eskainiko lukeena. Horrela, tresna jakin bat edozein eskema zehatz erabiltzeko diseinatu dela ere, geruza horri esker, edozein eskemaren arabera adierazitako anotazioak eskema jakin horretara bihurtuko lirateke, tresna horren funtzioa oztopatu gabe.

1.2. Prozesu mailako integrazioa: testuen prozesaketarako arkitektura banatua

Edozein esparrutako profesionalek jakintza zabala eta zehatza behar dute erabaki egokienak hartzeko. Gaur egun eskuragarri dagoen informazioaren tamaina erraldoia dela eta, interesgarria den informazio guztia teknologiaz baliatu gabe aurkitu eta prozesatzea ia ezinezkoa da gizakiarentzat. Hori dela eta, hain baliagarria den informazioa zaharkituta gelditzen ari da etengabe, eta horrek profesionalen erabakietan eragin zuzena dauka. Beraz, edozein berrikuntza edo aldaketaren aurrean behar bezala erantzun ahal iza-

teko informazio eguneratua eskuratzeko lehia izugarria da egungo ia edozein sektoretan. Zeregin horretan lagunduko duen teknologia behar-beharrezkoa da arlo horretan aurrerapauso berriak emateko, eta, arazoa oraindik ebatzita egotetik urrun dagoenez, azken urteetan gorakada nabarmena izaten ari da datu handien tekniken esparrua (Manyika *et al.*, 2011), honek informazio kopuru handiak ahalik eta denbora-tarterik txikienean prozesatzea baitu helburu.

Hizkuntzaren prozesamenduaren arloan ere eragin zuzena dauka aipatutako arazoak. Izan ere, informazio gehiena egituratu gabeko dokumentu gisa aurkitzen denez (Boulton eta Hammersley, 2006; Buneman *et al.*, 1997), webgune edo egunkarietako artikuluetan kasu, dokumentuen arteko korreferentziaren ebazpena edo gertakarien ezagutzea bezalako hizkuntzaren prozesamenduko atazetan milioika testu-dokumentu prozesatu behar dira ahalik eta atzerapen txikienarekin. Newsreader proiektuan, esaterako, gertakarien ezagutzea egitea da helburua, hizkuntza anitzetan, gertakari bakoitza noiz, non eta nori gertatu zaion erauziz eta gertakari berberari dagozkion testuak erlazionatuz, besteak beste. Proiektuan, egunean dozenaka mila dokumentu analisi konplexuen bidez eta ordu gutxiren buruan prozesatzeko ahalmena duen sistema garatu da. Testuen prozesaketaren eskalagarritasunak berebiziko garrantzia dauka proiektuan, beraz.

Prozesaketa handiak egiteko bi eredu nagusi daude: batch eta streaming prozesaketa-ereduak (Shahrivari, 2014). Batch ereduaren arabera, prozesatuko diren datu guztiak bildutakoan hasiko da prozesaketa, eta, beraz, hasieratik daude sarrerako datu guztiak eskuragarri. Aldiz, hainbeste zabaldu diren Twitter[2] eta bestelako sare sozialak, edota RSS[3] bezalako teknologiak direla eta, informazio berria eta oso ugaria ari da etengabe sortzen. IBM erraldoiaren arabera, 2.500 petabyte datu sortzen ditugu interneteko erabiltzaileok egunero[4], eta horren % 75 egiturarik gabekoa da[5]. Etengabeko informazio-jario horrek paradigma-aldaketa bat ekarri du, dagoeneko ez baita beharrezkoa

[2] `https://www.twitter.com` (kontsulta: 2017-05-08)

[3] `https://eu.wikipedia.org/wiki/RSS` (kontsulta: 2017-05-08)

[4] `https://www-01.ibm.com/software/data/bigdata/what-is-big-data.html` (kontsulta: 2017-05-08)

[5] `http://www.bbc.com/news/business-26383058` (kontsulta: 2017-05-08)

prozesaketa hasterako datu guztiak prest izatea. Aldiz, prozesaketa etengabe martxan egon daiteke datu berriak noiz iritsiko zain, iritsi ahala prozesatuz, eta, horrela, sistema beti azken uneko datuekin egunean mantenduz; azken honi streaming eredua deitzen zaio.

Hizkuntzaren prozesamenduan egiten diren prozesaketa astunenak, gehienetan, analisi-kate batekin testu-dokumentu kopuru erraldoiak prozesatu behar direnean gertatzen dira. Corpusak milioika dokumentuz osa daitezke, eta horietako bakoitza analisi-kateko HP bakoitzak prozesatzen du, lortutako irteera kateko hurrengo HPari bidaliz. NewsReader proiektuko analisi-katea, esaterako, 15 prozesatzailek osatzen dute (Beloki *et al.*, 2016). Hori horrela izanik, dokumentu bakoitza prozesatzeko minutuak behar izan daitezke, eta dokumentu asko prozesatu behar diren kasuetan oso litekeena da denbora-arazoak izatea.

Prozesaketa-denborak murrizteko modu bat konputagailu azkarragoak erostea da, eskalagarritasun bertikalaren alde eginez. Baina soluzio hori, garestia izateaz gain, oso mugatua da, konputagailu bakarrarekin lor daitekeen prozesaketa-ahalmena bera ere oso mugatua baita. Soluzio egokiena, egungo egoeran, prozesaketa hainbat konputagailutan banatuta exekutatzea da, ingurune banatuetan, alegia. Modu horretan eskalagarritasun horizontala lor daiteke, bertikala baino askoz ere merkeagoa eta ahaltsuagoa dena. Horrelako ingurune batean eraginkorrena, beharbada, analisi-kateko HP bakoitza MapReduce (Dean eta Ghemawat, 2008) bezalako paradigma ezagun baten arabera berrinplementatzea litzateke, HP bakoitzaren exekuzioa bera barrendik paralelizatuz. Hala ere, prozesatzaile bakoitza berrinplementatzea izugarrizko lana litzateke, eta ez oso hedagarria gainera, kateari prozesatzaile berri bat gehitzea lan nekeza bihurtuko bailitzateke. Irtenbide hori baztertuz, tesi-lan honetan (6. kapituluan), edozein analisi-kate ingurune banatu batean ezarri eta prozesaketa paraleloa ahalbidetuko duen sistema bat aurkeztuko dugu. Ingurune horretan, HPak paraleloan exekutatuko dira, dokumentu multzo handien prozesaketak modu esanguratsuan azkartuz. Sistemaren barne-paralelizazioaren funtzionamendua erabiltzailearentzat gardena izango da, nodoen arteko sinkronizazioak eta datu-trukeak modu automatikoan egingo baitira.

Beraz, dokumentu kopuru handiak modu eskalagarrian prozesatzeko sistema bat aurkeztuko dugu. Sistemak ezartzen zaion azpiegitura fisikoari ahalik eta etekin handiena aterako dio. Hau da, zenbat eta makina gehiago eta ahaltsuagoak elkartu, lortuko diren exekuzio-denborak orduan eta hobeagoak izango dira.

1.3. Tesiaren kokapena eta ekarpen nagusiak

Tesi-lan hau Euskal Herriko Unibertsitateko Ixa ikerketa-taldean[6] garatu dugu. Ixa taldeak 28 urte daramatza euskararen prozesamendu automatikoan oinarrizko baliabideak eta aplikazioak sortzen. Izatez, hizkuntzaren prozesamenduaren adar asko jorratzen dira Ixan, eta tesi-lan honen ekarpena ezinbestekoa da lan guztiak integratu eta taldearen oinarria sendotzeko.

Bestalde, tesi-lan honek lotura estua izan du NewsReader proiektuarekin (FP7-ICT- 2011-8-316404). NewsReaderren garatutako sistemak gaur egun hain ugariak diren testu-formatuko albisteetatik informazioa erauzten du. Besteak beste, zer gertatu den, non, noiz eta nori gertatu zaion erauzten da, informazio hori egituratu gabeko dokumentuetatik jaso eta datu-base egituratuetan gordez, eta, horrela, datu horiek erabiltzea errazago eginez. Tesi-lan honetan, NewsReader sistemaren prozesaketarako arkitektura diseinatu eta inplementatu dugu, sistemari testu-dokumentu kopuru erraldoiak modu eskalagarrian prozesatzeko gaitasuna emanez.

Laburbilduz, hauek izan dira tesi-lan honen ekarpen nagusiak: AWA anotazio-eskemaren hobekuntza eta formalizazio-lana, NAF anotazio-eskemaren garapena, anotazio"-eskemen arteko elkarreragingarritasunean oinarrizko pausoak definitzea eta testu-dokumentu kopuru erraldoien prozesaketa eskalagarria gauzatzeko sistemaren diseinua eta garapena. Ekarpenen inguruko azalpen zabalagoak 7.1 atalean aurki daitezke.

[6] `http://ixa.eus` (kontsulta: 2017-05-08)

1.4. Tesiaren egitura

Tesia bi bloke nagusitan antolatuta dago: anotazio linguistikoen blokea eta testuaren prozesaketa masiborako arkitekturari dagokion blokea.

Anotazio linguistikoen blokea, era berean, hiru kapitulutan banatu dugu. 2. kapituluan anotazio linguistikoen inguruko arloaren egungo egoera landu dugu. Bertan, azken urteetan sortu eta erabili izan diren anotazio-eredu ezagunenak aurkeztu ditugu, arreta nagusia eredu abstraktuetan jarriz. Anotazio linguistikoen inguruko elkarreragingarritasunaren inguruan egindako lanak ere bildu ditugu. 3. kapituluan bi anotazio-eskema aurkeztu ditugu: Anotazio Amaraunen Arkitektura (AWA) eta NLP Annotation Format (NAF). Azkenik, 4. kapituluan, anotazio-eskema desberdinak erabiltzen dituzten tresnen arteko elkarreragingarritasun-arazoaren inguruan jardun dugu, arazoa ebazteko jarraitu beharreko lehenengo urratsak aurkeztuz.

Testuen prozesaketa masiborako arkitekturaren inguruko blokea bi kapitulutan banatu dugu. 5. kapituluan arloaren egungo egoera landu dugu. Horretarako, lehenik, datu kopuru erraldoiak prozesatzeko azken urteetan garatu dituzten sistemarik ezagunenen inguruko azterketa bat egin dugu, eta ondoren, hizkuntzaren prozesamenduaren arloan datu handien inguruan egindako lanak aztertu ditugu. 6. kapituluan, berriz, dokumentu kopuru handiak modu banatuan prozesatzeko sistema eskalagarri bat aurkeztu dugu.

Bukatzeko, 7. kapituluan tesiaren ondorioak bildu ditugu.

1.5. Argitalpenak

Jarraian, tesiarekin lotutako argitalpenak bildu ditugu. Artikuluak, ikus daitekeen bezala, bai kongresuetan eta bai aldizkarietan argitaratu ditugu:

- Agerri R., Agirre E., Aldabe I., Altuna B., Beloki Z., Laparra E., López de Lacalle, M., Rigau G., Soroa A. eta Urizar R. NewsReader project. *Procesamiento del lenguaje natural*, 2014.

- Artola X., Beloki Z. eta Soroa A. A stream computing approach towards scalable NLP. *Language Resources and Evaluation Conference (LREC)*,

2014.

- Artola X., Beloki Z. eta Soroa A. Using Stream Computing Techniques to Process Big Quantities of Textual Information. *International Journal of Computer Science: Theory and Application*, 2014.

- Fokkens A., Soroa A., Beloki Z., Ockeloen N., Rigau G., Robert van Hage W. eta Vossen P. NAF and GAF: Linking linguistic annotations. *Proceedings 10th Joint ISO-ACL SIGSEM Workshop on Interoperable Semantic Annotation*, 2014.

- Agerri R., Artola X., Beloki Z., Rigau G. eta Soroa A. Big data for natural language processing: a streaming approach. *Knowledge-Based Systems*, 2015.

- Beloki Z., Artola X. eta Soroa A. Datu handien tekniken erabilera testu-corpus eskergak prozesatzeko. *Ikergazte*, 2015.

- Kattenberg M., Artola X., Beloki Z., Soroa A., Fokkens A., Huygen P. eta Verstoep K. Two Architectures for Parallel Processing of Huge Amounts of Text. *Language Resources and Evaluation Conference (LREC)*, 2016.

- Carlotto T., Beloki Z., Artola X., eta Soroa A. Interoperability of annotation schemes: Using the Pepper framework to display AWA documents in the ANNIS interface. *Language Resources and Evaluation Conference (LREC)*, 2016.

- Beloki Z., Artola X. eta Soroa A. A scalable architecture for data-intensive natural language processing. *Natural Language Engineering*, 2017.

II Atala

TRESNEN ANOTAZIO-ESKEMEN MAILAKO INTEGRAZIOA

2. KAPITULUA

Informazio linguistikoaren adierazpen-ereduak: arloaren egungo egoera

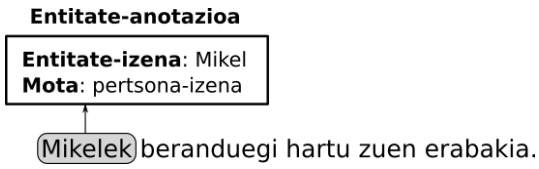
Hizkuntzaren prozesamenduaren hastapenetatik jorratu den gaia izan da anotazio linguistikoen adierazpenarena. Aspalditik anotazio-eskema asko definitu diren arren, azken bi hamarkadetan azaleko eskemetatik haratago joan eta eredu abstraktuen definizioek hartu dute garrantzia. Horien artean, grafoetan oinarritutako ereduak gailendu dira. Kapitulu honetan, anotazio linguistikoen inguruan orain arte egindako lanik garrantzitsuenak azalduko ditugu.

Horrekin hasi aurretik, sarrera bat egingo dugu hurrengo kapituluetan zehar maiz agertuko diren hainbat kontzeptu argituz, irakurleak haien esanahia hasieratik uler dezan.

2.1. Sarrera eta kokapena

Anotazio linguistikoa da, behar bada, tesi-lan honetan gehien aipatuko dugun kontzeptua. Hurrengo kapituluak behar bezala ulertzeko, kapitulu honetan anotazioekin erlazionatutako kontzepturik garrantzitsuenak azalduko ditugu, labur bada ere.

Sarrerako kapituluan esan dugun bezala, hizkuntzaren prozesamenduko tresnen integrazioa da landu dugun gaia, eta tresna horiek informazio linguistikoarekin egiten dute lan. Anotazioak tresna horiek testu zatiei gehitzen



2.1 irudia: Anotazio linguistiko baten adibidea.

dieten informazio linguistikoa dira. Adibidez, 2.1 irudian ikus daiteke entitate-izen bati dagokion anotazio linguistikoa.

Analisi-kate bat ordena jakin batean exekutatzen diren hizkuntza-prozesatzaileen multzoa da, eta hizkuntzaren prozesamenduan ohikoa izaten da testuak analisi-kateen bidez prozesatzea. Horrelakoetan, kateko lehenbiziko tresnak testu gordina jasotzen du sarrerako datu gisa, testutik dagokion informazio linguistikoa erauzi, eta hurrengo tresnari bidaltzen dio. Bigarren tresnak, sarrera-datu gisa lehenbizikoak sortutako informazio linguistikoa jaso eta gauza bera egiten du, bere emaitza hirugarren tresnari bidaliz. Normalean, tresna bakoitzak maila linguistiko bateko anotazioak sortzen ditu, anotazio-geruza berri bat sortuz. Hau da, tokenizatzaileek token-anotazioak sortzen dituzten bezala, entitate-izenen ezagutzaileak testuko entitate-izenak identifikatu eta dagokien informazioa esleitzen die: entitate mota, dagokion wikipediako sarreraren helbidea etab. Anotazio horiekin, token- eta entitate-geruzak izango genituzke. Analisi-kate osoa exekutatu eta gero, hainbat anotazio-geruza izango ditu testuak.

Gertatzen dena da informazio linguistikoa anotatzeko nahi adina modu daudela. Informatikaren munduko beste edozein datu motarekin gertatzen den bezala, informazio linguistikoa adierazteko ere formatu edo anotazio-eskema asko daude. Horietako batzuek informazio berdina adieraz dezaketen arren, modu desberdinean egiten dute: atributuen izen desberdinak erabiltzen dituztelako, atributuen egitura desberdina delako etab. Beste askotan, maila bereko anotazioak adierazi arren, ez dute zehazki informazio berdina adierazteko gaitasunik izaten. Anotazio-eskema batzuekin, adibidez, informazio morfologiko konplexua adieraz daiteke, eta beste batzuekin, informazio morfologikoari dagokionez, kategoria gramatikala besterik ezingo litzateke adierazi.

Anotazioen adierazpide fisikoa anotazio-eskemaren araberakoa da. Ohikoa da anotazioak XML lengoaiaz kodetzea, baina JSON edo datu-baseen bidezkoak ere aurki daitezke. Era berean, anotazioak testuan bertan txerta daitezke, edo testutik bereizita gorde, dagokien testu zatiei erreferentzia eginez. Bigarren horri *standoff* eredua deitzen zaio.

Anotazio-eskema asko daudenez, tresna guztiek ez dituzte eskema guztiak ezagutzen. Hori dela eta, ez da erraza iturri desberdinetatik eskuratutako tresnak kate berean exekutatzea. Horri elkarreragingarritasun-arazoa (*interoperability*) deitzen diogu. Arazo hori konpontzea oso zaila da anotazio-eskemetako elementuak *ad-hoc* sortu badira, hau da, eredu abstraktu bati jarraitu gabe. Izan ere, kontuan izanik anotazio-eskema batek anotazio mota asko adierazteko gaitasuna izan dezakeela, eta anotazio mota bakoitzaren barruan mota askotako datuak egon daitezkeela, bi eskemaren arteko elkarreragingarritasuna lortzeko oso lagungarria da anotazio mota desberdinetako elementuek guraso-klase komunak izatea, eskema osoari koherentzia emanez. Eskemak anotazio-eredu abstraktu baten gainean eraikitzeak asko laguntzen du horretan.

2.2. Anotazio-eredu abstraktuak

Bertsio digitalean zeuden testu-dokumentuen kopurua hazten hasi zen neurrian, informazio horren biltegiratzea modu egituratu batean egiteak gero eta garrantzia handiagoa hartu zuen, dokumentuen gainean bilaketak, kontsultak eta bestelako prozesaketak egiteko ahalmena horren araberakoa izango baitzen. Testu-corpusak kodetzeko helburu horrekin, 1987. urtean, Text Encoding Initiative (TEI)[1] kontsortzioa sortu zuten. Geroztik, azken 30 urte hauetan, formatu digitaleko testuen adierazpenerako gidalerroak garatu ditu TEIk. Hasieratik arrakasta handia izan zuen proiektuak, gaur egun, oraindik, TEI gidalerroek eragin handia baitute informazio linguistikoaren adierazpenaren arloan.

DARPAk (AEBko Defentsarako Ikerkuntza-Proiektu Aurreratuen Agentzia), 1991. urtean, TIPSTER[2] programa jarri zuen martxan, gobernu, industria

[1] `http://www.tei-c.org` (kontsulta: 2017-05-08)

[2] `http://www.itl.nist.gov/iaui/894.02/related_projects/tipster/` (kontsulta:

eta unibertsitateetako ikertzaileen artean hizkuntzaren prozesamenduan aurrerapenak egiteko. Zazpi urte iraun zuen proiektuak, 1998. urtean inbertsio faltagatik amaiera eman baitzioten. Hizkuntzaren prozesamenduko hiru ataza landu ziren nagusiki: dokumentuen bilaketa, informazioaren erauzketa eta testuen laburpen automatikoa. Garatutako modulu eta tresna guztiak integratzeko beharrezkoa izan zen arkitektura bat garatzea, eta horretarako, besteak beste, anotazio-eskema komun bat (Grishman, 1998) diseinatu zuten.

Izaera orokorreko anotazio-eredu abstraktuak definitzeko garaian, grafo-egiturak izan dira gehien zabaldu direnak. 1990eko hamarkadaren bukaeran hasi ziren erabiltzen, eta geroztik agertu diren beste eredu askok oinarri gisa erabili dituzte, kapitulu honetan zehar azalduko dugun bezala.

Anotazio-ereduetan grafoak erabiltzen aitzindariak Bird eta Liberman (1999) izan ziren. Haien ereduak audio-grabaketak anotatzeko balio behar zuen, eta ez, tesi-lan honetan zehar gehiago landuko den kasuan bezala, testu-dokumentuak anotatzeko. Informazio linguistikoaren adierazpenerako honako formalismoa proposatu zuten autoreek: grafo zuzendu bat, non nodoak audio-seinalearen denbora-markak diren, eta nodoen arteko ertzak gako-balio bikotez osatutako edozein ezaugarri multzo. Horrela, ezaugarri multzo horrek ertzaren jatorriko eta helburuko nodoen denbora-marken arteko zatia deskribatzen du. Eredu hori anotazio-grafoen eredu (AG) izendatu zuten. AGak, batez ere, denboraren arabera alineatutako informazioa (audio- eta bideo-transkripzioak) modelatzeko erabiltzen dira.

AGekin egindako lanaren jarraipena ATLAS arkitektura izan zen (Bird *et al.*, 2000). Honen helburu nagusia anotazio-formatu zehatzetatik haratago doazen abstrakzioak proposatzea zen. Bi abstrakzio maila identifikatu eta landu zituzten: logikoa eta fisikoa. Maila abstraktu logikorako AGen eredua erabili zuten, anotazioak sortu, editatu eta ezabatzeko funtzioak eskaintzen zituen API batekin batera, eta maila abstraktu fisikoa XMLn oinarritzen zen adierazpide bat izan zen: AIF. Horiekin batera, ATLAS arkitekturan, hirugarren maila bat ere sartu zuten: aplikazio maila. Maila hori anotazioekin lan egiteko tresnek osatzen dute: hizkuntza-prozesatzaileetatik (HP)

2017-05-08)

hasita, ebaluazioa, kontsulta eta anotazio-eskemen arteko bihurketak egiteko tresnetaraino. Beraz, ATLAS, anotazio-eredu bat baino gehiago, anotazio--framework oso bat da.

AGekin eta ATLASekin oso erlazionatuta dagoen beste framework bat ere proposatu zuten Maeda *et al.*ek (2002): Annotation Graph Toolkit (AGTK). ATLASen aplikazio mailaren parekoa litzateke framework hori, izan ere, AGetan oinarritutako anotazioekin lan egiteko tresna multzo batek osatzen baitu. Besteak beste, eskaintzen ditu anotazio-grafoetako datuak erabiltzeko liburutegi bat eta beste hainbat anotazio-eskemaren arabera kodetutako anotazioak inportatzeko funtzionalitatea. Framework honekin lotuta, corpusak elkarlanean anotatzeko sistema zentralizatu baten diseinua ere aurkeztu zuten Ma *et al.*ek (2002).

Urte berean GATE aurkeztu zuten Cunningham *et al.*ek (2002). GATEk, hizkuntza-prozesatzaileak garatu eta elkarren artean integratzeko frameworka izanik, eredu abstraktu komun bat erabiltzen du. Eredu hori TIPSTERen aldaera bat da, ATLAS ingurunearekin bateragarri egina. GATEk oihartzun nabaria izan zuen arloan geroztik, hainbat HP tresna kate berean aplikatu behar ziren proiektuetan, batez ere.

UIMAk (Unstructured Information Management Architecture) (Ferrucci eta Lally, 2004) ere, GATEren antzeko funtzionamendua izanik, HP tresnen artean informazio linguistikoa trukatzeko anotazio-eredu komun baten alde egin zuen. CAS eredua definitu zuten UIMAren autoreek. Ereduak XMLn oinarritutako adierazpide komun bat eskaintzen du, norberak bere egitura propioak defini ditzan eredu horri jarraituz. Semantikoki ez da oso aberatsa, ereduak berak ez baitu azaleko egitura komun bat besterik eskaintzen. Horrela, HP prozesatzaile berri bat garatzean, garatzaileak sarrera eta irteerako CAS egiturak deklaratuko ditu, tresnen arteko elkarreragingarritasuna erraztuz.

Teich *et al.*ek (2001) aurkeztutako ereduak bai AGetatik eta bai ordenatutako grafo zuzendu aziklikoetatik (ODAG) edaten zuen. Geruza anitzeko corpusen adierazpide egoki bat lortu nahi izan zuten, testu beraren gaineko iturri desberdinetako anotazioak integratzeko. AGetatik, geruzak bereizita gordetzearen ideia jaso zuten bereziki, eta ODAGetatik, egitura hierarki-

koak modu naturalean definitzeko aukera. Standoff eredu bat lortu zuten, non geruza bakoitzeko anotazioek jatorrizko testuaren karaktereei dagozkien *offset*-ak[3] adierazteko atributuak dituzten. Adierazpidearen oinarrian XML dago.

AGetan oinarrituta, Ide eta Romary-k (2004a) Language Annotation Framework (LAF) izeneko beste anotazio-eredu abstraktu bat aurkeztu zuten. Izatez, ISO erakundearen baitan garatutako estandar bat da, komunitateak anotazio-eskemekiko zeukan desadostasun orokorraz jabetuta, batzorde bat antolatu baitzuten estandar bati bidea zabaltzeko asmoz (Ide eta Romary, 2002). Batzordea arloan adituak ziren hainbat pertsonak osatu zuten. XMLn oinarritutako Corpus Encoding Standard (XCES) gidalerroak (Ide *et al.*, 2000) dira LAFen jatorria, hortik abiatuta egin baitzituzten ereduaren lehenbiziko pausoak (Ide eta Romary, 2001, 2002). Ondoren, hortik abiatu zen aipatutako batzordea LAF garatzeko.

LAFen helburua ez da inoiz izan gainontzeko eskemak ordezkatzea. Eredu abstraktua izanik, eskemen arteko bihurketak egiteko tarteko eredu izatea da bere helburu nagusia, horrela, hizkuntza-prozesatzaileen arteko elkarreragingarritasuna bideratzeko asmoz. Eredua oinarrizko bi kontzepturen inguruan eraiki zen: anotazio mota bakoitzak izango ditu egitura abstraktuak, batetik, eta egitura horiek osatzeko beharrezko diren datu-kategoriak, bestetik. Behin egitura abstraktuak definituta, elkarreragingarritasuna lortzeko aski litzateke datu-kategorien erregistro (DCR, *Data Category Registry*) orokor bat ezartzea (Ide eta Romary, 2004b), tresnek iturri desberdinetatik jasotako atributu (datu-kategoria) desberdinen arteko erlazioa uler dezaten. LAFen datu-eredua Bird eta Liberman-en (1999) AGetan oinarritzen da, eta beraz, LAF ere grafoetan oinarritzen den eredua da.

LAF diseinatu zuen batzorde berak, aurrerago, LAFi oihartzun handiagoa emango zion XML serializazio bat aurkeztu zuen: GrAF (Ide eta Suderman, 2007). Adierazpide fisiko horrek edozein formatutan dauden anotazioak LAF ereduaren arabera adierazteko gaitasuna eskaintzen du. Horretarako, formatuaren erabiltzaileak anotazioak GrAFen egituraren arabera kodetu behar ditu, horrela, anotazioak, izatez edozein formatutan egonik ere, egitura ko-

[3]Karaktere jakin batek testuan duen posizioa.

mun baten arabera kodetuta egotea lortuz. Horrez gain, grafoen algoritmoak AGen gainean aplikatzeari buruzko hausnarketa egiten dute, eta erabilera posible batzuk erakutsi: anotazio-eskemen arteko bihurketak, testu zati berberaren gainean definitutako anotazioen elkarketa, anotazio-grafoen bisualizazioa eta estatistiken kalkulua.

LAFekin eta GrAFekin egindako lanek eragina izan dute komunitatean. Horren adibide, medikuntza-arloko dokumentuen eta horien anotazio linguistikoak adierazteko eraikitako CDA+GrAF eredua (Meystre *et al.*, 2012). HL7, Clinical Document Architecture (CDA) eta GrAF elkartuz osasun-arloko dokumentuak adierazteko gai den eredua osatu zuten. Dokumentuaren inguruko metainformazioa (gaixoaren izena, data eta txostenaren autorea, besteak beste) CDA erabiliz kodetzen dute, eta testuaren gaineko anotazio linguistikoak kodetzeko, berriz, GrAF erabiltzen dute.

Anotazio-grafoetan oinarritutako beste eredu bat NITE datu-eredua da (Carletta *et al.*, 2003; Evert *et al.*, 2003). AGetan ez bezala, anotazioak grafoko nodoen bitartez adierazten dira, eta ez ertzen bidez. Gainera, nodo bakoitzak hainbat eremu izan ditzake: anotazio motaren etiketa, gako-balio motako atributuak, offsetak, anotazioen arteko erlazio hierarkikoak eta erlazionatutako anotazioak. Egitura konplexu honi esker, jatorrizko AGek zuten arazo nagusia konpondu zuten, erlazio hierarkikoak definitzeko gaitasunik eza (Teich *et al.*, 2001; Brugman eta Wittenburg, 2001; Ide eta Romary, 2002), alegia. Erlazio hierarkikoei esker, atributuen balioak, bakunak izateaz gain, beste gako-balio batzuez osatutako egiturak ere izan daitezke. Ertz motak bereiztearena da berrikuntza nagusia AGekin alderatuz, erlazio hierarkikoak eta lotura-erlazioak bereizten baitira.

NITEk, eta, oro har, ODAGek, anotazio-grafoen ahuleziak konpontzen dituztela ikusita, horietan oinarritutako ereduak zabalduz joan ziren. Esate baterako, Dipper *et al.*ek (2004) proposatu zuten anotazio-ereduak NITErena hedatzen du bi ezaugarri gehiago erantsiz: corpus osoak (alegia, dokumentu bat baino gehiago) grafo bakarrean biltzeko ahalmena eta anotazio konplexuak definitzeko aukera, horretarako erlazio hierarkikoei etekina ateraz. Proiektuaren helburua alemanezko corpus diakroniko baten adierazpena bideratzea zen. Inplementazio fisiko bezala XMLn oinarritutako formatu bat

proposatzen duten arren, datu-base erlazionalak erabiltzen dituzte corpusa biltegiratzeko, gero kontsultak egitea eraginkorragoa eta erosoagoa izan dadin.

NITE eta LAF ereduekin konpara daitekeen beste eredu bat PAULA (Chiarcos *et al.*, 2008) da, aurreko urteetan egindako beste lan batzuen ondorioz jaiotako eredua (Dipper eta Götze, 2005; Dipper, 2005). LAFekin alderatuz, ertz motak bereizten ditu, semantika desberdinak esleituz bakoitzari, NITE ereduan gertatzen denaren antzera (erlazio hierarkikoak etab.). NITErekin alderatuz, aldiz, ez da erlazio hierarkikoetan hainbeste zentratzen, lotura-erlazioei ere garrantzia handia ematen baitzaie PAULAn.

PAULA web semantikoaren mundura ere eraman zuten. Horretarako, bihurketa bat egin zuten eredua RDF[4] eta OWL (Bechhofer, 2009) lengoaietara pasatzeko, POWLA aurkeztuz (Chiarcos, 2012a). Horri esker PAULAz adierazitako anotazioak baliabide linguistikoekin esteka daitezke, aurrerago aipatuko dugun elkarreragingarritasunaren arazoa (ikus 2.4 atala) konpontzeko urrats gisa.

Salt (Zipser eta Romary, 2010) anotazio linguistikoak adierazteko grafoetan oinarritutako datu-eredu abstraktu bat da. Saltek eta PAULAk ezaugarri komun asko dituzte, datu-eredua oso antzekoa baita. Eredu abstraktua izanik, semantikoki ez da aberatsa, edozein anotazio-eskemarekin bat etortzeko gaitasuna izatea baitu helburu. Saltekin batera Pepper garatu zuten, oinarritzat Salt eredua hartuta, edozein bi anotazio-eskemaren artean bihurketa egiten laguntzen duen tresna. Aipatutako tresnak ANNISekin (Zeldes *et al.*, 2009) batera erabiliz, konbinazio ahaltsua lortzen da. ANNIS maila anitzetan anotatuta dauden corpusak kontsultatu eta bistaratzeko tresna da. Horrela, Salt, Pepper eta ANNIS tresnak erabiliz, edozein eskemaren arabera anotatutako corpusak bisualizatzeko eta kontsultatzeko aukera ematen dute. Horren inguruan guk geuk ere egin dugu ekarpenik: tesi-lan honetan aurkeztuko dugun Anotazio Amaraunen Arkitektura anotazio-eskemaren (3.1 atala) dokumentuak ANNIS interfazean kargatu, erakutsi eta erabiltzeko integrazioa gauzatu dugu (Carlotto *et al.*, 2016).

[4] `https://www.w3.org/RDF` (kontsulta: 2017-05-09)

FoLiA (Format for Linguistic Annotation) (Van Gompel eta Reynaert, 2013) anotazio-eredu komun bat ezartzeko azken saiakeretako bat izan zen. Anotazio-eredu abstraktu baten eta anotazio-eskema baten artean dago, hedagarria izan arren eta anotazioen balio posibleak definitzea erabiltzailearen esku gelditzen den arren, anotazio moten multzo mugatu bat eskaintzen baitu. Beraz, LAF bezalako eredu abstraktu batekin alderatuta, erabilera praktiko batera gehiago zuzendutako eredua da. Gainontzekoetatik bereizten duen ezaugarri nagusietako bat anotazioak dokumentuan bertan txertatzearena da, standoff eredua jarraitu ordez. Gainera, anotazio guztiak dokumentu bakarrean biltzen dira. Anotatutako dokumentuak prozesatzeak baliabide gutxiago behar izateko hartu zuten diseinu-erabaki hori. Anotazio-eskema ere baden aldetik, XMLn oinarritutako sintaxia erabiltzen du anotazioak fisikoki adierazteko.

Grafoak alde batera utzita, objektuei zuzendutako ereduekin ere egin dira saiakerak. Brugman eta Wittenburg-ek (2001) Abstract Corpus Model (ACM) aurkeztu zuten. Javaz baliatu ziren eredua inplementatzeko. Klase abstraktuz osatutako eredu batetik abiatuta, edozein anotazio-eskema inplementa daiteke horietatik heredatzen duten klaseak inplementatuz. ATLA-Sekin bateragarri egiteko saiakera ere egin zuten, honek komunitatean lortu zuen babesa zela eta.

Eberle *et al.*ek (2012), bestalde, hiru dimentsiotako analisi mailak mantendu nahi izan zituzten jatorrizko testuen gainean. Dimentsio bertikala maila linguistiko desberdinetako anotazio-geruzek osatzen dute. Dimentsio horizontalean, iturri desberdinetatik sortutako maila linguistiko berberari dagozkion anotazio multzoak daude. Azkenik, hirugarren dimentsioa hizkuntza-prozesatzaileen bertsio desberdinek sortutako anotazio multzoek osatzen dute. Horiek denak mantentzeko, datu-base erlazionaletan oinarritzen den egitura bat inplementatu zuten. Ez zuten, izatez, anotazioak adierazteko eredu abstrakturik aurkeztu, baina HPen bertsioak kudeatuz, gainontzeko ereduek askotan kontuan hartzen ez duten arazoa jarri zuten mahaiaren gainean.

2.3. Datu estekatuak eta informazio linguistikoa

Azken urteetan web semantikoarekin (Berners-Lee *et al.*, 2001) lotutako lanak asko ari dira zabaltzen. Izaera askotako baliabide linguistikoak sarean eskuragarri egonik (Wikipedia[5], WordNet (Miller, 1995)...), gizakiak ez ezik, konputagailuek ere datuok erabiltzeko gaitasuna izatea lortu nahi da. Horretarako gakoa datu estekatuetan dago. Datu estekatuek hainbat iturritako datuak elkarren artean formalki erlazionatzen dituzte, URIen[6] bitartez. Horrela, adibidez, Wikipediako eta WordNeteko sarrera bana kontzeptu berberari dagozkiola defini daiteke, baliokidetasun hori maila konputazionalean erabiltzea ahalbidetuz.

Datu estekatuen teknikak informazio linguistikoaren testuingurura eramateko helburuarekin ere hainbat saiakera egin dituzte. Anotazio-eskema bakoitzak ezaugarri linguistikoak eta beren balio posibleak adierazteko bere terminologia propioa erabiltzen duenez, anotazioak sortu edo erabiliko dituzten tresnek anotazio-eskema jakin batera egokituta egon behar dute. Ondorioz, lan nekeza bihurtzen da eskema desberdinetara egokituta dauden tresnak integratu eta batera erabiltzea. Hori konpontzeko soluzio bezala, datu-kategorien erregistro bat eraikitzea eta eskema bakoitzaren kontzeptuak erregistro estandar horretara mapatzea proposatu zuten (Ide eta Romary, 2002).

Bide horretan, GOLD ontologia sortu zuten (Farrar eta Langendoen, 2003). Ontologia hizkuntzaren fenomeno desberdinak deskribatzeko ezaugarri eta atributuen biltegi bat da. Horri esker, anotazio-eskema desberdinetako atributu-izen eta etiketa-balioak ontologiako balioetara mapatuz, edozein tresna anotazio-eskema horietako edozeinen edukia ulertzeko gai izatea lortu nahi zuten (elkarreragingarritasun kontzeptuala).

Bide beretik, ISOcat datu-kategorien erregistro estandarra (Kemps-Snijders *et al.*, 2008) sortu zuten. ISOcat tresna semantikoen sare zabalago baten parte da (Schuurman eta Windhouwer, 2011). Noski, erregistroaren estandarizazioa erabilgarria izan dadin, baliabide linguistikoek anotazioen eta da-

[5] `https://www.wikipedia.org` (kontsulta: 2017-05-08)

[6] URI: baliabide bat sare osoan modu unibokoan identifikatzen duen gakoa.

gozkien erregistro-balioen arteko loturak egin behar dituzte (Windhouwer *et al.*, 2010). Loturak egiteko, hasiera batean XML erabili arren, RDF eta OWL bezalako lengoaiak gero eta gehiago zabaltzen joan dira (Windhouwer eta Wright, 2012), 1:1 motakoak baino mapaketa konplexuagoak modu naturalean definitzeko aukera ematen baitute. ISOcat-eko kategorien zerrenda luzea da, eta askotan kategoria batzuk elkarrengandik oso gertu daude semantikoki. Horrelakoetan, argigarria litzateke kategorien arteko erlazioak ere modu formal batean definituta baleude, baina hori ISOcat-en espezifikazioetatik kanpo gelditzen da. Testuinguru horretan kokatzen da RELcat (Windhouwer, 2012), ISOcat-eko kategorien arteko erlazioak formalki definitzen baititu.

Beste ekarpen garrantzitsu bat OLiA, anotazio linguistikoen ontologiena izan zen (Chiarcos, 2012c). GOLD eta ISOcat erabilgarri egonik, hainbat arazori aurre egin behar zitzaien anotazioak erregistro orokor batekin lotzeko garaian: 1) Erregistro bat baino gehiago zeudenez erabilgarri, loturak denekin egin beharko lirateke; 2) ez zegoen loturak egiteko formalismoen inguruko adostasunik (XML, RDF, OWL...), eta 3) erreferentziazko erregistroan aldaketaren bat egiten bazuten, haren menpeko lotura guztiak birdefinitu behar ziren. OLiAren soluzioa tarteko ontologia bezala ezartzea izan zen: anotazioen kategoriak OLiAko kontzeptuekin lotu behar ditu erabiltzaileak, eta OLiAk berak mantenduko ditu OLiAren eta gainontzeko kategoria-erregistroetako kontzeptuen arteko loturak. OLiA ontologiak OWL/DL[7] lengoaiaz idatzita daude, eta ondorioz, 1:1 baino mapaketa konplexuagoak ere egin daitezke.

POWLA (PAULA eredua OWLez adierazita) inplementatu izana PAULA anotazio-eredu abstraktuaren arabera adierazitako anotazioen eta OLiAren arteko loturak egiteko gaitasuna lortzearren izan zen. Horrela, PAULA ereduari esker elkarreragingarritasun estrukturala lor zitekeen, eta OLiAko mapaketei esker elkarreragingarritasun kontzeptuala.

Gai honetan egindako lanak uztartu eta bide komun batean aurrera egiteko, Open Linguistics Working Group (OWLG) lantaldea osatu zuten (Chiarcos *et al.*, 2011). Lantaldearen helburu nagusia Linguistic Linked Open Data sa-

[7] `https://www.w3.org/2001/sw/wiki/OWL` (kontsulta: 2017-05-08)

rea (LLOD) osatzea da, erabilgarri dauden baliabide linguistikoetako datuak elkarren artean lotuz.

Datu estekatuen testuinguruan egindako beste ekarpen bat Lemon eredua izan zen (McCrae *et al.*, 2011). Lemon baliabide lexikalen errepresentazioa deskribatzeko garatu zuten. Horretarako, dagoeneko aipatu ditugun GOLD eta ISOcat bezalako ontologiak eta WordNet bezalako baliabide lexikoak erlazionatzea du helburu, datu estekatuen sare bat osatuz. Baliabide lexikoak web semantikoan partekatzeko, baliabideek ontologietan esleituta dauzkaten adierazpen semantikoekin lotzeko sistema sendoa eskaintzen du. Ereduaz baliatuz hainbat lan egin dira: WordNet eta Wiktionary Lemon-era bihurtu eta elkarren arteko loturak definitu dituzte (McCrae *et al.*, 2012), eta DBpediaren ingelesezko lexiko bat Lemon erabiliz eraiki dute (Unger *et al.*, 2013), besteak beste.

Lemon-en antzeko helburua du NLP Interchange Format-ek (NIF) ere, hizkuntza-prozesatzaileen irteerako anotazioak bestelako baliabide linguistikoekin lotzeko formalismoa eskaintzen baitu (Hellmann *et al.*, 2012).

Azken urteetan, Universal Dependencies proiektuaren baitan, hizkuntzen artean kontsistentea den sintaxi-zuhaitzen anotazioa lantzen ari dira (Nivre *et al.*, 2016). Helburua da hizkuntza guztietan informazio sintaktikoa modu berean anotatzea, kategoria komunak erabiliz.

2.4. Elkarreragingarritasuna

Aurreko ataletan aipatutako lan askok anotazioen arteko elkarreragingarritasuna lortzea dute helburu nagusi. Anotazio linguistikoekin lan egiten duten tresnek, aplikazio bakarrean integratu eta elkarren artean komunikatu ahal izateko, adierazpen-eredu komun bat erabili behar dute. Adierazpen-eskema fisikoa desberdina izanagatik ere, badaude integrazio maila hori lortzeko teknikak.

Baliabide linguistikoen kopurua asko hazi da XXI. mendeko lehenbiziko hamarkadan zehar, eta bigarren hamarkada honetan baliabideen arteko elkarreragingarritasunaren arazoa konpontzeko interesa zabaldu egin da. Horren adibide, AEBn eta Europan martxan jarri diren bi proiektu handi: Sustaina-

ble Interoperability for Language Technology (SILT) eta Fostering Language Resources Network (FLaReNet), hurrenez hurren (Ide *et al.*, 2009).

Ide eta Pustejovsky (2010) elkarreragingarritasunaren oinarriak finkatzen saiatu ziren. Batez ere, arazoa argitasunez identifikatu nahi izan zuten, eta arazoari heltzeko lau bloke nagusi bereizi zituzten: baliabide linguistikoak deskribatzeko metadatuak, datu-kategoriak eta haien semantika, anotazioen publikaziorako betebeharrak eta hizkuntzaren prozesamenduko softwarea banatzeko betebeharrak.

Chiarcosen arabera (2012b), bi elkarreragingarritasun maila bete behar dira: estrukturala eta kontzeptuala. Elkarreragingarritasun estrukturala beharrezkoa da jatorri desberdinetako anotazioek egitura komun bat izan dezaten, eta hori lortzeko, anotazio-eredu abstraktu komun baten arabera modelatzen dira anotazio-eskema desberdinen arabera adierazitako anotazioak. Horri esker, tresnak gai dira anotazioen barne-egitura interpretatzeko. Hori ez da nahikoa, ordea, egitura berbera izan arren, teoria linguistiko desberdinen arabera modelatutako anotazioek ez baitituzte atributu berberak erabiliko fenomeno linguistikoak deskribatzeko. Hori dela eta, atributu desberdinen arteko erlazioak nonbait definituta egotea ere beharrezkoa da, atributu horien semantika konputazionalki interpretatu ahal izateko (elkarreragingarritasun kontzeptuala).

Bird *et al.*ek (2000) beste ikuspuntu batetik landu zuten elkarreragingarritasuna ATLAS diseinatu zutenean. Beren ustez, hiru mailatako arkitekturetan dago tresnen arteko integrazioa lortzeko gakoa. Ordura arteko tresna gehienek bi mailatako arkitektura inplementatzen zuten, aplikazio mailara eta adierazpen-maila fisikora mugatua. Autoreen ustez, bi horien arteko hirugarren maila bat gehitu behar da, bai aplikazio mailarekiko eta bai adierazpide fisikoarekiko independentea, eta, nolabait, anotazioen adierazpena abstraituko duena. Hirugarren maila horrek Chiarcos-en (2012b) elkarreragingarritasun estrukturala ebaztea luke helburu. Maila horretan koka daitezke lehenago azaldutako anotazio-eredu abstraktuen inguruko lanak.

Elkarreragingarritasun kontzeptuala konpontzeko datu estekatuen bidea landu izan da gehien. GOLD (Farrar eta Langendoen, 2003) eta ISOcat (Kemps-Snijders *et al.*, 2008) ontologiek anotazio linguistikoen erregistro orokor izatea

dute helburu, eta OLiA ontologiak horien gaineko abstrakzio bat egiten du, erabiltzaileek interfaze bakarrarekin lan egin ahal izan dezaten. DCR erregistroa ere anotazioen erreferentziazko erregistro izateko helburuarekin jarri zuten martxan (Ide eta Romary, 2004b). Lemon eta NIF bezalako ereduek ere beren ekarpena egiten dute, datu estekatuen sarea handitzen laguntzen baitute, tresna linguistikoen arteko elkarreragingarritasuna lortzeko bidean. Era berean, OWLG lantaldeak elkarreragingarritasunaren arazoari datu estekatuen bidetik aurre egiteko ardura hartu zuen (Chiarcos *et al.*, 2011).

Anotazio-eskemen arteko bihurketak egitea elkarreragingarritasuna lortzeko pauso garrantzitsuenetako bat da. Hori da Ide eta Sudermanek (2009) egin zutena. GrAFen autoreak izanik, GATE eta UIMAren eskemetatik GrAFerako bihurketa (eta alderantziz) nola egin proposatu zuten. Buntek (2010) metodologia bat deskribatu zuen anotazio-eskema baten sintaxi abstraktua eta konkretua bereizita definitzeko. Hori oinarri hartuz, Ide eta Bunt-ek (2010) anotazioak edozein eskematatik GrAFera bihurtzeko metodologia aurkeztu zuten. Bide beretik, GrAF ereduaren arabera kodetutako MASC[8] corpusa ANNIS kontsulta-tresnan nola inportatu zuten azaltzen dute (Neumann *et al.*, 2013). Horretarako, anotazioak GrAFetik relANNIS eskemara bihurtu behar izan zituzten.

Pareja-Lora eta de Cea-k (2010), ontologietan oinarrituz, tresna linguistikoen arteko elkarreragingarritasuna gauzatu zuten. Hainbat lema-etiketatzaile aplikatu zituzten testu beraren gainean, eta bakoitzaren emaitzak konparatuz, emaitza orokorra hobetzea lortu zuten.

[8] `http://www.anc.org/data/masc` (kontsulta: 2017-05-08)

3. KAPITULUA

Anotazio-eskemak: Anotazio Amaraunen Arkitektura eta NAF

Kapitulu honetan bi anotazio-eskema aurkeztuko ditugu. Anotazio Amaraunen Arkitektura eta NLP Annotation Format. Lehena tesi-lan honekin hasi aurretik sortua izan zen, baina orain formalizazio-lan bat egin dugu berarekin, elkarreragingarritasunari arreta berezia jarriz. Bestea, bere aldetik, eskema praktikoa eta sinplea izateko helburuekin sortu dugu.

3.1. Anotazio Amaraunen Arkitektura

Anotazio Amaraunen Arkitektura (ingelesez *Annotation Web Architecture* edo AWA) (Artola *et al.*, 2009), anotazio linguistikoak adierazteko Ixa taldean garatutako eredua da.

Sarreran esan dugun bezala, ezinbestekoa da analisi-kateak osatzen dituzten prozesatzaileek beren arteko sarrera eta irteerako formatuak ulertzea. Ixa taldean garatutako euskararako prozesatzaile guztien arteko komunikazioa behar bezala egiteko helburuarekin sortu zen AWA, tresna guztien irteerako anotazioen adierazpidea deskribatuko duen anotazio-eskema.

Anotazio-eskema berri bat sortzearen arrazoi nagusia euskararen morfologiaren konplexutasuna da. Aurki daitezkeen anotazio-eskema gehienak ingelesezko anotazioak adierazteko helburuarekin diseinatu zituzten, eta ingelesaren morfologia sinpleagoa dela kontuan izanik, euskarazko egitura morfo-

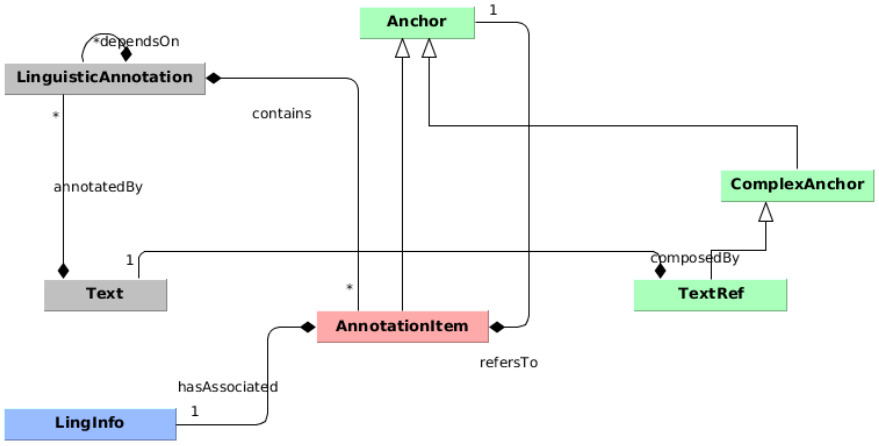
logikoak xehetasunez adieraztekotan, anotazio-eskema berri baten beharra zegoela ikusi zen.

Interpretazio anitzeko kasuak, hau da, anbiguotasuna dagoen kasuak ere ondo adierazteko gaitasuna ezinbestekoa zen. Izatez, hizkuntza morfologikoki hain aberatsa izateak hitzen interpretazio anbiguoen kopurua ere asko hazten du. Prozesatu beharreko testuetan horrelako interpretazio anbiguoak aurkitzean, anotazio-eskemak interpretazio guztiak adierazteko gaitasuna eskaini behar du, nahiz eta testuinguru jakin batean interpretazio bakarra izango den zuzena.

Datu-ereduak maila askotako anotazioak adierazteko aukera ere izan behar zuenez, hasieratik izan genuen kontuan anotazio-eskemak har zezakeen konplexutasun handia. Horregatik, jarraian azalduko dugun bezala, lehenbizi oinarrizko datu-eredu bat diseinatu, eta horren gainean inplementatu dugu anotazio-eskema zehatza.

3.1.1. AWAren datu-eredua

Anotazio-eskema batek hainbat motatako anotazioak adierazteko aukera eskaini behar duenean, garrantzitsua da eskema hori datu-eredu komun baten gainean inplementatzea (Bird eta Liberman, 1999). Besteak beste, anotazio-eskema desberdinak erabiltzen dituzten tresnak aplikazio berean integratzeko orduan azaleratzen dira anotazio-eskema eredu abstraktu baten gainean eraiki izanaren onurak (Ide eta Romary, 2004a; Chiarcos *et al.*, 2008). Adibidez, demagun testu bat hainbat hizkuntza-prozesatzailerekin prozesatu nahi dugula, eta prozesatzaileak iturri desberdinetatik lortu ditugunez, anotazio-eskema desberdinak erabiltzen dituztela. Prozesatzaile bakoitzak, normalean, aurrez aplikatutako beste prozesatzaileen irteera erabiliko du bere irteera sortzeko. Horretarako, beharrezkoa da prozesatzaileak besteen irteerak ezagutzea, edo behintzat, haiekin bateragarri izatea. Hori lortzea askoz ere errazagoa da eskema hori datu-eredu jakin baten gainean inplementatuta dagoenean, kasu horretan egitura komunak errepikatuko baitira maila linguistiko guztiei dagozkien anotazioetan zehar. Eskema eredu jakin baten gainean inplementatu ez bada, eta hainbat mailatako anotazioetan errepikatzen diren fenomenoak adierazteko egitura arbitrarioak erabili badira, bateragarritasun



3.1 irudia: AWAren datu-ereduaren diagrama.

hori inplementatzea lan nekeza bihurtzen da. Horren adibide da anotazioek beste anotazioei erreferentziak egiteko modua berbera izatea erreferentziak egin behar diren kasu guztietan, esate baterako.

3.1 irudian ikus daiteke AWAren anotazio-eskemaren atzean dagoen datu-ereduaren diagrama. Datu-eredua gai da maila linguistiko anitzeko (morfologikoa, semantikoa...) anotazioak adierazteko. Maila bakoitzeko anotazioek egitura jakin bat izango dutenez, horren araberako multzokatze bat egiten da. Maila linguistiko bereko anotazioz osatutako multzo horiei anotazio-geruza deritzegu (irudiko `LinguisticAnnotation` klasea).

Anotazioetan zentratuz, eta geruza eta corpusen antolaketarako elementuak (`LinguisticAnnotation`, `Text`) alde batera utziz, hiru elementuk osatzen dute eredua: aingurek (`Anchor`), estekek (`AnnotationItem`) eta informazio linguistikoa biltzen duten analisi linguistikoek[1] (`LingInfo`):

- **Aingurak**: Aingurek anotazioak zein unitateri buruzko informazio linguistikoa ematen duen adierazten dute. Askotan, unitatea jatorrizko testuko hitz bat, edo, orokorrago esanda, testu zati bat, izaten da. Beste askotan, berriz, beste geruza bateko anotazioak iza-

[1]Hemendik aurrera, sinplifikatzearren, "analisiak" idatziko dugu "analisi linguistikoak"-en baliokide gisa.

ten dira. 3.1 irudian ikus daitekeen bezala, aingurak bi multzo nagusitan sailka daitezke. Alde batetik, edozein anotazio izan daiteke aingura (`AnnotationItem` klasea `Anchor` klasearen azpiklasea baita), baina beste aldetik, kasu askotan aingurak konplexuagoak izaten dira (`ComplexAnchor`), anotazio bakarra izan ordez, anotazioen konbinazio bat, adibidez. Aingura jatorrizko testu zati bat denean ere aingura berezia izango da, karaktere zerrenda baten azpimultzo bat erreferentziatu behar baitu (`TextRef`). Ainguren inguruko informazio zabalagoa 3.1.2.1 atalean aurki daiteke.

- **Informazio linguistikoa (analisiak)**: Anotazioak berak testu zati bati buruz edo beste geruzetako anotazioei buruz ematen duen informazioa da, hots, aingura bati dagokion informazio linguistikoa. Anotazio baten informazio linguistikoari analisi linguistiko ere deitzen diogu. Adibidez, hitz baten kategoria edota lema zein diren adieraz daiteke bertan. AWAn ere TEI P4-k (Sperberg-McQueen eta Burnard, 2001) definitutako ezaugarri-egiturak erabiltzen dira informazio linguistikoa kodetzeko. Horri esker, gako-balio motako ezaugarri arruntak ez ezik, gako-balioen arteko egitura hierarkikoak ere eraiki daitezke, gako baten balioa bakuna izan ordez, beste gako-balio bat (edo zerrenda edo multzo bat) den kasuetan (ikus 3.1.1.1 atala).

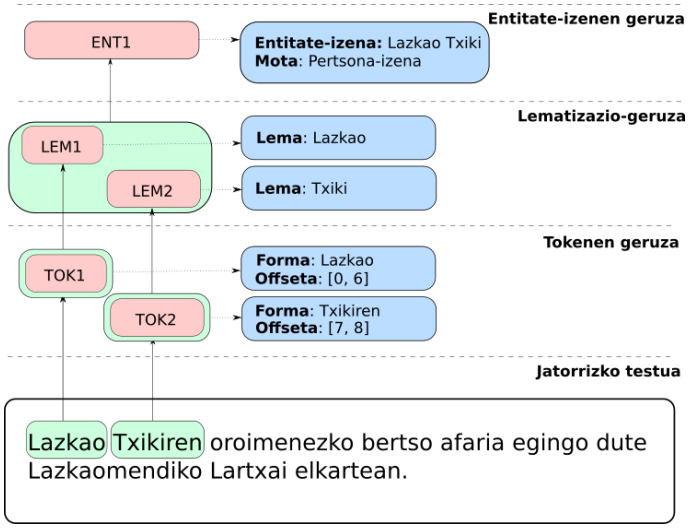
- **Estekak (anotazioak)**: Anotazioa bera osatzen dute, aingura bat analisi batekin lotuz. Estekek ahalbidetzen dute anotazio batean anbiguotasuna adieraztea, aingura jakin bat hainbat analisirekin lotuz, hainbat estekaren bidez. Hartara, anbiguotasuna ebazteko, nahikoa da esteketako bat zuzentzat markatzea. Bestalde, esteketan ere anotazioari berari buruzko bestelako atributuak gehi daitezke, ez daudenak zuzenean ez ainguarekin eta ez analisiarekin lotuta, bien arteko loturarekin (hots, anotazioarekin) baizik. Aipagarrienetako bat aingura-esteka bikotearen konfiantza maila da, ohikoa baita prozesatzaileek balio hori ere ematea. Era berean, anotazioa desanbiguatu bada, desanbiguazioa makinak automatikoki egin duen edo eskuz egin den ere estekaren atributu gisa adieraz daiteke. Anbiguotasunaz gain, aingura eta analisien arteko loturak bereizita adierazteko beste arrazoietako bat da analisi

bera aingura desberdinekin lotu daitekeela. Laburbilduz, aingura bakoitza analisi batekin baino gehiagorekin lotu daiteke, eta analisi bera aingura batekin baino gehiagorekin ere bai.

Aingura-anotazio-analisi hirukoteen izaera dela eta, AWAk standoff eredua jarraitzen du: anotazioak ez daude jatorrizko testuan bertan txertatuta (Carletta *et al.*, 2003); aldiz, beren geruzako beste anotazioekin batera mantentzen dira, eta ainguren erreferentzia bat gordetzen dute.

Anotazioen arteko erreferentziak, beste eredu gehienekin alderatuz gero, modu desberdinean gauzatzen dira. Analisi linguistikoak, guretzat, testu zati baten edo beste anotazio baten (edo batzuen) gaineko informazio linguistiko gehigarria ematen duten informazio egituratua dira (ikus 3.2 irudia). Informazio linguistikoan (irudian urdinez), gako-balio motako ezaugarriak gehi daitezke, baina ez dago beste anotazio baten erreferentziarik egiteko modurik, erreferentzia guztiak ainguran (irudian berdez) egiten baitira. Izan ere, aingura-esteka-analisi hirukoteen izaera dela eta, anotazioa zein beste anotazioren gainean definitzen den ainguran bertan adierazten da. Anotazioen arteko erlazio mota bakarra, beraz, A anotazioa B anotazioaren gainean definituta dagoenean gertatzen da. Kasu horretan, A-k B-ri buruzko informazio gehigarria ematen du, beste maila linguistiko batekoa normalean, eta B anotazioa A-ren aingura dela esaten dugu.

Orain arte esandakoa adibide batekin argituko dugu. Demagun 3.2 irudiko testuaren gaineko tokenizazioaren, lematizazioaren eta entitate-izenen identifikazioaren anotazio linguistikoak adierazi nahi ditugula. Testua *Lazkao Txikiren oroimenezko bertso afaria egingo dute Lazkaomendiko Lartxai elkartean.* izanik, irudian ikus daitezke dagozkion anotazioetako batzuk. Esaldian 11 token daude, puntu-ikurra barne. Entitate-izenak, berriz, hiru dira: Lazkao Txiki, Lazkaomendi eta Lartxai. Kasu honetan, testuko *Lazkao* eta *Txikiren* hitzak token banaren aingura dira (`TOK1` eta `TOK2`). Token bakoitzarentzat lematizazio-anotazio bat sortzen da lematizazio mailan (`LEM1` eta `LEM2`). Kasu honetan tokenak lematizazio-anotazioen aingura direnez, tokenak, gorriaz gain, berdez inguratuak ere badaude irudian. Azkenik, *Lazkao Txiki* entitatearen aingura, aipatutako bi lematizazio-anotazioek osatzen dute, entitate-izen hori ez baitago zuzenean testuko hitzen gainean definituta,



3.2 irudia: Anotazio linguistikoen itxura orokorra esaldi oso baten gainean (anotazioak gorriz, aingurak berdez eta anotazioen informazio linguistikoa urdinez).

lematizazio-geruzako `LEM1` eta `LEM2` anotazioen gainean baizik, eta, kasu honetan ere, hori da `LEM1` eta `LEM2` lauki berde baten barruan sartuta agertzearen arrazoia. Beraz, tokenek testu zati baten gaineko informazio linguistiko gehigarria ematen dute, lematizazio-anotazioek token baten edo token multzo baten gainekoa, eta, era berean, entitateek lematizazio-geruzako anotazioen gaineko informazioa ematen dute. Modu horretan ulertu behar dira, AWAren datu-ereduaren kasuan, anotazioak eta beren arteko erlazioak.

Adibidean ikusi dugun bezala, normalean, tokenizazio-geruza bat dago testuaren gainean. Testuko token bakoitzak bere token-anotazioa izan ohi du definituta. Hortik aurrerako geruzak tokenen geruzako edo haren gaineko beste geruzetako anotazioen gainean definitzen dira (3.2 irudiko `LEM1`, `LEM2` eta `ENT1`, adibidez).

Ikus dezagun anotazioen arteko erreferentziak hobeto ulertzeko beste adibide bat. AWAren dependentzia sintaktikoen anotazio bakoitzean, adibidez, beste anotazioen bi erreferentzia daude: burua eta modifikatzailea. 3.3 irudian bi dependentzia-anotazio dituen esaldi baten adibidea ikus daiteke. Esaldia *Mikelek azterketa gainditu du.* izanik, dependentzia bat *Mikelek gainditu-*

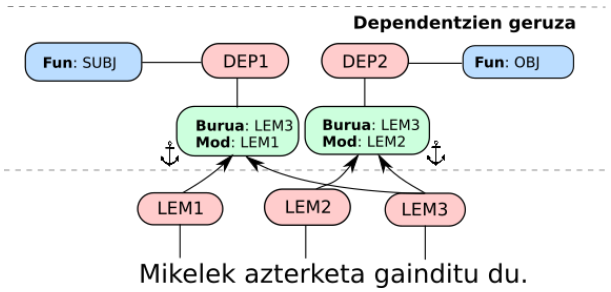
3.3 irudia: Bi dependentzia-anotazioren irudikapen grafikoa. Bi anotazioen burua *gainditu* da, eta modifikatzaileak *Mikelek* eta *azterketa*.

-ren subjektua dela adierazten duena da, eta bestea, *azterketa gainditu*-ren objektua dela adierazten duena. Bietan burua *gainditu* da, eta modifikatzaileak *Mikelek* eta *azterketa*. Funtzio sintaktikoak, berriz, `SUBJ` eta `OBJ` dira, hurrenez hurren. Dependentzietatik beren aingura diren lematizazio-anotazioetarako erreferentzia horiek, AWAren datu-ereduaren arabera, dependentzia-anotazioaren ainguran gauzatzen dira. Dependentzia-anotazio bat, beraz, bi lematizazio-anotazioren arteko erlazio baten gainean definitutako anotazio gisa uler daiteke, 3.4 irudian ikus daitekeen bezala.

Jarraian, datu-ereduaren inguruko hainbat kontzeptu xehetasun gehiagorekin azalduko ditugu. Lehenik eta behin, informazio linguistikoa adierazteko erabiltzen diren ezaugarri-egiturak aurkeztuko ditugu. Ondoren, anotazioen anbiguotasuna zertan datzan eta AWAn nola tratatzen den azalduko dugu. Azkenik, interpretazio-aingurak deituriko aingura mota bereziak aurkeztuko ditugu.

3.1.1.1. Ezaugarri-egitura motatuak

Anotazio baten informazio linguistikoa konplexua izan daiteke. Askotan informazioa atributu gutxi batzuetara mugatzen da. Adibidez, token-anotazio



3.4 irudia: Dependentzia-anotazioen adibidea.

batek ez du, normalean, atributu askorik edo konplexurik behar. Tokenari dagokion testuko stringa identifikatzeko atributuez gain (offseta eta luzera), beste bizpahiru atributu baino ez dira definitzen kasu gehienetan. Informazioaren konplexutasunak ez dauka zerikusirik, ordea, morfosintaxi-mailako anotazioetan. Kasu horretan, hitz bakoitzaren barne-egitura deskonposatu eta zehaztasunez adierazi behar da, eta horretarako egitura hierarkikoak ezinbesteko bihurtzen dira.

Ikus dezagun adibide bana aipatutako bi kasuetarako. Kasurik sinpleenean, token baten informazio linguistikoa adierazi nahi dugu, honako atributu hauetara mugatuz: tokenaren forma eta mota. Tokena *gogorregia* izanik, horrelako XML egitura sinple bat nahikoa genuke aipatutakoa bere osotasunean adierazteko:

```
1 <token mota="hitz">gogorregia</token>
```

Demagun orain, *gogorregia* tokenaren egitura morfosintaktikoa adierazi nahi dugula. Oina *gogor* izango da, eta jarraian bi atzizki ditu itsatsita: *egi* eta *a*. Demagun oinaren eta morfema bakoitzaren hainbat ezaugarri ere adierazi nahi ditugula. Gainera, oinaren eta morfemen ezaugarriez gain, hitz mailako ezaugarri morfosintaktiko orokorrak zein diren ere adierazi nahi dugu adibide honetan. Adierazi nahi dugun informazioaren izaera egituratua denez (hitzaren barruan morfemak daude, morfemen barruan ezaugarriak, etab.), egitura ez da aurrez ikusi dugun tokenen kasuan bezain sinplea. TEIk proposatutako ezaugarri-egitura motatuak erabiliko ditugu, 3.5 irudian bezalako zerbait lortuz[2]:

Adibidean ikus daitekeen bezala, sortzen den egitura nahiko konplexua da. Hain zuzen ere, `fs` (*feature structure* edo ezaugarri-egitura euskaraz) eta `f` (*feature* edo ezaugarri euskaraz) elementuekin eraikitzen dira ezaugarri-egitura motatuak. `fs` bakoitzak ezaugarri-egitura mota bat definitzen du, eta hainbat `f` elementuz osatzen da. `f` elementuen bidez ezaugarriak adierazten dira, eta bi motatako balioak har ditzakete, oinarrizko datu-mota batekoak

[2]adibide hau AWAz ez litzateke zehazki horrela adieraziko, hau AWAren sinplifikazio bat besterik ez baita

```
<fs type="morfosintaxia" id="M-A-ADJ-ARR-122">
  <f name="forma"><str>gogorregia</str></f>
  <f name="ezaugarriak">
    <fs type="ezaugarri-lista">
      <f name="kategoria"><str>ADJ</str></f>
      <f name="azpikategoria"><str>ARR</str></f>
    </fs>
  </f>
  <f name="oina">
    <fs type="lema">
      <f name="sarrera"><str>gogor<str></f>
      <f name="ezaugarriak">
        <fs type="ezaugarri-lista">
          <f name="kategoria"><str>ADJ</str></f>
          <f name="azpikategoria"><str>ARR</str></f>
        </fs>
      </f>
    </fs>
  </f>
  <f name="morfema">
    <fs type="lema">
      <f name="sarrera"><str>egi<str></f>
      <f name="ezaugarriak">
        <fs type="ezaugarri-lista">
          <f name="kategoria"><str>GRA</str></f>
        </fs>
      </f>
    </fs>
  </f>
  <f name="morfema">
    <fs type="lema">
      <f name="sarrera"><str>a<str></f>
      <f name="ezaugarriak">
        <fs type="ezaugarri-lista">
          <f name="kategoria"><str>DEK</str></f>
        </fs>
      </f>
    </fs>
  </f>
</fs>
```

3.5 irudia: TEIk proposatutako ezaugarri-egiturak erabiliz, horrela adierazten dira ezaugarri linguistikoak AWAn.

edo `fs`-ak, bat edo gehiago (listak edo multzoak). Behin ezaugarri-egitura eta ezaugarri motak definituta, berrerabili egin daitezke ezaugarri-egitura konplexuak osatuz. Ezaugarri baten barruan beste ezaugarri-egitura bat osoa definitzeak egitura hierarkikoak eraikitzeko aukera ematen du, ezinbestekoa ezaugarri linguistiko konplexuak adierazteko. Oinarrizko datu-moten artean, hauek dira eskuragarri daudenak:

- `str`: Ohiko stringa.
- `nbr`: Zenbakizko balioa.
- `plus` / `minus`: Balio boolearra.
- `sym`: Sinboloa: erabili behar da balioa aurredefinitutako zerrenda batetik aukeratu behar den kausetan.

`fs` elementuek har dezaketen atributu bakarra `type` da, ezaugarri-egituraren mota adierazten duena. Adibidez, morfemen informazio linguistikoa adierazteaan, `type` atributuaren balioa `morfema` izan daiteke, `fs` hori morfema bati dagokiola zehaztuz. Analisi linguistikoaren lehen mailako `fs`-ak `id` atributua ere hartu behar du, analisiaren identifikadorea izango dena. `f` elementuek, beren aldetik, bi atributu har ditzakete: ezaugarriari izena ematen dion `name` atributua, eta, bere balioa bakuna izan ordez, egitura konposatua bada, `org` atributua, zeinaren balio posibleak `list` eta `set` diren, eta `f`-aren barne-egitura zerrenda edo multzo bat dela adierazten duten, hurrenez hurren.

Formalismo hori erabiliz, AWAren maila linguistiko bakoitzaren analisien egitura zehatzak definitu ditugu RelaxNG (RNG) eskemak erabiliz[3]. Horrela, zein analisi mota ager daitezkeen, bakoitzak barruan zein elementu har ditzakeen, eta elementu bakoitzaren barne-egitura zein den formalki deskribatu ditugu.

3.1.1.2. Anbiguotasuna

Datu-eredua diseinatzeko garaian kontuan izan dugun ezaugarri bat anbiguotasuna izan da. Geruza bereko anotazio batek baino gehiagok aingura

[3]RNG eskema osoak hurrengo helbide honetan aurki daitezke: `http://ixa2.si.ehu.eus/~zbeloki001/awa_rng/`

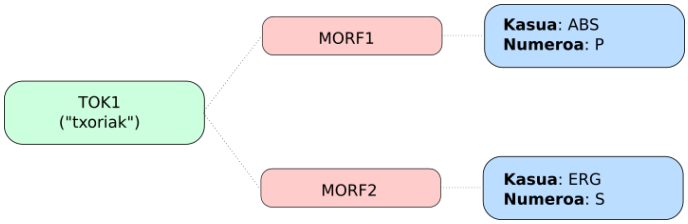
berbera daukatenean, anbiguotasuna dagoela esaten da. Adibidez, euskaraz ohikoa den *-ak* atzizkiak anbiguotasun morfosintaktikoa dakar berarekin. Morfosintaxi-mailako prozesatzaile batek, adibidez, *txoriak* tokena aurkitzen duenean, bi interpretazio emango dizkio tokenari: ergatibo singularra eta absolutibo plurala. Kasu horretan, beraz, bi analisi morfosintaktiko desberdin izango genituzke definituta aingura beraren (*txoriak* tokenaren) gainean.

Anbiguotasuna modu desberdinean tratatuko da aplikazioaren arabera. Batzuetan anotazio guztiak gorde nahi izango dira, baina besteetan anbiguotasuna ebatzi eta zuzenak ez direnak alde batera utziko dira. Hasteko, prozesatzaile guztiek ez dute anbiguotasuna identifikatu eta aingura bakoitzarentzat anotazio bat baino gehiago sortzeko gaitasunik. Kasu horietan ez dago kudeaketa berezirik egin beharrik, baina zer egin prozesatzaileek aingura bakoitzarentzat hainbat anotazio posible sortzen dituztenean? Honako hauek dira jokabide ohikoenak:

- Anbiguotasuna hasieratik ebatzi: Aplikazio jakin batzuetan nahikoa da, interpretazio anbiguoen aurrean, testuinguru horretan zuzena izateko probabilitate altuena daukanarekin bakarrik gelditzea, gainontzekoak baztertuz. Sinpletasuna bilatzen denerako aukera egokiena da, disko-espazioan eta prozesaketa-denboran aurreztea ekar baitezake.

- Interpretazio anbiguoak iraunkorki gorde: Hizkuntza aztertzea helburu duten corpusak prozesatzean, esaterako, ohikoa da anbiguotasunaren ondorioz sortzen diren anotazio guztiak gordetzea, horrek azterketa linguistiko sakonagoak egiteko aukera ematen baitu gerora. Erabakiak hartzeko ikasketa automatikoa erabiltzen duten aplikazioen kasuan ere interesgarria izan daiteke anotazioen interpretazio posible guztiak eskuragarri izatea.

- Anotazio anbiguoak epe laburrerako gorde: Hizkuntza-prozesatzaile guztiak ez dira gai anbiguotasuna ebazteko. Askotan, beste maila linguistiko bateko prozesatzaileek sortutako informazio linguistikoa beharrezkoa da horretarako. Beraz, kasu batzuetan, beharrezkoa da anbiguotasuna analisi-kate osoaren prozesaketak irauten duen denboran zehar mantentzea.



3.6 irudia: *txoriak* tokenaren bi interpretazio morfosintaktikoak. Horrela adierazten da AWAn anbiguotasuna, aingura berari hainbat analisi esleituz.

Anotazioen eta haien ainguren arteko erlazioen izaera dela eta, anbiguotasuna adierazteko oso modu naturala eta berezkoa dauka AWAren datu-ereduak, izan ere, anbiguotasuna egon edo ez, anotazio bakoitzaren adierazpidea berdin mantentzen baita. Horri esker, anotazioak inkrementalki sortzeko prozesua oso sinple egiten du AWAk. Izan ere, behin aingura eta analisi bana lotuz anotazio bat sortu ondoren, aingura berberari beste analisi bat esleitu behar bazaio, bigarren anotazio hori sortzea berehalakoa izango da, bi anotazioek ez baitute elkarren arteko dependentziarik izango, eta, beraz, aurretik beste analisi bat edukitzeak ez baitio inola ere eragingo (ikus 3.6 irudia).

3.1.1.3. Interpretazio-ainguruk

Orain arte ikusi dugunaren arabera, anbiguotasuna dagoenean, aingura baten gainean hainbat anotazio defini daitezke, bakoitza interpretazio linguistiko alternatibo bat izanik. Zenbaitetan, ordea, anotazio anbiguoak ezaugarri komun baten edo batzuen arabera multzokatuz sortzen dira interpretazio alternatiboak. Izan ere, anotazio baten informazio linguistikoa zenbat eta konplexuagoa izan, anbiguotasun maila altuagoa izateko arriskua dago. Kasu askotan, ordea, ezaugarri jakin batzuk bakarrik hartu nahi ditugu kontuan, eta beraz, ezaugarri horien balioak berdinak dituzten anotazioak bakarra bezala tratatu nahi ditugu. AWAren datu-ereduak interpretazio-ainguren bidez ebazten du arazo hori, anotazioak ezaugarri komun baten edo batzuen arabera multzokatzea ahalbidetuz, ondoren, aingurak multzo horiekin eraikitzeko.

Har dezagun, adibide bezala, *publikoak* hitza. Lema *publiko* izanik, izena (`IZE`) edo adjektiboa (`ADJ`) izan daiteke, testuinguruaren arabera. Lemati-

zatzaileak hitzei funtzio sintaktikoak ere esleitzen dizkienez, demagun adjektibo-interpretazioari bi funtzio sintaktiko esleitzen dizkiola, eta beste bi izen-interpretazioari. 3.7 irudiko lehenengo lau lerroetan lematizatzaileak *publikoak* hitzaren kasuan sortutako emaitza ikus daiteke. Anotazio bakoitza `lSfI`[4] bat da, eta bai izenaren eta bai adjektiboaren interpretazioei objektu (`@OBJ`) eta subjektu (`@SUBJ`) funtzio sintaktikoak esleitu dizkie. Orain, demagun maila semantikoan, Euskal WordNet-en adibidez, adiera bana dagoela *publiko* izenari eta adjektiboari lotuta. Ez litzateke zehatzena bi adierak *publiko* tokenari zuzenean lotzea. Aldiz, izenari dagokion adiera izen-interpretazioari esleitu beharko litzaioke, eta adjektiboari dagokiona, adjektibo-interpretazioari. Horretarako, bi aingura berri sortuko genituzke, bakoitzak kategoria bereko lematizazio-anotazioak bilduz (3.7 irudiko erdialdea). Aingura horiei (`lSfISet1`[5] eta `lSfISet2`) interpretazio-aingura deritzegu, eta horiekin lotuko genituzke semantikoki erlazionatuta dauden interpretazioak, `wsdI1` eta `wsdI2` anotazio semantikoak lortuz, 3.7 irudiko azken bi lerroetan ikus daitekeen bezala. Interpretazio-ainguren erabilerak hainbat onura eskaintzen ditu:

- Hainbat interpretazio multzokatzea ahalbidetzen du, edozein granularitate mailatan. Lehengo adibidean, interpretazioak kategoriaren arabera bildu ditugu, nahiz eta funtzio sintaktikoak berdinak ez izan. Horri esker, adiera semantiko bakoitza kategoria jakin bati lotu ahal izan diogu.

- Multzokatzeko aukera izanik, lehertze konbinatorioa ekidin daiteke. Adibidez, bi kategoria posible, hiru funtzio sintaktiko posible eta bost adiera dituen hitz batek 30 anotazio semantiko beharko lituzke. Interpretazioak kategoriaka multzokatuz, bost anotazio aski lirateke, adiera bakoitzeko bana.

Interpretazio-ainguren izaera eta erabilera hobeto ulertzeko, sintaxi-mailako adibide bat ere azalduko dugu. Hala ere, azpimarratu nahi dugu interpreta-

[4] `lSfI` izena ingelesezko *lemmatization syntactic function interpretation* terminotik dator, eta, funtsean, funtzio sintaktikoen interpretazio bat dela esan nahi du.

[5] `lSfISet` motako aingura batek `lSfI` anotazioen multzo bat errepresentatzen du.

```
publikoak :: lSfI4 :: publiko.ADJ.ABS.P
publikoak :: lSfI7 :: publiko.ADJ.ERG.S
publikoak :: lSfI8 :: publiko.IZE.ABS.P
publikoak :: lSfI11 :: publiko.IZE.ERG.S

// Interpretazio-aingurak:
lSfISet1 = { lSfI8, lSfI11 }
lSfISet2 = { lSfI4, lSfI7 }

lSfISet1 :: wsdI1 :: euswn-publiko.ize.1
lSfISet2 :: wsdI2 :: euswn-publiko.adj.1
```

3.7 irudia: *publikoak* hitzaren interpretazio posibleak. Erabilitako notazioa: *Aingura::Anotazioa(esteka)::Informazio linguistikoa*.

zio-aingurak datu-eredu orokorraren ezaugarri bat direla, eta, beraz, ez daudela maila linguistiko jakin batzuei bakarrik lotuta. Sintaxi-mailako adibidea aurkezteko chunker-aren irteera erabiliko dugu. Chunkerrak azaleko sintaxia egiten du, esaldiak zatitan (*chunk*) banatuz. Adibidez, *Mendizorrotzako ekitaldi publikoak ikusi ditu* esaldia bi zatitan banatuko luke: *Mendizorrotzako ekitaldi publikoak* eta *ikusi ditu*. 3.7 irudiko adibidean lematizazioaren irteera gainetik azaldu dugu. Atal honetan, sakonago aztertuko dugu.

Lematizatzaileak interpretazio bat baino gehiago eslei diezaioke hitz bakoitzari (ikus 3.8 irudiko `LSfI` instantziak). *publikoak* hitzari arreta jarriz gero, analizatzaileak bi kategoria identifikatu dituela ikus daiteke, izena eta adjektiboa. Bien kasuan, deklinabide-kasu eta numero bikote desberdinak identifikatu dituela ere ikus daiteke: absolutibo/plural (`ABS/P`) eta ergatibo/singular (`ERG/S`) bikoteak. Gainera, absolutibo kasuari objektu (`@OBJ`), subjektu (`@SUBJ`) eta predikatu (`@PRED`) funtzio sintaktikoak esleitu dizkio, eta ergatibo kasuari, subjektua bakarrik.

Lehenago aipatu bezala, interpretazio-aingurei esker konbinazio kopurua asko murriztu daiteke, eta azaleko sintaxiaren kasuan oso onuragarria da hori. Zatiak anotatzeko orduan, `LSfI` kategoria eta kasu/numero berdinak dituzten interpretazioak bilduz sortzen dira aingurak, funtzio sintaktikoak ez baitira esanguratsuak maila honetan. Horrela, adibidean (3.9 irudia), lau aingura (`LSfISet`) sortu dira *publikoak* hitzarekin. *ikusi* hitzaren kasuan, aditz- eta izen-kategoriak identifikatu dira. Lehenari `@-JADNAG` funtzio sintaktikoa es-

```
Mendizorrotzako :: lSfI1 :: Mendizorrotza.IZE.LOK.S.DEF.@NCOMPL>
Mendizorrotzako :: lSfI1 :: Mendizorrotz.IZE.LOK.S.@NCOMPL>

ekitaldi :: lSfI3 :: ekitaldi.IZE.@KM>

publikoak :: lSfI4 :: publiko.ADJ.ABS.P.@OBJ
publikoak :: lSfI5 :: publiko.ADJ.ABS.P.@PRED
publikoak :: lSfI6 :: publiko.ADJ.ABS.P.@SUBJ
publikoak :: lSfI7 :: publiko.ADJ.ERG.S.@SUBJ
publikoak :: lSfI8 :: publiko.IZE.ABS.P.@OBJ
publikoak :: lSfI9 :: publiko.IZE.ABS.P.@PRED
publikoak :: lSfI10 :: publiko.IZE.ABS.P.@SUBJ
publikoak :: lSfI11 :: publiko.IZE.ERG.S.@SUBJ

ikusi:: lSfI12 :: ikusi.ADNAG.PART.@-JADNAG
ikusi:: lSfI13 :: ikusi.IZE.ABS.INDEF.@OBJ
ikusi:: lSfI14 :: ikusi.IZE.ABS.INDEF.@PRED
ikusi:: lSfI15 :: ikusi.IZE.ABS.INDEF.@SUBJ

ditu :: lSfI16 :: *edun.ADLAG.PRES.3SG_ABS.3SG_ERG.@+JADLAG
ditu :: lSfI17 :: ukan.ADNAG.PRES.3SG_ABS.3SG_ERG.@+JADNAG
```

3.8 irudia: Lematizazioaren irteera *Mendizorrotzako ekitaldi publikoak ikusi ditu* esaldia analizatuta.

```
// Mendizorrotzako
lSfISet1 = { lSfI1 }
lSfISet2 = { lSfI2 }

// ekitaldi
lSfISet3 = { lSfI3 }

// publikoak
lSfISet4 = { lSfI4, lSfI5, lSfI6 }    // adj. interpr.
lSfISet5 = { lSfI7 }                  // adj. interpr.
lSfISet6 = { lSfI8, lSfI9, lSfI10 }   // ize. interpr.
lSfISet7 = { lSfI11 }                 // ize. interpr.

// ikusi
lSfISet8 = { lSfI12 }                 // adi. interpr.
lSfISet9 = { lSfI13, lSfI14, lSfI15 } // ize. interpr.

// ditu
lSfISet10 = { lSfI16 }
lSfISet11 = { lSfI17 }
```

3.9 irudia: Lematizatzaileak sortutako interpretazioak multzokatzen.

```
// Mendizorrotzako ekitaldi publikoak
lSfISetSeq1 = < lSfISet1, lSfISet3, lSfISet4 >
lSfISetSeq2 = < lSfISet1, lSfISet3, lSfISet5 >

// ikusi ditu
lSfISetSeq3 = < lSfISet8, lSfISet10 >

lSfISetSeq1 :: chunkI1 :: IS.ABS.P.lSfISet3
lSfISetSeq2 :: chunkI2 :: IS.ERG.S.lSfISet3

lSfISetSeq3 :: chunkI3 ::
```

3.10 irudia: Interpretazio multzoen sekuentzien bidez, hurrengo pausoetan erabiliko diren interpretazio-aingura konplexuak osa daitezke.

leitu zaio, eta bigarrenari objektu, subjektu eta predikatu funtzio sintaktikoak.

Beraz, kontuan izanik hitz bakoitzak interpretazio multzo bat edo gehiago izan dezakeela, zati-anotazio baten aingura interpretazio multzoen sekuentzia bat (`LSfISetSeq`[6]) izango da (3.10 irudia). Adibideko lehenbiziko zatiaren kasuan, bi `LSfISetSeq`-i lotuta ager daiteke. Biek hitz-kate berbera erreferentziatzen duten arren (*Mendizorrotzako ekitaldi publikoak*), interpretazio desberdinari egiten diote erreferentzia: lehenak *publiko* hitzaren absolutibo/plural intepretazioari eta bigarrenak ergatibo/singular interpretazioari. Ondorioz bi zati-anotazio lortzen dira, eta bakoitzak dagokion informazio linguistikoa dauka atxikita: `IS.ABS.P.lSfISet3` eta `IS.ERG.S.lSfISet3`, hurrenez hurren. Zehazki, informazio linguistiko horrek honako ezaugarri hauek biltzen ditu: sintagma mota, deklinabide-kasua, numeroa eta zatiaren burua zein den adierazten duen aingura.

3.1.2. AWAren anotazio-eskema

Azaldutako datu-ereduaren gainean inplementatu dugu AWA. Izan ere, datu-eredu batean, formalismo eta egitura orokor bat adostu arren, ez da anotazioen semantikarik definitzen. Hau da, datu-eredu horretako formalismoari jarraituz, atributu jakin batzuk definitu behar dira, bakoitzari bere esan-

[6] `lSfISetSeq` izena `lSfI` anotazio multzoen (*set*) sekuentzia (*seq*) izatetik datorkio

gura linguistikoa emanez. AWAk geruzatan antolatutako eredua jarraitzen duenez, maila linguistiko bakoitzarentzat anotazio mota bat definitu dugu. Sortzen diren analisi guztiek, halabeharrez, mota horietakoren batekoak izan behar dute, eskema ez baita hedagarria.

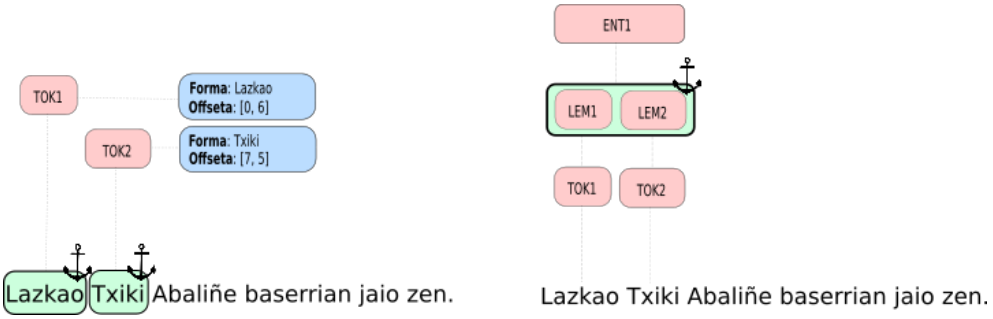
AWAk honako anotazio-geruzak ditu inplementatuta: tokenak, HAULak (hitz anitzeko unitate lexikalak), segmentazio-anotazioak, morfosintaxi-anotazioak, dependentzia-zuhaitzak, lematizazio-anotazioak, entitateak, korreferentzia-anotazioak, azaleko sintaxi-anotazioak eta osagaietan oinarritutako sintaxia.

Mota bakoitzeko anotazioek beren informazio linguistikoaren egitura propioa dute. Aingura motak ere ugariak dira. Jarraian, AWAren inguruko xehetasun gehiago azalduko ditugu. Lehenik eta behin, diseinatu ditugun anotazioen aingura motak aurkeztuko ditugu. Ondoren, AWAren bi serializazioak deskribatuko ditugu: testu-dokumentu mailakoa eta corpus mailakoa. Azkenik, AWA erabiltzen duten aplikazioak aztertuko ditugu.

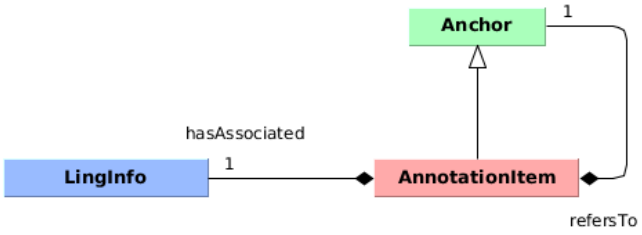
3.1.2.1. Anotazioen aingurak

Anotazio linguistiko orok aingura bat izan behar du, anotazioaren informazio linguistikoa zein elementuri dagokion adierazten duena. Izatez, anotazio bat aingura eta analisi bana lotzean datza. 3.11 irudiaren ezker aldean berdez koloreztatutako bi aingura ikus daitezke: *Lazkao* eta *Txiki* testu zatiak, `TOK1` eta `TOK2` token-anotazioak haien gainean definitu baitira. Eskuin aldean berdez koloreztatuta ageri den aingura, berriz, `LEM1` eta `LEM2` anotazioek osatzen dute, entitate-anotazioa haien gainean definitu baita.

Aingurak, gehienetan, aurrez definitutako beste anotazio batzuez osatzen dira, baina testuz (aurreko paragrafoko adibidean ikusi dugun bezala) edota bestelako balio atomiko batzuez ere osa daitezke, kasu berezietan. AWAren datu-ereduaren irudi orokorra berreskuratzen badugu, ainguretan zentratuz eta aingurekin loturarik ez duten gainontzeko elementuak baztertuz, 3.12 irudiko eskema lortuko genuke. Irudiaren arabera, anotazio guztiek (`AnnotationItem`) daukate aingura bat (`refersTo` erlazioa), eta anotazioak aingura ere izan daitezke (horregatik azpiklase-erlazioa).

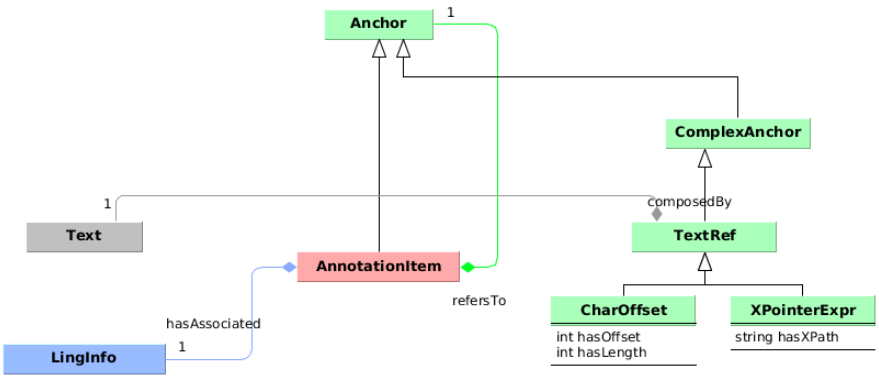


3.11 irudia: Anotazioen aingurek informazio linguistikoa zein elementuri buruz ematen den zehazten dute.



3.12 irudia: Datu-ereduaren eskema orokorra, ainguretan zentratuz.

Aipatu berri dugun ainguren kasu berezi horietako bat, izatez, testu zatiak identifikatzen dituzten aingurei dagokie. Bereziak dira, kasu honetan, token-anotazioen kasuan bakarrik erabili ohi direlako, hau da, anotazio mota jakin batean bakarrik erabiltzen direlako, eta ez gutxi erabiltzen direlako, testu-dokumentu baten anotazioak adierazi nahi diren kasu ia guztietan erabili behar izaten baitira. Testua erreferentziatzeko aingurak bi azpimultzotan sailkatu daitezke: offset bidezko erreferentziazkoak (`CharOffset`) eta XPath bidezkoak (`XPointerExpr`). Lehenbizikoak zenbaki motako bi atributuz osatzen dira: testu zatiaren lehenbiziko karakterearen offseta eta luzera. Bigarrenak, XPath espresio bat daukan karaktere-kate batekin definitzen dira. 3.12 irudiko eskeman ez da horrelako aingurarik aurreikusten. Horregatik, aingura konplexuak definitu ditugu (`ComplexAnchor`), eta klase horretatik eratorriko dira anotazio soil bat izatetik haratago doazen aingura guztiak. 3.13 irudian aurreko eskemaren hedapen bat ikus daiteke, aingura konplexuak gehitu-



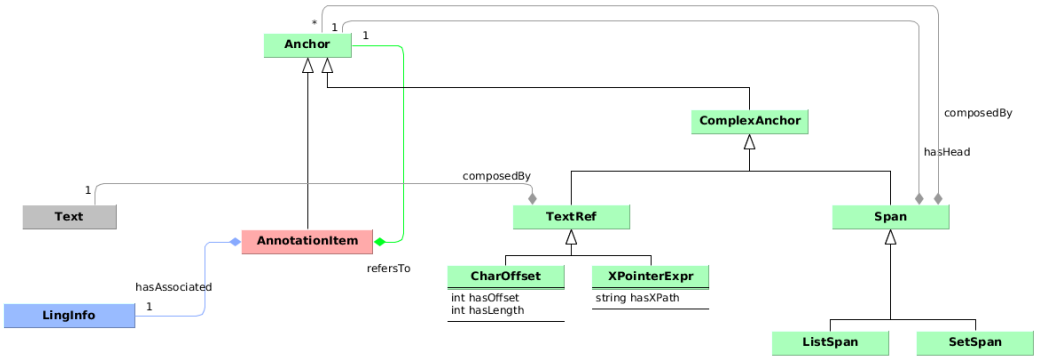
3.13 irudia: AWAren ainguren eskema, aingura konplexuak, eta zehazki TextRef aingurak, gehituta.

ta, eta, konkretuki, testu zatiak identifikatzeko aingurak gehituta (irudiko `TextRef` elementua).

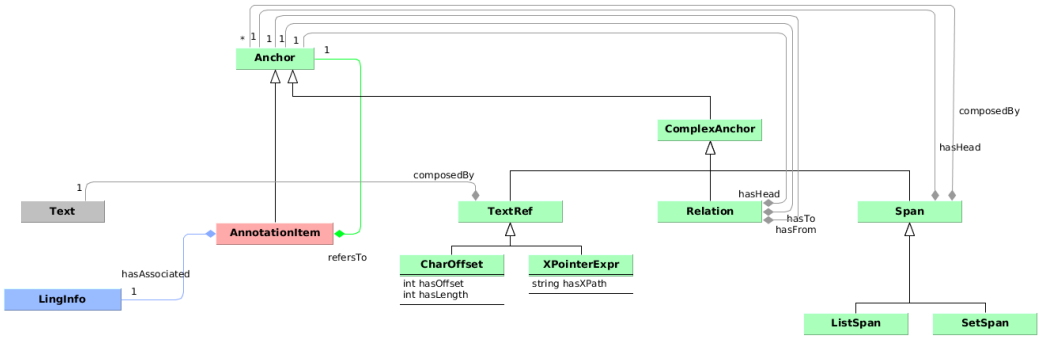
Zenbaitetan, aingurak hainbat anotazioz osatzen dira. Adibidez, entitate--anotazioen aingurak hainbat lematizazio-anotazioz osatzen dira, entitate bat hainbat terminok eratu baitezakete. 3.2 irudian ikusi dugun adibidean, `ENT1` (Lazkao Txiki) anotazioaren aingura `LEM1` (Lazkao) eta `LEM2` (Txiki) anotazioek osatzen zuten. Horrelakoetan ere aingurak egitura konplexuago baten beharra daukanez, *span* egiturak (`Span`) definitu ditugu (ikus 3.14 irudia). Span aingurek anotazio multzo bat har dezakete, eta horien artean burua[7] zein den adierazteko atributua ere badute. Spanak gehiago xehatuz, bi span mota bereiz ditzakegu: zerrenda motako spanak (`ListSpan`) eta multzo motakoak (`SetSpan`). Lehenbizikoen kasuan, ainguraren parte diren anotazioen arteko ordena esanguratsua da, eta bigarrenean ez.

Aipatutakoez gain, beste aingura mota baten beharra ere ikusi dugu gure proiektuetan zehar: erlazio-aingurak (`Relation`). Dependentzia sintaktikoen kasuan, esaterako, aingura bi lematizazio-anotaziok osatzen dute. Anotazio bikote horiek, ordea, ezingo genituzke span-aingura batekin behar bezala adierazi, izan ere, ez baitugu anotazioen zerrenda edo multzo bat adierazi nahi, anotazio batetik beste batera doan erlazio zuzendu bat baizik. Esan

[7] Anotazio multzo baten elementu nagusia.



3.14 irudia: AWAren ainguren eskema, spanak gehituta.



3.15 irudia: AWAren ainguren eskema, erlazio-aingurak gehituta.

bezala, erlazio-aingurak definitu ditugu kasu horietarako, honako atributu hauekin: `from` (erlazioaren jatorri-elementua), `to` (helburu-elementua) eta `head` (burua). 3.15 irudian AWAren ainguren eskema ikus daiteke, aingura mota guztiak identifikatu eta gero.

Orain arte azaldutako aingura motekin anotazio linguistiko gehienen aingurak adierazteko gaitasuna daukagu. Askotan, aldiz, anotazio mota bakoitzarentzat aingura mota propioa sortu nahi izango dugu. Adibidez, entitate-anotazio baten aingura gisa zerrenda motako span bat (`ListSpan`) erabil genezakeen arren, bertan lematizazio-anotazioen zerrenda bat definituz, `ListSpan` batek, izatez, zerrendan edozein motatako anotazioak gehitzea onartzen du. Hori konpontzeko, `EntitySpan` aingura mota berri bat eratorrarazi dugu

`ListSpan` elementutik, bere osagaiak lematizazio-anotazio motakoak izan behar dutelako murriztapena gehituz.

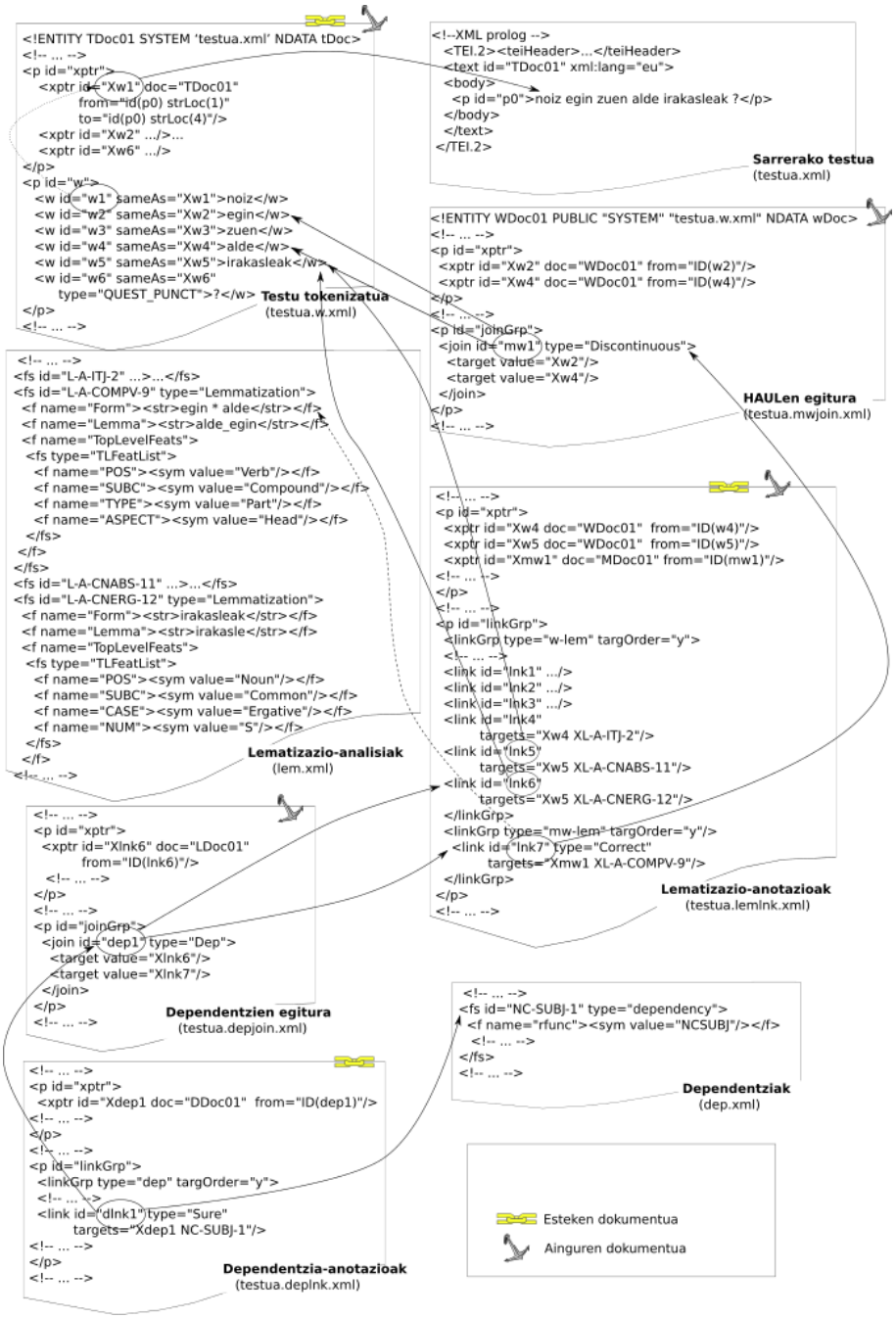
3.1.2.2. Testu-dokumentu mailako serializazioa

3.1.1 atalean deskribatu dugun egituraren arabera, anotazio-eskema baten datu-ereduak serializazio-eskema bat behar du anotazioak tresnen edo gizakien artean elkarbanatu edota sisteman biltegiratu ahal izateko. AWA serializatzeko, bi adierazpide fisiko inplementatu ditugu: atal honetan aurkeztuko dugun testu-dokumentu mailakoa (XML fitxategietan oinarritua) eta hurrengo atalean deskribatuko dugun corpus mailakoa (datu-baseetan oinarritua).

Erabiliena testu-dokumentu[8] mailako serializazioa da. Kasu honetan, prozesatutako testu-dokumentu bakoitza XML dokumentu-sare batean serializatzen da. 3.1 taulan ikus daitekeen bezala, informazio mota bakoitza XML dokumentu batean biltzen da, eta, era berean, XML dokumentu bakoitza fitxategi fisiko batean. Esaterako, token-analisiak *.w.xml* luzapena duen fitxategi batean gordetzen dira, eta lematizazio-analisiak *.lem.xml* luzapena duen beste batean. Gainera, token bakoitza bere lematizazio-analisiekin lotzen duten estekak ere beste fitxategi batean gordetzen dira, *.lemlnk.xml* luzapenarekin. Era berean, HAULak *.mwlnk.xml* luzapena duen fitxategian gordetzen dira. Ikus daiteke nola testu-dokumentu bakoitzeko hainbat XML dokumentu sortzen diren, dokumentu-amaraunak osatuz. Corpus-dokumentu bakoitzaren anotazio-amarauna osatzen duten XML dokumentuek elkarren arteko lotura eta erreferentziak egiten dituzte. Kontuan izanik corpus bakoitzak testu-dokumentu asko izan ditzakeela, erraz uler daiteke zergatik deitu diogun ereduari Anotazio Amaraunen Arkitektura. 3.16 irudian ikus daiteke testu-dokumentu baten gainean eraikitako anotazio-amarauna.

AWAren serializazio hau TEI P4 gidalerroetan oinarritzen da (Sperberg-McQueen eta Burnard, 2001). Anotazioen egituraren definizioa ezaugarri-egituren bidez egiten denez, orokorra da eskema mailan, hau da, serializaziorako erabiltzen diren XML elementuak berdinak dira geruza eta anotazio mota guztietan zehar. Gainera, aingurak, analisiak eta estekak serializatzeko

[8] Corpuseko dokumentuak eta anotazioen XML dokumentuak bereizteko, testu-dokumentu eta XML dokumentu deitu diegu, hurrenez hurren.



3.16 irudia: Testu-dokumentu mailako serializazioaren eskema. Inplementazio hau XML dokumentuetan oinarritzen da.

Anotazio mota	Aingurak	Estekak	Informazio linguistikoa
Tokenizazioa	.TEI	.w.xml	.w.xml
HAULak	.w.xml	.mwlnk.xml	.mwlnk.xml
Segmentazioa	.w.xml	.seglnk.xml	.seg.xml
Morfosintaxia	.w.xml	.morflnk.xml	.morf.xml
Lematizazioa	.w.xml	.seglnk.xml	.seg.xml

3.1 taula: AWAren XML serializazioaren arabera sortzen diren XML dokumentuak. Anotazioen osagai bakoitza zein fitxategitan aurkitzen den ikus daiteke.

elementu orokorrak dauzka AWAk. Beraz, esan dezakegu serializazioa sinplea eta intuitiboa dela. Salbuespen bakarra tokenen eta HAULen serializazioa da, elementu horiek adierazpide propioa baitute TEI gidalerroen arabera, eta gidalerro horiek jarraitu ditugunez, hori ere bere horretan utzi dugu.

Token bakoitza `w` elementu batean serializatzen da. Tokenaren aingurak TEIk proposatzen dituen `xptr` elementuen bidez adierazten dira. `xptr` elementu batekin amarauneko beste dokumentuetako elementu baten erreferentzia gordetzen da, kasu honetan, jatorrizko testuko token batena. Horretarako karaktere-posizioen erreferentziak (offsetak) erabiltzen dira. Ondoren, `w` elementua `xptr` horrekin lotzen da. `w` bakoitzak, tokenaren formaz gain, hainbat atributu har ditzake:

- `id`: Tokenaren identifikadorea.
- `sameAs`: Tokena `xptr` batekin lotzen du, atributuari `xptr` elementuaren identifikadorea esleituz.
- `type`: Tokenari buruzko informazio gehigarria. Puntuazio-ikurrak, testuan letra larriz idatzita daudenak, zenbakiak etab. bereizteko erabiltzen da.
- `rend`: Tokena jatorrizko testuan nola zegoen aurkeztuta edo renderizatuta adierazten du. Adibidez, testua zentratuta edo azpimarratuta.

TEIren `join` elementuen bidez elementu ez-jarraituekin osatutako testu zati

bat identifika daiteke. AWAn HAULak adierazteko erabiltzen dira. HAUL bakoitzak honako atributu hauek hartzen ditu:

- `id`: HAULaren identifikadorea. Normalean, *mwX* forma hartzen dute, *X* HAUL zenbakia izanik.
- `type`: `Correct` edo `NoCorrect` balioak har ditzake. Atributu honek desanbiguazioa egin ondoren hartzen du zentzua, eta testuinguru horretan zuzena den edo ez adierazten du.

`join` elementu bakoitzaren barruan, HAULa osatzen duen token bakoitza `target` elementu batekin adierazten da. Tokenak beste XML dokumentu batean definituta daudenez, haiei egin behar diete erreferentzia `target` elementuek. Horretarako, atributu hauen balioak definitu behar dira:

- `name`: Erreferentzia mota. Kasu honetan erreferentzia tokenei egin nahi diegunez, `wId` balioa hartuko du.
- `value`: Tokenen kasuan bezala, HAULen kasuan ere `xptr` elementuak erabiltzen dira amarauneko XML dokumentuen arteko erreferentziak egiteko. Horretarako, `xptr` elementu bakoitzak token bati erreferentzia egiten dio. Horrela, `value` atributuari `xptr` baten identifikadorearen balioa esleituz, token bati erreferentzia egitea lortzen da.

Informazio linguistikoa, hots, maila desberdinetako analisi linguistikoei dagokien informazioa, TEIren gidalerroetan definitutako ezaugarri-egituren bidez adierazten da (ikus 3.1.1.1 atala).

Ezaugarri-egituren formalismoa TEIk definitzen duen arren, maila linguistiko bakoitzeko analisiak adierazteko ezaugarri-egitura propioa diseinatzea norberaren esku gelditzen da. Kasu honetan, AWAk bere ezaugarri-egitura propioak dauzka definituta. Definizio guztiak RNG eskemen bidez formalizaztu ditugu.

Aingura eta analisien arteko estekak ere TEI gidalerroak jarraituz egiten dira, `link` elementuak erabiliz. Besterik gabe, esteka bakoitza `link` elementu batekin adierazten da, eta bi atributu ditu:

- `targets`: Estekak lotzen dituen ainguraren eta analisiaren identifikadoreak, zuriune batez bereizita.

- `type`: Atributu honek estekaren esleipena eskuz edo automatikoki egin den adierazten du.

3.1.2.3. Corpus mailako serializazioa

Dokumentu mailako serializazioaz gain, datu-baseen bidez inplementatutako corpus mailako serializazioa ere inplementatu dugu AWAn. Datu-base erlazionalak eta XML datu-baseak, biak erabiltzen dira, kasu honetan, corpusari dagokion anotazio-amaraun osoa biltegiratzeko. Horrela, informazio linguistikoa Berkeley XML datu-base batean biltzen da, eta token eta estekak MySQL datu-base erlazional batean. Serializazio honen abantaila nagusietako bat, kontsultak egiteko azkartasunaz gain, corpus mailako anotazioen biltegi izatearena da. Horri esker, dokumentuetan zehar behin eta berriz errepikatzen diren formei dagozkien analisiak, zenbait mailatan behintzat (lematizazio-mailan, esaterako), behin bakarrik gorde daitezke, errepikapenik gabe, disko-espazioan, eraginkortasunean eta mantengarritasunean irabaziz. 3.17 irudian corpus mailako serializazioaren eskema bat ikus daiteke.

Informazio linguistikoa Berkeley XML datu-baseetan gordetzen denez, `fs` eta `f` elementuen bidez adierazten dira kasu honetan ere, testu-dokumentu mailako serializazioan bezala. Kontuan hartu beharreko bakarra analisi berdinak errepikatuta ez biltegiratzea da. Horretarako, lematizazioen kasuan, adibidez, testuko hitz-formari dagokion analisia datu-basean badagoen egiaztatzen da lehenbizi, eta, baldin badago, horren erreferentzia besterik ez da gordetzen.

Bestalde, analisiak Berkeley XML datu-baseen edukiontzietan sailkatzen dira. Horrela, analisien gaineko kontsultak eraginkorragoak dira, definitutako indizeak edukiontzi mailakoak baitira. Maila linguistiko bakoitzeko analisiak edukiontzi batean gordetzen dira.

Berkeley XML datu-baseak datu egituratuak eta hierarkikoak gordetzeko aukera egokia dira, baina badaude izaera hierarkikoa ez duten datuak gordetzeko modu eraginkorragoak. Horretarako datu-base mota eraginkorragoak dira

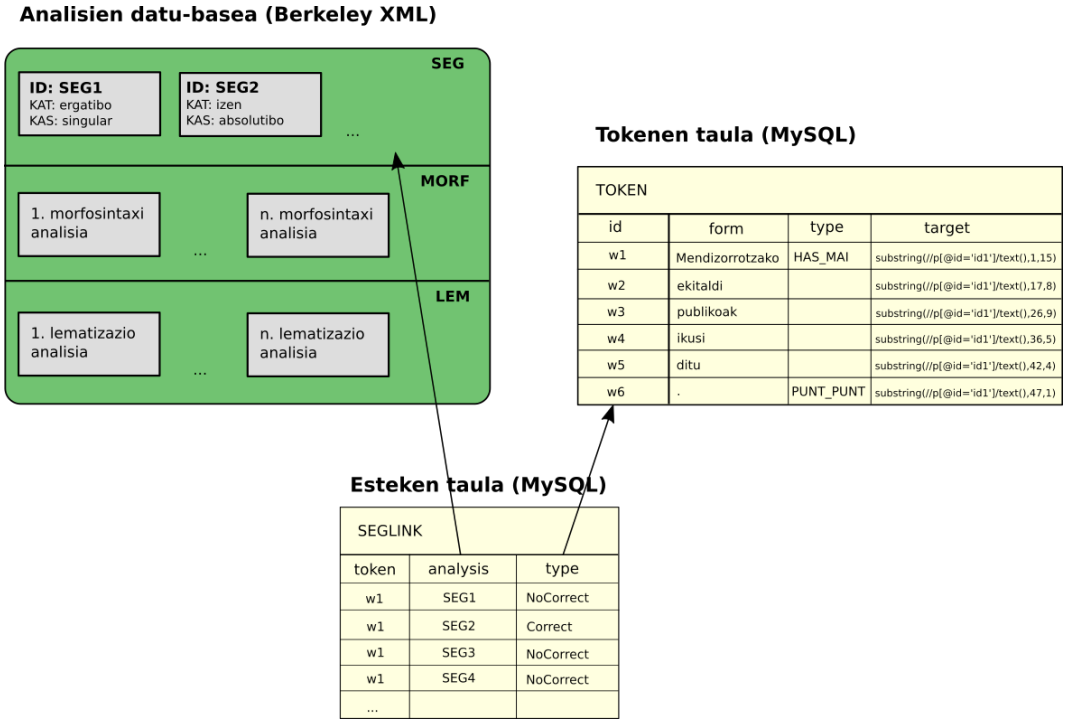
erlazionalak. Hori dela eta, estekak datu-base erlazionaletan gordetzen dira, izan ere, esteka bakoitzak analisi bat eta bere aingura identifikatu besterik ez baitu egin behar. Horrekin batera, tokenak eta HAULak ere datu-base erlazionalean gordetzen dira, arrazoi hauengatik:

- Elementu bereziak dira: TEIn beren egituraren definizio propioa dute, gainontzeko analisiek ez bezala. Unitate nahiko estandarrak dira, gaur egun, hizkuntzaren prozesamenduan.
- Egitura sinplea dute: Token eta HAULen egitura ez da hierarkikoa. Atributu kopuru finitu bat dauka, denak maila berekoak.
- Ohiko elementuak dira kontsultetan. Aplikazio mailan, oso ohikoa da HAULak eta, batez ere, tokenak atzitzea, analisi gehienen aingurak tokenez osatzen baitira.

Horrenbestez, tokenak `W` taulan gordetzen dira, HAULak `MW` taulan, eta gainontzeko geruzetako bakoitzaren estekak, bakoitza taula berri batean. Horietan denetan, XML serializazioaren atributu berberak erabiltzen dira informazioa adierazteko. Testu-dokumentu mailako serializazioarekin gertatzen zen moduan, hemen ere egitura komunak erabiltzen dira geruza eta anotazio mota desberdinetarako, serializazioa sinple eta intuitibo mantenduz.

3.1.2.4. AWAren ekosistema

AWA natiboki darabilten tresnak bi dira: *Morfeus*, euskararako analizatzaile morfosintaktikoa, eta *Eustagger*, euskararako etiketatzaile eta desanbiguatzaile morfosintaktikoa. Hala ere, AWAk maila linguistiko gehiagorentzako adierazpidea ere eskaintzen du, eta natiboki erabili ez arren, postprozesatzaile batek honako beste tresna hauen irteerak AWAren eredura bihurtzen ditu: Ixati, euskarazko testuen azaleko sintaxi-mailako prozesatzailea, entitate-izenak identifikatzeko Eihera+ tresna integratua duena; Edgk/MaltIxa, analizatzaile sintaktiko-estatistikoa; EusWN, Wordneten euskarazko bertsioaren gainean lan egiten duen algoritmo bati esker hitzen adiera-desanbiguazioa egiten duen tresna; eta entitate-izenak dagozkien Wikipediako orriekin lotzen dituen desanbiguatzailea.



3.17 irudia: Corpus mailako serializazioaren eskema. Inplementazio hau datu-baseetan oinarritzen da.

Ereduaren inguruan ekosistema bat osatu da urteetan zehar, hainbat tresna baitaude AWArekin lan egiten dutenak. Aipatu berri dugu testuak prozesatuz emaitza AWAren ereduari jarraituz ematen duten hizkuntza-prozesatzaileak zein diren. AWArekin lan egiten laguntzen duten beste tresna batzuk ere badaude, ordea:

- **LibiXaML**: AWA anotazioak sortzeko, kontsultatzeko eta editatzeko liburutegia. C++ lengoaiaz inplementatuta dago, eta ehundik gora klase dauzka. Aingura eta estekekin lan egiten laguntzeaz gain, informazio linguistikoa xehetasunez tratatzen du, AWAren ezaugarri-egiturekin lan egiteko klaseak ere inplementatuta baititu.

- **Armiarma**: Lexikoaren Behatokia proiektuaren baitan garatutako tresna da, corpusen gainean kontsultak egin eta anbiguotasuna eskuz ebaz-

teko balio duena. Horretarako, corpusak AWA formatuan egon behar du, datu-baseen bidezko inplementazioari jarraituz. Noski, dokumentu mailako inplementaziotik corpus mailakora bihurketak egiteko tresna ere eskuragarri dago.

- **EULIA**: Hizkuntza-prozesatzaileen kateak definituz testu-dokumentuak prozesatu eta sortutako AWArekin lan egiteko interfaze grafikoa eskaintzen duen aplikazioa. Sortutako analisiak modu bisualean aztertzea ahalbidetzen du, eta baita manualki desanbiguatzea ere. Armiarmaren aurrekaria da.

- **Abar-hitz**: Euskararen zuhaitz-bankua sortu eta kontsultatzeko interfaze grafikoa. Bereziki, hizkuntzalariei dependentzia-zuhaitzak eraikitzen laguntzea du helburu.

3.2. NLP Annotation Format (NAF)

NAF anotazio linguistikoak adierazteko eskema bat da. (Fokkens *et al.*, 2014). KYOTO proiektuan[9] sortu eta erabilitako *Kyoto Annotation Format* (KAF) (Bosma *et al.*, 2009) eskeman du jatorria. Izatez, KAF eredua NewsReader[10] proiektuaren beharretara egokitzean sortu zen NAF.

Izaera orokorreko anotazio-eskema izanik, maila linguistiko anitzetako anotazioak adierazteko ahalmena dauka. Une honetan, token-anotazioez gain, terminoak, dependentzia-erlazioak, osagaietan oinarritutako zuhaitz sintaktikoak, azaleko sintaxia, entitate-izenak, korreferentzia, iritziak, atribuzioa, rol semantikoak, denbora-espresioak, gertakizunak, faktualtasuna eta testuaren gaien gaineko anotazioen egiturak dauzka definituta.

NAFen ezaugarrietako bat bere standoff izaera da, anotazioak jatorrizko testutik bereizita mantentzen baititu. Beraz, alde batetik, jatorrizko testua dago, eta beste alde batetik, anotazio linguistikoak, geruzatan sailkatuta, nahiz eta dokumentu berean dauden.

[9] `http://www.kyoto-project.eu` (kontsulta: 2017-05-08)
[10] `http://www.newsreader-project.eu` (kontsulta: 2017-05-08)

NAF dokumentuak, gehienetan, XML dokumentuetan adierazten dira: XML dokumentu bana testu dokumentu bakoitzeko. XML oso hedatuta dagoen markaketa-lengoaia izanik, anotazio linguistikoak adierazteko lengoaia egokia da, besteak beste, bere egitura hierarkikoak eta elementuen arteko erreferentziak adierazteko gaitasuna dela eta. Irakurleak ideia bat egin dezan, 3.18 irudian sinplifikatutako NAF dokumentu baten adibidea ikus dezake.

NAF dokumentuek hiru zati nagusi dauzkate: goiburua (`nafHeader`), jatorrizko testua (`raw`) eta anotazio linguistikoak (`text`, `terms`, `entities`...). Goiburuan jatorrizko dokumentuaren inguruko metainformazioa gehitzen da: testuaren iturria, sortze-data etab. Horrekin batera, testua zein hizkuntza-prozesatzailerekin (HP) prozesatu den ere bertan ikus daiteke (`linguistic-Processors`). Adibidean, esaterako, ikus daiteke nola lau HPk hartu duten parte prozesaketan. Standoff eredua izanik, jatorrizko testua hasieran kokatzen da eta anotazioak ondoren, bereizita eta elkarren artean erreferentziak eginez. Anotazioak geruzatan multzokatzen dira (adibideko `text`, `terms` eta `entities`), eta multzo bakoitza maila linguistiko bateko anotazioek osatzen dute (`wf`, `term`, `entity`...). Hurrengo ataletan xehetasun gehiagorekin azalduko ditugu NAFen ezaugarriak.

NAF diseinatzean kontuan izan zen NAF dokumentuak testu-corpus erraldoiak prozesatuz sortuko zirela. Horrenbestez, anotazioak prozesu paraleloen bidez sortzeko gaitasunak berebiziko garrantzia zeukan. Hori lortzeko, anotazio-geruzek, posible den neurrian, elkarrekiko izaera independentea daukate. Horrela, paraleloan exekutatzen diren prozesuek dokumentu berari dagozkion anotazioak sor ditzakete. Geruza batzuek elkarren artean dependentziak dauzkate halabeharrez, eta ondorioz, kontuz aukeratu behar da zein izango diren paraleloan sortuko diren anotazio-geruzak.

Bestalde, anotazio-eskema lehenago aipatutako proiektuetan aplikatzeko sortu zenez, erabilerara zuzendua dago, hau da, praktikoa izatea izan du hasieratik helburu nagusi. AWA landutako datu-eredu baten gainean inplementatuta dagoen bezala, ez da hori NAFen kasua, nolabaiteko datu-eredu bat badaukan arren, geruza batetik bestera zenbaitetan aldatu egiten baita aingurak eta erreferentziak egiteko modua. Dena dela, badauka oinarrizko egitura komun bat, NAFen datu-egitura osatzen duena eta anotazio mota

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<NAF version="v3" xml:lang="en">
  <nafHeader>
    <fileDesc creationtime="2003-01-04"/>
    <public publicId="N432" uri="2003/1/4/N432.xml"/>
    <linguisticProcessors layer="text">
      <lp name="ixa-pipe-tok-en" timestamp="2013-11-22 3:49:05"/>
    </linguisticProcessors>
    <linguisticProcessors layer="terms">
      <lp name="ixa-pipe-pos-en" timestamp="2013-11-22 3:49:06"/>
    </linguisticProcessors>
    <linguisticProcessors layer="entities">
      <lp name="ixa-pipe-nerc-en" timestamp="2013-11-22 3:49:14"/>
      <lp name="ixa-pipe-spotlight" timestamp="2013-11-22 3:49:16"/>
    </linguisticProcessors>
    ...
  </nafHeader>
  <raw>Ford to sell Volvo?
  Ford's recent financial woes, combined with news that Renault has
  secured a huge bank loan, have pundits linking the French car maker
  to a possible purchase of Volvo, reprising merger attempts of the
  early 90s.
  </raw>
  <text>
    <wf id="w1" length="4" offset="0" sent="1">Ford</wf>
    <wf id="w2" length="2" offset="5" sent="1">to</wf>
    <wf id="w3" length="4" offset="8" sent="1">sell</wf>
    <wf id="w4" length="5" offset="13" sent="1">Volvo</wf>
    ...
  </text>
  <terms>
    <term id="t1" lemma="Ford" morphofeat="NNP" pos="R" type="close">
      <span>
        <target id="w1"/> <!--Ford-->
      </span>
    </term>
    ...
  </terms>
  <entities>
    <entity id="e1" type="organization">
      <references>
        <span>
          <target id="t1"/> <!--Ford-->
        </span>
      </references>
      <externalReferences>
        <externalRef
            reference="http://dbpedia.org/resource/Ford_Motor_Company"
            resource="spotlight_v1"/>
      </externalReferences>
    </entity>
    ...
  </entities>
</NAF>
```

3.18 irudia: NAF dokumentu baten adibidea.

```
<terms>
  <term id="t1" lemma="the" pos="D">
    ...
  </term>
  <term id="t2" lemma="newspaper" pos="N">
    ...
  </term>
</terms>
<deps>
  <!-- NMOD(newspaper, the) -->
  <dep from="t2" to="t1" rfunc="NMOD" />
</deps>
```

3.19 irudia: Dependentzia-anotazioek terminoei egiten diete erreferentzia, identifikadoreak erabiliz.

guztietara hedatzen dena.

NAFen datu-egitura oso sinplea da. Geruzatan antolatutako eskema denez, anotazio mota desberdinak eskaintzen ditu. Geruza bakoitzeko anotazioen informazioa desberdina den arren, geruza guztietarako balio duten oinarrizko formalismo batzuk daude, NAFen datu-eredua osatzen dutenak.

3.2.1. Anotazioen arteko erreferentziak eta aingurak

Anotazioek testu zati bati edo beste anotazioei buruzko informazio linguistiko gehigarria ematen dute. Hortaz, eta standoff ereduari jarraituz, anotazioen artean erreferentziak egiteko formalismoa beharrezkoa da. AWAren inguruan esan dugu, bere datu-eredu landua dela eta, erreferentziak anotazioen ainguretan bakarrik egiten direla. NAFen kasuan, datu-eredua sinpleagoa izanik, erreferentziak anotazioaren edozein atribututan ager daitezke, aingurak eta informazio linguistikoa ez baitaude fisikoki bereizita. Anotazio bakoitzak dokumentu mailan unibokoa den identifikadore bat daukanez, eta anotazio guztiak dokumentu berean bilduta daudenez, erreferentzia egiteko atributuaren baliotzat erreferentziatutako anotazioaren identifikadorea jarri besterik ez da egin behar. 3.19 irudiko adibidean dependentzia batek bere jatorri-anotazioari eta helburu-anotazioari egindako bi erreferentzia ikus daitezke, `dep` anotazioaren `from` eta `to` atributuetan.

Erreferentziekin lotuta, AWAko ainguren kontzeptua berreskuratuko dugu. Anotazio baten aingura, anotazio hori zein testu zatiren edo zein anotazioren gainean eraiki den adierazten duen informazioa dela esaten dugu. Hortaz, ikusi berri dugun adibidean, `dep` anotazioaren aingura `t1` eta `t2` anotazioek osatzen dute, dependentzia-anotazioa bi termino-anotazio horien gainean definitu baita.

Anotazio-eskema baten datu-eredua diseinatzeko orduan, onuragarria da aingurak adierazteko formalismo komun bat deskribatzea, anotazio guztiek erabiliko dutena. AWAren kasuan garrantzia handia eman zaio aspektu horri, aingurak gainontzeko elementuetatik bereiziz eta izaera propioa eskainiz, baita adierazpen fisikoaren mailan ere. NAFen, ordea, ez da gauza bera gertatzen, eta ainguren kontzeptua lausoagoa da, praktikotasunaren mesedetan beti ere. Horren adibide aurrez ikusi dugun dependentzia-anotazioaren aingura, ezin baita begi bistaz bereizi `from` eta `to` atributuek anotazioaren aingura osatzen duten ala anotazioaren informazio linguistikoaren parte diren. Hala ere, hiru aingura mota definitzeko formalismoak dauzka NAFek zehaztuta: testu zati bat erreferentziatzeko aingurak, anotazio zerrenda motako aingurak eta bi anotazioren arteko erlazio bat erreferentziatzeko aingurak.

Aingura gehienak anotazio zerrenda baten bidez osatzen dira, eta kasu horietarako egitura berezi bat eskaintzen du NAFek: *span* egitura. Horrenbestez, 3.20 irudiko adibidera jotzen badugu, entitateen aingurak termino zerrenda batez osatzen direla kontuan hartuta, horrela adieraziko genituzke haien aingurak.

Ikusi bezala, adibideko termino bakoitzak token (NAFen *word form*) bati egiten dio erreferentzia, baina token bat baino gehiagori erreferentzia egin beharko balie, `span` elementuaren barruan token bakoitza `target` elementu baten bidez adieraziko litzateke.

Adibidean dagoeneko ikusi dugun beste aingura mota bi anotazioren arteko erlazio baten erreferentzia egiten duena da. Horretarako, anotazio bati `from` eta `to` atributuak definitu behar zaizkio, bakoitzak erlazioaren jatorri eta helburuko anotazioari erreferentzia eginez.

Ohikoa den beste aingura mota bat testu zatiak erreferentziatzekoa da. NA-

```
<terms>
  <!--Paul-->
  <term id="t4" type="close" lemma="Paul" pos="R" morphofeat="NNP">
    <span>
      <target id="w4"/>
    </span>
  </term>
  <!--Newman-->
  <term id="t5" type="close" lemma="Newman" pos="R" morphofeat="NNP">
    <span>
      <target id="w5"/>
    </span>
  </term>
</terms>
<entities>
  <entity id="e1" type="person">
    <references>
      <!--Paul Newman-->
      <span>
        <target id="t4"/>
        <target id="t5"/>
      </span>
    </references>
  </entity>
</entities>
```

3.20 irudia: `span` elementuekin anotazio zerrenda bati egiten zaio erreferentzia. Adibide honetan, Paul Newman entitateak Paul eta Newman terminoei, biei, `span` bakarrarekin egiten die erreferentzia.

Fen kasuan, zeregin horretarako XML elementu berezirik definitu ez den arren, *offset* eta *length* atributuak erabiltzen dira. 3.21 irudiko adibidean hainbat tokenen definizioak ikus daitezke. Bakoitzak testu zati jakin bati erreferentzia egiten dio.

3.2.2. Anotazio txertatuak

Anotazioen arteko loturak egiteko ez dira beti erreferentziak erabiltzen. Anotazio batzuk beste mota jakin bateko anotazioekin lotuta agertzen dira beti. Menpekotasun hori dela eta, erreferentzia kopurua murriztu eta eredua sinpleago egiteko, anotazio horiek besteen barruan txertatzen dira, egitura hierarkikoak sortuz.

Hori gertatzen da, adibidez, rol- eta predikatu-anotazioen kasuan. Rol-ano-

```
<text>
  ...
  <wf id="w680" length="3" offset="3712" sent="33">the</wf>
  <wf id="w681" length="9" offset="3716" sent="33">newspaper</wf>
  <wf id="w682" length="1" offset="3725" sent="33">,</wf>
  ...
</text>
```

3.21 irudia: Tokenek (`wf`) offsetak erabiliz egiten diete erreferentzia dagozkien testu zatiei.

tazio guztiak predikatu-anotazio bati dagozkio, eta, beraz, hari lotuta sortzen dira. Lotura horiek erreferentzia bidez ebaztea posible litzateke, baina kasu honetan rolek predikatu bati lotuta egon ezean zentzurik ez dutenez, rolak zuzenean predikatuen barruan txertatzen ditugu, 3.22 irudian ikus daitekeen bezala.

```
<predicate id="pr1">
  <span> <!-- Predikatua zein terminori dagokion -->
    <target id="t1"/>
  </span>
  <role id="rl1" semRole="A1"> <!-- "pr1" predikatuarekin lotutako
                                   rol-anotazio bat -->
    <span> <!-- Rola zein terminori dagokion -->
      <target id="t4" />
      <target id="t5" />
    </span>
  </role>
</predicate>
```

3.22 irudia: Batzuetan, anotazioek elkarren erreferentzia gorde ordez, bata bestearen barruan definitzen dira, haien arteko lotura estua den kasuetan. Horrela gertatzen da, irudian ikusten den bezala, rol-anotazioekin.

Egitura horrek, gainera, beste arazo bat ekiditeko ere balio dezake. Aurreko adibidean (3.22), erreferentziak erabili izan bagenitu rolaren aingura predikatua dela adierazteko, rolak bi aingura lituzke: bata rola zein predikaturi dagokion adierazteko, eta bestea, rola zein terminok osatzen duten adierazteko. Beraz, erreferentziak soilik erabili beharko bagenitu, aingura

konplexuagoen beharra izango genuke, AWAn gertatzen den bezala. Hala ere, hasieratik esan dugun bezala NAF ahalik eta sinpleen mantendu nahi izan dugunez, anotazioak txertatzearen alde egin dugu.

3.2.3. Anotazioen identifikadoreak

NAFen datu-ereduaren kasuan, anotazio mota gehienek har dezakete identifikadore bat, `id` atributua erabiliz. Noski, identifikadoreek unibokoak izan behar dute dokumentu mailan. Identifikadoreen egiturari dagokionez, geruza bakoitzak identifikadoreen lehen karaktereak izango diren kode bat izan behar du. Geruza mailan unibokoa den zenbaki batekin amaituko da identifikadorea. Adibidez, tokenen kodea `w` izanik, identifikadore posible bat `w1` litzateke, eta entitateen kodea `e` izanik, identifikadore posible bat `e7` litzateke.

Anotazio mota batzuk ez dira inoiz erreferentziatuak izaten, ez baitira inoren aingura inoiz. Kasu horietan identifikadorerik ere behar ez dutenez, anotazio mota batzuek ez dute identifikadore-atributurik definituta. Horixe da, adibidez, dependentzia-anotazioen kasua.

3.2.4. NAF eta datu estekatuak

Bestalde, NAF oso erlazionatuta dago datu estekatuen kontzeptuarekin, hasieratik erabaki baitzen anotazioak bestelako baliabideekin (ezagutza-baseak, ontologiak...) lotzea bultzatu behar zela. Horrela, egitura berezi bat diseinatu zen anotazioak bestelako baliabideetako elementuekin lotzeko: `externalRef` elementua.

Adibidez, 3.23 irudian bezala lotuko genuke *Paul Newman* entitatea bere DBpediako sarrerekin (Mendes *et al.*, 2011).

Anotazio bakoitza nahi adina baliabiderekin estekatu daiteke, baliabidearen eta baliabideko elementuaren identifikadoreak adieraziz. Gainera, erreferentzia mota, erlazioaren egoera, anotazioaren sortzailea izan den HPa eta konfiantza maila adierazteko hautazko eremuak ere defini daitezke. Adibidean *Paul Newman* entitatea DBPediako bi sarrerarekin lotu da, aktorearekin eta musika taldearekin, hurrenez hurren. Hala ere, konfiantza-balioek garbi uz-

```
<entity id="e47" type="person">
  <references>
    <span>
      <!--Paul Newman-->
      <target id="t571"/>
      <target id="t572"/>
    </span>
  </references>
  <externalReferences>
    <externalRef resource="spotlight_v1"
                 reference="http://dbpedia.org/resource/Paul_Newman"
                 confidence="1.0" />
    <externalRef resource="spotlight_v1"
                 reference="http://dbpedia.org/resource/Paul_Newman_(band)"
                 confidence="0.0" />
  </externalReferences>
</entity>
```

3.23 irudia: NAFen erraza da anotazio bat kanpoko ezagutza-base batekin lotzea. Irudiko adibidean, *Paul Newman* entitatea DBPedia Spotlighteko Paul Newman aktorearekin eta Paul Newman musika taldearekin lotu da. Konfiantza balioaren bidez, testuinguru jakin horretan aktoreari dagokiola adierazi da.

ten dute, testuinguru horretan, entitatea aktoreari dagokiola. 3.24 irudiko adibidean, anotazio bat hainbat baliabiderekin nola lotu daitekeen ikus daiteke. Horretarako, predikatu-anotazio bat PropBank, VerbNet eta FrameNet baliabideekin lotu da.

3.2.5. NAFen ekosistema

NAF urte gutxitan asko zabaltzen ari den anotazio-eskema da. Izan ere, NewsReader bezalako dimentsio handiko proiektu batean sortu eta erabili izanak, eta KYOTO eta OpeNER[11] proiektuetan sortu eta erabilitako KAF eskematik jaio izanak ospe handia eman dio.

NAF dokumentuak sortu eta editatzeko lana errazteko, hainbat liburutegi garatu dira lengoaia desberdinetarako. Pythonerako, esate baterako, Py-

[11] `http://www.opener-project.eu` (kontsulta: 2017-05-08)

```
<predicate id="pr10">
  <!--bought-->
  <externalReferences>
    <externalRef reference="buy.01" resource="PropBank"/>
    <externalRef reference="get-13.5.1" resource="VerbNet"/>
    <externalRef reference="Commerce_buy" resource="FrameNet"/>
    <externalRef reference="contextual" resource="EventType"/>
  </externalReferences>
  <span>
    <target id="t55"/>
  </span>
  ...
</predicate>
```

3.24 irudia: Hainbat `externalRef` elementurekin, anotazio bat ezagutza-base bat baino gehiagorekin lotu daiteke.

naf garatu zuten[12]. Tesi honen baitan Kaflib liburutegia garatu dugu[13] Java lengoaiarako, eta, izenarengatik pentsa daitekeen bezala, hasiera batean KAF dokumentuentzat sortu arren, gaur egun NAF dokumentuentzat egokituta dago. Liburutegia Java lengoaiaz idatzita dago, eta edozein geruzatako anotazioak sortu eta editatzeko funtzioak dauzka, mota askotako kontsultak egiteko funtzio eraginkorrak ere eskainiz.

NAF dokumentuak irakurri eta sortzen dituzten hizkuntza-prozesatzaileen kopurua ere handia da. Euskararako, bi tresna sorta daude: IXA pipes tresnak (Agerri *et al.*, 2014b) eta ixaKat (Otegi *et al.*, 2016). IXA pipes hizkuntza anitzetarako izanik, euskarazko testuak prozesatzeko gai diren honako tresna hauek eskaintzen ditu: tokenizatzailea, kategoria-etiketatzea, entitate-izenen ezagutzailea eta zatikatzaile edo *chunker*-a. ixaKatek, bere aldetik, honako tresna hauek eskaintzen ditu: tokenizatzailea, lematizatzailea, analizatzaile sintaktikoa, korreferentzia-ebazlea, rol semantikoen etiketatzailea eta zatikatzailea.

Ingeleserako, aukera zabalagoa da. Batetik, ingeleserako IXA pipes tresnak daude, honakoak egiten dituztenak: tokenizazioa, kategoria-etiketatzea, analisi sintaktikoa, dependentzia-etiketatzea, entitate-izenen ezagutzea eta

[12] `https://github.com/ixa-ehu/pynaf` (kontsulta: 2017-05-08)
[13] `https://github.com/ixa-ehu/kaflib` (kontsulta: 2017-05-08)

sailkapena, entitate-izenen desanbiguazioa, wikifikazioa[14], korreferentziaren ebazpena, rol semantikoen etiketatzea eta testuen sailkapena. Bestalde, NewsReader proiektuaren baitan garatutako tresna ugari dago eskuragarri, beste hauek egiteko gai direnak: denbora-espresioen ezagutzea, hitzen adiera--desanbiguazioa, gertakizunen korreferentziaren ebazpena, denbora-erlazioen erauzketa, erlazio kausalen erauzketa, faktualtasunaren ezagutzea eta iritzien erauzpena.

Gaztelaniarako ere IXA pipes tresnak eta NewsReader proiekturako sortutako beste hainbat tresna daude eskuragarri, eta ataza hauek gauzatzeko gai dira: tokenizazioa, kategoria-etiketatzea, analisi sintaktikoa, dependentzia-etiketatzea, denbora-espresioen ezagutzea, entitate-izenen ezagutzea eta sailkapena, hitzen adiera-desanbiguazioa, entitate-izenen desanbiguazioa, wikifikazioa, korreferentziaren ebazpena, rol semantikoen etiketatzea, gertakizunen korreferentziaren ebazpena eta testuen sailkapena.

Horiez gain, besteak beste, nederlandera, frantsesa eta italierarako ere NAF darabilten tresnak eta analisi-kate osoak ere eraiki eta erabili dira.

Aipatutako tresna guztien inguruko informazio osatuagoa IXA pipes-en[15] eta NewsReader proiektuaren webguneetan aurki daiteke.

[14]Terminoak DBPedia Spotlight-en arabera desanbiguatzen ditu.
[15]`http://ixa2.si.ehu.es/ixa-pipes` (kontsulta: 2017-05-08)

4. KAPITULUA

Anotazio linguistikoen arteko elkarreragingarritasuna

Hizkuntzaren prozesamenduan, anotazio linguistikoen bidez errepresentatzen da analizatutako testuei dagokien informazio linguistikoa. Mundu ideal batean, anotazioak adierazteko eskema estandar bakarra legoke eta hura erabiliko litzateke aplikazio guztietan. Baina errealitatea oso bestelakoa da, hizkuntzen arteko desberdintasunak, hizkuntza-teknologien arloko proiektuen helburu eta ikuspuntu desberdinak, hizkuntzalarien arteko desadostasunak eta teoria linguistiko desberdinen ezaugarriak direla medio anotazio-eskema berriak sortzen baitira han eta hemen, proiektu edota lantalde desberdinen baitan.

Baliabideen aberastasun horrek arazoak dakartza iturri desberdinetatik eskuratutako tresnak integratzen saiatzean, eta, izatez, soluzio zaila daukan arazoa da gainera. Kapitulu honetan arazo horren inguruan jardungo dugu. Hasteko, arazoa bera zertan datzan xehetasun gehiagorekin aurkeztuko dugu, eta, ondoren, arazoari aurre egiten lagunduko duen proposamen bat aurkeztuko dugu.

4.1. Elkarreragingarritasunaren arazoa

Hizkuntza prozesatzeko kate ugari dago munduan zehar. Katea norberak sortutako moduluekin eraiki bada, HPen arteko komunikazioa nola egingo den ere kontuan hartuko zen ezer inplementatu aurretik, eta, beraz, HPen

arteko integrazioa arazorik gabe egingo dela pentsa daiteke. Ohikoa da, ordea, beste norbaitek sortutako HPak erabiltzea prozesaketa-kateak eraikitzeko, gaur egun eskura daukagun kalitatezko HPen aukera oso zabala baita. Horren adibide dira, esate baterako, Freeling (Padró eta Stanilovsky, 2012), CoreNLP (Manning *et al.*, 2014) eta IXA pipes (Agerri *et al.*, 2014b). Orduan agertzen dira arazorik handienak, iturri desberdinetatik eskuratutako HPak integratzea oso lan nekeza izan baitaiteke (Hellmann *et al.*, 2013). HP baten irteeraren formatua eta kateko hurrengoaren sarrerarena bat ez badatoz, bigarrenak ez du jasotako datuak interpretatu eta erabiltzeko gaitasunik izango. Horrelakoetan, arazoak ekiditearren, irteera-formatu berbera darabilten HPak erabiltzera mugatzen dugu geure burua, horrek prozesaketa-kate ahaltsuak eraikitzeko dakartzan muga guztiekin. Aldiz, elkarren formatuak ulertzen ez dituzten HPak konbinatzea erabakiz gero, HPek sortutako datuak formatu batetik bestera pasatuko dituzten moduluak eraikitzera behartuta gaude, eta anotazio-eskemen konplexutasuna zenbaterainokoa den kontuan izanda, hori ere lan nekeza bihurtzen da.

Beste batzuetan, prozesaketa-katea osatuta egon arren, HP jakin bat beste batekin ordezkatzea ere ohikoa izaten da, helburu bera duten HP alternatiboen artean onena aukeratzeko. Horrelakoetan ere arazo berberaren aurrean aurki gaitezke, oso litekeena baita HP berri horiek erabiltzen duten anotazio-eskema desberdina izatea.

Elkarreragingarritasunaren arazoak sailkatzeko Chiarcosen (2012b) sailkapenean oinarritu gara. Chiarcosek elkarreragingarritasun estrukturala eta kontzeptuala bereizten ditu. Lehenbiziko kasuan, egitura aldetik desberdinak diren anotazio-ereduen arteko elkarreragingarritasuna lantzen du, eta bigarren kasuan, berriz, kontzeptuak izendatzeko desadostasunak dakartzan elkarreragingarritasun-arazoak lantzen ditu. Jarraian, sailkapen horretan oinarrituta identifikatu ditugun elkarreragingarritasun-arazoen mailak azalduko ditugu:

- Kasurik sinpleenean, datu jakin bat izen desberdineko atributuetan ematen da. Adibidez, anotazio baten konfiantza maila adierazten duen balioa `confidence` atributuan adierazten da FoLiA eta NAF anotazio-eskemen kasuan, baina baliteke beste eskemaren batek `conf` izeneko atributua erabiltzea. Sintaxia litzateke desberdintasun bakarra, infor-

mazio berbera emango bailukete bi atributuek. Era berean, atributuen balio gisa erabilitako terminoak ere desberdinak izan daitezke, kontzeptu berbera adierazi arren. Arazoetako bat, beraz, atributuak eta balioak izendatzeko sailkapen desberdinak erabiltzeak dakarrena da. Arazo hori Chiarcosen elkarreragingarritasun kontzeptualaren baliokidea da. Chiarcosek OLiA ontologien biltegia erabiltzea proposatzen du, zeinetan hainbat anotazio-eskemaren atributu, izen eta kontzepturen arteko mapaketak biltzen diren. Tesi-lan honetan ez dugu elkarreragingarritasuna maila honetan landuko.

- Balio atomikoez ari garenean, ohikoa da, lehenbiziko kasuan ikusi dugun bezala, izendapen-arazoak besterik ez izatea. Askotan, ordea, anotazioek egitura konplexuagoa izaten dute, eta datuak konposatuak (hainbat balioz osatutakoak) edota hierarkikoak (modu hierarkikoan antolatutako hainbat balioz osatutakoak) izan daitezke. Horrela, informazio jakin bat adierazteko granularitatea aldatu egin daiteke, eredu bakoitzak informazio horri ematen dion garrantziaren edo konplexutasunaren arabera. Horren adibide da NAFen eta AWAn informazio morfologikoa adierazteko erabiltzen diren moduen arteko desberdintasuna (4.1 irudia). NAFen, `lemma` atributuan hitzaren lema ematen da, eta `pos` atributuan kategoria gramatikala. Gainontzeko informazio morfologikoa `morphofeat` izeneko atributu bakun batean kodetuta ematen da. AWAn, berriz, ezaugarri-egitura konplexuak erabiltzen dira informazio morfologikoa kodetzeko, eta ematen den informazioa bera ere aberatsagoa da. Arazo hau Chiarcosen elkarreragingarritasun estrukturalaren parekoa litzateke. Hau da tesi-lan honetan landuko dugun elkarreragingarritasun maila.

- Gaur egun informazioaren adierazpenerako oso zabalduta dagoen XML lengoaia da anotazio linguistikoak adierazteko ere gehien erabiltzen dena. Hala ere, baliteke zenbaitetan XMLren antzekoa den JSON bezalako lengoaiaren batez adierazita egotea, edota, hainbatetan gertatzen den bezala, era jakin batean antolatutako testu hutsez. Beraz, anotazioak maila fisikoan ere desberdin adierazita ager daitezke. Chiarcosek berak maila fisikoa kontuan hartzen ez duen arren, aintzat hartu beha-

rreko arazoa iruditzen zaigu, praktikan arazoak izango baititugu maila fisikoan besterik bereizten ez diren anotazioak erabiltzen dituzten tresnak integratzeko ere.

Ikusi, esate baterako, 4.2 irudiko adibidean ageri diren bi anotazioak. Biak anotazio-eskema desberdinei dagozkie, eta itxura ere oso desberdina dute. Hala ere, baliokideak dira ematen duten informazioari dagokionez, egitura eta sintaxia baitira aldatzen diren bakarrak.

Lehenbiziko adibidean XMLz adierazitako entitate-anotazio bat ikus daiteke, besteak beste. Bai anotazioak, bai aingurak eta bai informazio linguistikoa bereizita adierazten dira anotazio-eskema jakin horretan. Bi tokenek osatzen dute entitatearen aingura, entitatea bera *Lazkao Txiki* baita, eta tokenak *Lazkao* eta *Txiki*. Informazio linguistikoa `linginfo` elementu batean adierazten da, eta entitate mota zein den esaten du (kasu honetan "IZEN_BEREZIA"). Bigarren adibidean, aldiz, JSONez adierazitako anotazioak ikus daitezke. Kasu honetan, aingura eta informazio linguistikoa, biak anotazioari dagokion elementu beraren barruan daude txertatuta. Hain desberdinak izanagatik ere, bi anotazioek barruan daukaten informazioa bera da. Anotazioak baliokideak dira. Hala ere, bietako bat erabiltzeko prestatuta dagoen tresnak ez du bigarrena erabiltzeko gaitasunik izango, berariaz horretarako inplementatu ez bada behintzat.

4.2. Elkarreragingarritasunaren bila

Jarraian, elkarreragingarritasunaren arazoari aurre egiteko diseinatu dugun eredua azalduko dugu. Lehenik eta behin, RDF eta OWL lengoaiak zer eta zertarako diren eta nola erabiltzen diren deskribatuko dugu. Ondoren, HPen arteko elkarreragingarritasuna lortzen laguntzeko diseinatu dugun anotazio--ereduen abstrakzioa aurkeztuko dugu, eta arazoari aurre egiten nola lagun dezakeen. Azkenik, tesi-lan honetan aurkeztu ditugun AWA eta NAF anotazio-eskemak eredu abstraktu horren arabera nola egokitu ditugun azaldu, eta planteatzen dugun teknikak bi eskemen arteko elkarreragingarritasuna lortzeko bidean zein ekarpen egiten duen arrazoituko dugu.

```
<!-- Anotazio-eskema: NAF -->
<!-- Hitza: wounded -->
<term id="t23" type="open" lemma="wound" pos="V" morphofeat="VBD">
  ...
</term>
```

```
<!-- Anotazio-eskema: AWA -->
<!-- Hitza: laguntzen -->
<fs type="analisia">
  <f name="forma"><str>laguntzen</str></f>
  <f name="goimailako-ezaugarriak">
    <fs type="goimailako-ezaugarri-lista">
      <f name="KAT"><sym value="ADI"/></f>
      <f name="AZP"><sym value="SIN"/></f>
      <f name="ADM"><sym value="ADOIN"/></f>
      <f name="ASP"><sym value="EZBU"/></f>
      <f name="ADOIN"><str>lagun</str></f>
      <f name="FSL" org="list"><sym value="@-JADNAG"/></f>
    </fs>
  </f>
  <f name="osagaiak" org="list">
    <fs type="osagaia">
      <f name="oina">
        <fs type="lema">
          <f name="sarrera"><str>lagundu</str></f>
          <f name="ezaugarriak">...</f>
        </fs>
      </f>
      <f name="morfemak">
        <fs type="morfema">
          <f name="sarrera">tzen</f>
          <f name="ezaugarriak">...</f>
        </fs>
      </f>
    </fs>
  </f>
</fs>
```

4.1 irudia: Elkarreragingarritasun estrukturalaren arazoa ilustratzeko adibidea. Informazio morfologikoa adierazteko modua oso desberdina da NAFen eta AWAn. NAFen oinarrizko informazio morfologikoa ematen da: lema (`lemma`), kategoria (`pos`) eta bestelakoa (`morphofeat`). AWAren kasuan, aldiz, hitzaren informazio morfologiko aberatsaz gain, oinaren eta morfemena ere ematen da.

```
<!-- A anotazio-eskema -->
<token id="t1" forma="Lazkao" ... />
<token id="t2" forma="Txiki" ... />

<entitate id="e1" aingura="ref:a1" linginfo="ref:li1" />
<aingura id="a1">
  <tokenak>
    <token refid="t1" /> <!-- Lazkao -->
    <token refid="t2" /> <!-- Txiki -->
  </tokenak>
</aingura>
<linginfo id="li1" mota="IZEN_BEREZIA" />
```

```
<!-- B anotazio-eskema -->
{ tok: { id: "t1", text="Lazkao" ... } }
{ tok: { id: "t2", text="Txiki" ... } }

{
  ent: {
    id: "e1",
    span: ["t1", "t2"],
    type: "IZEN_BEREZIA"
  }
}
```

4.2 irudia: Irudiko bi anotazioek, informazio berdina ematen duten arren, itxura desberdina dute, bai egitura aldetik, bai kontzeptuen izendatze aldetik, eta bai erabilitako adierazpide fisikoaren aldetik (XML eta JSON).

4.2.1. RDF eta OWL

RDF[1] (*Resource Description Framework*) datuen deskribapenerako lengoaia bat da. Webeko informazioa aplikazioen artean elkarbanatzeko helburuarekin sortu zen. Datuen benetako semantika zein den deskribatzea ahalbidetzen du, pertsona edo aplikazio desberdinek datu horiek behar bezala erabil ditzaten. Web semantikoaren (Berners-Lee *et al.*, 2001) oinarrizko tresnatzat hartzen da.

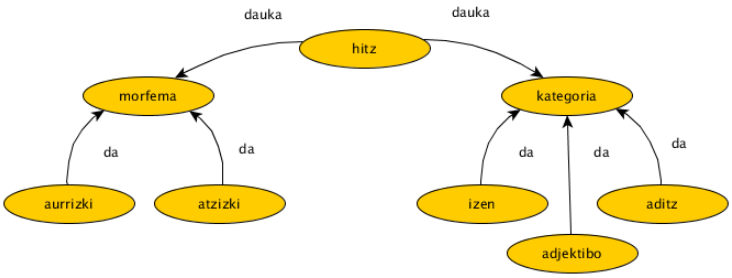
RDFren oinarrian hirukoteak daude. Izan ere, informazio guztia hirukoteen bitartez definitzen da. Adibidez, *jolastu* hitza aditza dela, *jolastu kategoriaDauka aditza* moduko hirukote batekin adieraz daiteke, eta 4.3 irudian bezala irudikatuko genuke.

4.3 irudia: RDF hirukote bat.

Ikus daitekeenez, lehenbiziko eta hirugarren elementuak adabegiak dira, eta bigarrena beste biak lotzen dituen ertza. Zehatzago esanda, lehenbiziko osagaiari subjektua deitzen zaio, bigarrenari predikatua eta hirugarrenari objektua. Normalean, subjektua eta objektua kontzeptuak izaten dira, eta predikatua, aldiz, ezaugarri bat. Horrela, hirukote bat baino gehiago elkar ditzakegu, subjektu edo objektu bakoitza hainbat ezaugarrirekin lotuta egon baitaiteke, grafo-egitura bat osatuz. Arlo jakin baten inguruko kontzeptuekin RDF hirukote egokiak definituz, arlo bakoitzaren semantika deskribatzen duten ontologia deritzen grafoak eraiki daitezke. 4.4 irudian ikus daiteke RDF ontologia sinple baten irudia.

RDFren ezaugarri aipagarrienetako bat haren grafo-izaera da. Bi arrazoi nagusi daude izaera horren atzean, RDF bera hobeto ulertzen laguntzen dutenak. Arrazoi nagusietako bat da RDF ez dela, adibidez XML den bezala, dokumentuen egitura deskribatzeko lengoaia bat, kontzeptuen deskribapen semantikoa egiteko lengoaia baizik; eta egitura zehazteko zuhaitz-egitura

[1] `https://www.w3.org/RDF` (kontsulta: 2017-05-08)



4.4 irudia: RDF hirukotez osatutako ontologia oso sinplea. Ontologia horren arabera, hitz batek kategoriak eta morfemak izan ditzake. Domeinua kontuan hartuta, badakigu hitzek kategoria bakarra izaten dutela, baina RDF hutsez ezin da horrelako kardinalitaterik zehaztu.

egokia den bezala, grafo-egitura egokiagoa da datuen semantika deskribatzeko. Beste arrazoia da RDF webeko baliabideak deskribatzeko helburuarekin sortu zutela, eta webaren izaera heterogeneoa dela-eta, ohikoa da bertako baliabideak modu deszentralizatuan agertzea. Horregatik, errazagoa eta naturalagoa da bi grafo elkartzea, bi zuhaitz elkartzea baino.

Hala ere, ontologia sinpleak eraikitzeko RDF egokia den arren, zenbait kasutarako ez da nahikoa. Batzuetan, ontologiak behar bezala eraikitzeko beharrezkoa izaten da logika formalean oinarritzen diren lengoaiak erabiltzea. Azken urteetan gehien zabaldu den mota horretako lengoaia OWL (Bechhofer, 2009) da, ontologiak eraikitzeko lengoaia ahaltsua.

OWL (*Web Ontology Language*) ontologiak modelatzeko W3C erakundeak proposatutako estandar bat da, deskribapen-logiketan oinarritua. Deskribapen-logikaren helburuak bi dira, oro har: arloen ezagutza adierazteko mekanismo formalak eskaintzea, batetik, eta ezagutza horren gainean arrazoitzeko gaitasuna ematea, bestetik. OWL dokumentuak RDF dokumentuak ere badira, RDFren gainean eraikitako lengoaia baita OWL.

Ontologiak eraikitzeko klase, instantzia eta ezaugarri kontzeptuak eskaintzen ditu OWLek. Instantziak objektuak diren bezala, klaseak instantzia multzoak dira. Horrela, instantzia bakoitza klase bati edo gehiagori dagokio. Klaseen arteko hierarkiak ere sor daitezke, hau da, klase bat beste baten az-

piklase izan daiteke, eta, kasu horretan, bere ezaugarri guztiak jasoko ditu. Azkenik, ezaugarriak klase bateko objektuek beste objektuekiko edo balio atomikoekiko izan ditzaketen erlazioak dira.

OWLek hiru azpilengoaia ditu, adierazgarritasun desberdinekoak. Adierazgarrienek ezagutza xehetasun handiagoz adierazteko ahalmena dute, baina arrazoitzeko behar duten denboraren eta konputazio-kargaren aldetik astunak izan daitezke, edo baita bideraezinak ere:

- OWL Lite: OWL lengoaiarik sinpleena da. Hierarkiak adierazteko eta haien arteko erlazio bakunak adierazteko gaitasuna du. Oinarrizko kardinalitatea adierazteko ere balio du. Ordainetan, arrazoitzeko konputazio-karga arinagoak eskatzen ditu.
- OWL DL: OWL Lite baino adierazgarriagoa da, eta arrazoitzeko erabakigarritasuna bermatzen du, hau da, arrazoiketa denbora-tarte finitu batean burutuko dela ziurtatzen du, nahiz eta konputazio-karga astunak behar izan ditzakeen.
- OWL Full: RDFren adierazteko mekanismo guztiak biltzen ditu, OWL familiako lengoaia adierazgarriena izanik. Ordainetan, ez du erabateko arrazoitzeko gaitasunik eskaintzen. Hau da, kasuaren arabera, baliteke arrazoiketa bideraezina izatea eta inoiz ez bukatzea.

Jarraian, RDFren muga aipagarri batzuk aipatuko ditugu, OWLen bidez modu naturalean ebatz daitezkeenak:

- RDFren ezagutza adierazteko mekanismoa oso mugatua da datuen gainean arrazonamendu automatikoa egiteko. RDFS (Brickley eta Guha, 2014) hedapenak gaitasun handiagoa gehitzen dio arrazonamendurako, baina OWL, horretara bideratuta egonik, ezagutza jakin batetik ezagutza eta ondorio berriak eraузteko lengoaia askoz ere ahaltsuagoa da.
- RDFren kardinalitatea adierazteko gaitasunik eza. Esate baterako, RDFn ezinezkoa da definitzea gure ontologian hitz batek maila jakin

bateko analisirik ez izatea onargarria den, edo, era berean, analisi bat baino gehiago edukitzea ere onargarria den.

- Ez dago ukapen bidezko espresiorik. Subjektu baten objektu bezala zein motatako objektuak onartzen diren zehatz daiteke, baina ez zein motatakoak onartzen ez diren.
- Ezin dira klase multzoak osatu. Ezin da adierazi subjektu edo objektu baten mota klase multzo bateko objektuek osatzen dutela, klase bakar batekoek baizik.

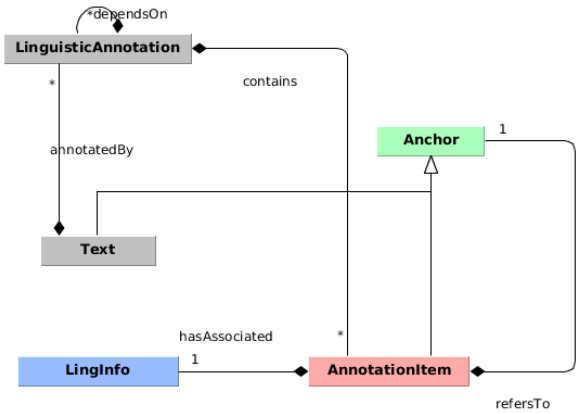
4.2.2. Anotazio-eskemen abstrakzioa

Anotazio-eskemen artean bihurketak egiteko aukera izanez gero, HPen integrazioa egin ahal izango genuke. Arazoa bi eskemaren arteko bihurketa egitearen konplexutasuna da, eskemak egitura konplexuak izaten baitira. Gure ondorioetako bat izan da bihurketa-prozesua errazteko, eta prozesua benetan egingarri izan dadin, oso lagungarri suerta daitekeela anotazio-eredu abstraktu bat diseinatzea, eskema guztiek oinarri komun bat izan dezaten. Horrela, bihurketa egin ahal izan aurretik, parte hartuko duten eskemak eredu abstraktuaren arabera egokituko genituzke. Behin hori burututa, eskemek oinarrizko egitura komuna izango lukete, mapaketa erraztuz.

Ezinbestekoa da eredu abstraktu egokia aukeratzea. Edozein eskemarentzat balio behar duenez, oso orokorra izan behar du, eskema guztietara egokituko dena. AWA bera antzeko ideiak buruan izanik garatu genuenez, bere oinarria oso egokia ikusi dugu eredu abstraktu bilakatzeko. Hortaz, informazio linguistikoan beti agertzen diren hiru elementu mota nagusi berreskuratu ditugu AWAtik: aingurak, informazio linguistikoa eta biak lotzen dituzten estekak.

Hiru elementu horiekin edozein anotazio-eredu abstraitu daitekeela uste dugu. Horregatik, bi datu-eredu elkarren artean elkarreragingarri egiteko, abiapuntu bezala, biak eredu abstraktu honi jarraituz formalizatzea proposatzen dugu. 4.5 irudian ikus daiteke eredu abstraktuaren eskema.

Aingurei dagokienez, hasiera batean anotazio bakoitza testu zati jakin bati dagokiola pentsa daitekeen arren, ez da beti zehazki horrela gertatzen. Asko-

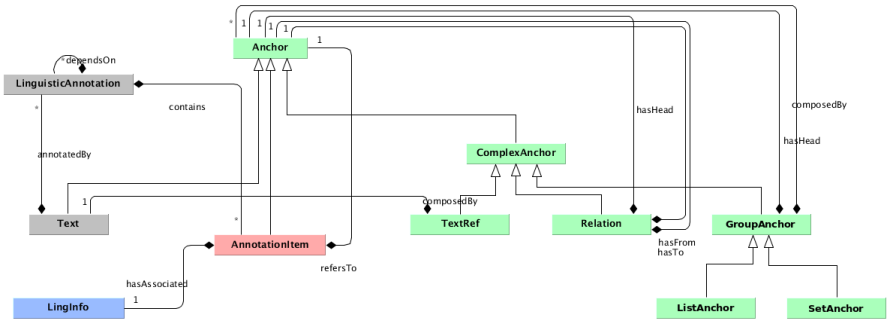


4.5 irudia: Eredu abstraktuaren irudikapena. Anotazioek aingura bat eta informazio linguistikoa daukate lotuta, eta anotazio bakoitzak, era berean, aingura izaera ere badauka, hurrengo mailako anotazioen aingura izan baitaiteke.

tan, anotazio bat beste anotazio bati dagokio, eta ematen duen informazioa, testu zatiari buruzkoa baino gehiago, beste anotazio bati buruzkoa izaten da. Hori dela eta, anotazio linguistikoak geruzatan multzoka daitezke, non geruza bateko anotazioek beste geruza batekoen gaineko informazioa ematen duten (ikus 3. kapitulua). Hori ikusita errazago ulertzen da aingurak eta haiei dagokien informazio linguistikoa bereiztearen arrazoia.

Eredua gehiago ere heda daiteke, ainguren atala batez ere. Izan ere, ainguren artean patroi batzuk identifikatu ditugu, anotazio bakar bat izatetik haratago doazenak. Zuzenean testuaren gainean sortutako anotazioen kasuan, esaterako, aingura testuko hitzez osatuta egongo da. Aingura anotazioz osatzen denean ere, beti ez dira anotazio bakunak izaten. Gure eredu abstraktuan aingura konplexuen kontzeptua gehitu dugu, horrelako aingura egituratuak biltzen dituena. Hau da, aingura bakuna anotazio bakar batez osatuko da, eta bestelako egitura guztiak aingura konplexutzat hartuko dira. Aingura konplexuak eredu abstraktuan integratu ondoren, eredua 4.6 irudian ikus daitekeen bezala gelditu da.

Jarraian, identifikatutako aingura motak zerrendatu ditugu. Aingura motak



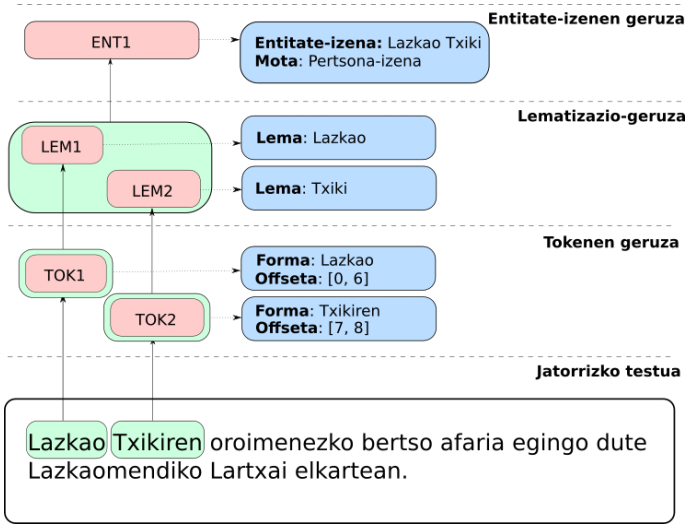
4.6 irudia: Eredu abstraktua, osorik.

grafikoki irudikatzeko, AWAren kapituluko (3.1) irudi bat berreskuratu dugu (4.7 irudia, lehengo 3.2). Bertan, esaldi baten gaineko hainbat anotazio linguistiko ageri dira, hiru motatakoak: tokenak, lemak eta entitateak. Hona hemen, bada, aingura motak:

- Testu-aingurak: Anotazio hauek zuzenean testuko hitzen gainean sortzen dira. Kasu honetan, aingura hitz zerrenda batez osatzen da, eta bi modu bereizi ditugu hitz zerrenda bat adierazteko: offset- eta luzera-atributuen bidezkoa, eta, testua XMLz kodetuta dagoen kasuetarako, XPointer bidezkoa. Irudian ikusten den bezala, tokenek dauzkate mota honetako aingurak, izan ere, tokenek testuko hitzei egiten baitiete erreferentzia. Mota horretako aingurak, eredu abstraktuan, `TextRef` elementuekin adierazten dira (ikus 4.6 irudia).

- Anotazio bakarra: Anotazioa aurreko maila bateko anotazio baten gainean definitzen da, eta aingura anotazio bakar batez osatuta egongo da. Irudiko lemen kasuan gertatzen da hau, lema bakoitzak token bakarrari egiten baitio erreferentzia. Eredu abstraktuaren arabera, anotazioa bera, hau da, `AnnotationItem` elementua, litzateke aingura.

- Anotazio zerrenda: Aingura anotazio multzo batek osatzen du. Anotazioen arteko ordena esanguratsua da. Eredu abstraktuan `ListAnchor` izenarekin gehitu dugu. Irudiko adibidean, entitate-anotazioaren aingura bi lemak osatzen dute.



4.7 irudia: Anotazio linguistikoen itxura orokorra esaldi oso baten gainean (anotazioak gorriz, aingurak berdez eta anotazioen informazio linguistikoa urdinez).

- Anotazio multzoa: Aingura anotazio multzo batek osatzen du. Anotazioen arteko ordena ez da esanguratsua. Eredu abstraktuan `SetAnchor` izenarekin gehitu dugu.

- Erlazio-aingura: Bi anotazioren arteko erlazio batek osatzen du aingura. Erlazioak norabide jakin bat dauka, eta bi anotazioetako batetik bestera doa, beraz. Aingura mota hau ezinbestekoa da zuhaitz-egiturak eta dependentzia-egiturak modelatzeko. Aingura horiek, eredu abstraktuan, `Relation` aingurekin adierazten dira.

4.2.3. AWA eta NAF anotazio-eskemak eredu abstraktuaren arabera mapatzen

Aurkeztu dugun eredu abstraktuak anotazio-eskemen arteko zubi-lanak egitea du helburu. Hau da, eskema desberdinek egitura oso desberdina dutenean zaila da horietako bat erabiltzeko inplementatutako HP batek bestea ere erabili ahal izatea. Bi eskemak egitura abstraktu komun batera egokitzen

baditugu bidearen zati handi bat eginda legoke, goi mailako egitura komuna izanik, mapaketa egitea errazagoa bailitzateke, eta hori da tesi-lan honetan landu duguna.

Hori erakusteko, anotazio-eskema bat eredu abstraktuarekin nola mapatu azalduko dugu lehenik. Ondoren, dagoeneko aurkeztu ditugun AWA eta NAF ereduak gure eredu abstraktura egokituko ditugu, eta, azkenik, hori egitearen abantailen inguruko hausnarketa egingo dugu.

4.2.3.1. Anotazio-eskemak eredu abstraktuaren arabera egokitzeko pausoak

AWAren kasuan, eredu abstraktua eraikitzeko oinarriak bertatik erauzi ditugunez, erraza da eskema eredu abstraktuarekin bat etor dadin egokitzea. Elkarreragingarritasunaren helburua, ordea, edozein anotazio-eskemaren arteko mapaketa egitea denez, edozein eskema eredu abstraktuarekin mapatzeko beharra izango dugu. Hori nekezagoa izan daiteke, edozein eskema ez baita aingura, informazio linguistiko eta esteken ereduari jarraituz eraiki, baina, hain justu, eredu abstraktu hori diseinatzearen arrazoia hiru kontzeptu horien orokortasuna izan zen. Hau da, uste dugu anotazio-eskema guztiek dauzkatela, inplizituki bada ere, hiru elementu horiek. Eskema berri bat gure eredu abstraktura egokitzeko lehenbiziko lana, hortaz, eskema horretan oinarrizko hiru elementu horiek identifikatzea da. Horretarako, pausoak ordena honetan jarraitzea da gure gomendioa:

1. Anotazio motak identifikatu: Pauso honetan, eskemak definitzen dituen anotazio mota guztiak identifikatuko ditugu. Normalean, maila linguistiko bakoitzeko mota bat edo gutxi batzuk egon ohi dira. Anotazio motak identifikatzea berehalakoa izaten da, anotazioak motaka sailkatuta egoten baitira eskema gehienetan.

2. Aingurak identifikatu: Anotazio mota bakoitzaren kasua aztertuz, identifikatu behar da bere edukiaren zein zatik adierazten duen anotazioa bera zein elementuren gainean eraiki den. Adibidez, tokenen kasuan ohikoa da atributuek anotazioa testuaren zein zatiri dagokion offseten

bidez adieraztea. Beste askotan, anotazioa beste maila bateko anotazioren baten gainean egoten da definituta. Kasu horietan, jatorrizko anotazioaren erreferentzia zein atributuk adierazten duen identifikatu beharko litzateke.

3. Informazio linguistikoa identifikatu: Aingurari buruz anotazioak ematen duen informazio linguistiko guztia sartzen da multzo honetan.

4.2.3.2. AWA eredu abstraktuaren arabera egokitzen

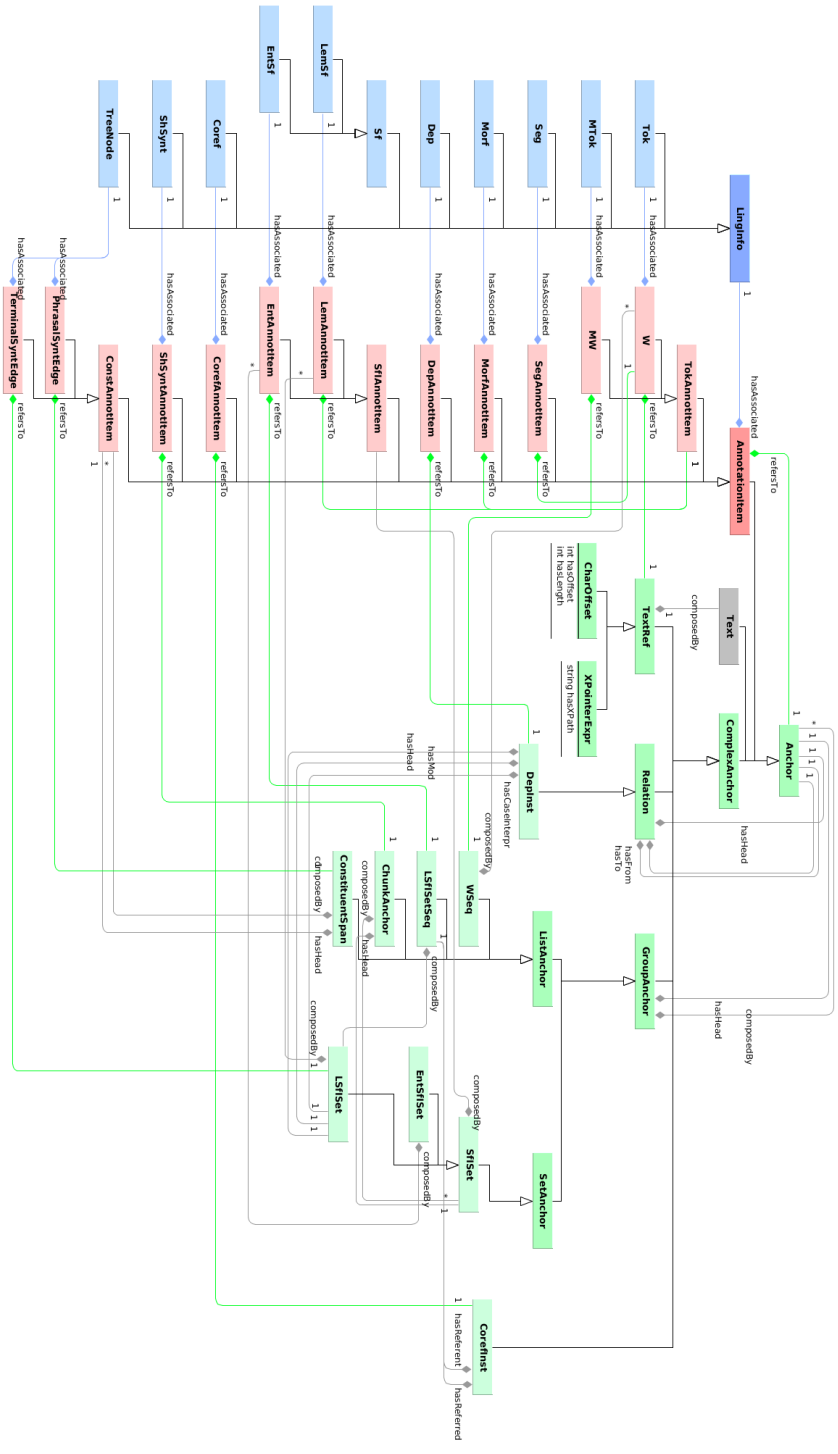
Funtsean, bai AWA eta bai eredu abstraktua anotazioen aingura, informazio linguistiko (`LingInfo`) eta esteken kontzeptuak kontuan hartuz eraiki ditugu. Laburbilduz[2], anotazio guztiak aipatutako hiru elementuez osatuta daude. Anotazioak aingura baten gainean definituta daude, aingura edozein testu zati edo beste edozein anotazioren konbinazio izan daitekeelarik, eta maila linguistiko jakin bateko informazio linguistikoa ematen dute. Azkenik, esteka bakoitzak aingura bat informazio linguistiko jakin batekin lotzen du, anotazioa osatuz.

Horrela, bada, AWAko anotazio mota guztiak, dagoeneko, egitura hori jarraituz definitu dira. Beraz, egokitzapena berehalakoa da. 4.8 irudian ikus daiteke ereduaren eskema osoa. Ikus daitekeen bezala, eredua aingura, informazio linguistiko eta esteketan sailkatuta dago, eta horregatik AWAko klase guztiek hiru horietako baten azpiklase dira.

Har dezagun morfosintaxi-mailako anotazioen kasua adibide bezala. Esteka, edo anotazioa, `MorfAnnotItem` klasea da, eta hau `AnnotItem` klase abstraktuaren azpiklasea da. Klase hori beste birekin lotuta dago: aingurarekin (`refersTo` erlazioaren bidez) eta informazio linguistikoarekin (`hasAssociated` erlazioaren bidez). Ikus daitekeenez, aingura `TokAnnotItem` klasea da; horrek esan nahi du morfosintaxi-mailako anotazioak tokenen gainean definitzen direla. Bestalde, `Morf` motako objektuek anotazioen informazio linguistikoa biltzen dute. Informazio horren egitura ez da irudian ikusten, konplexuegia baita irudian sartzeko.

Ikus dezagun beste adibide bat: HAULen kasua. HAUL anotazioen klasea

[2]Informazio gehiago AWAren atalean (3.1)



4.8 irudia: AWA anotazio-eskema eredu abstraktura egokituta

`MW` da AWAn (ingelesezko *multiword* terminotik dator). Gainera, klase hori `TokAnnotItem`-en azpiklasea da, AWAn HAULak token mota berezi bat bezala hartzen baitira. HAULen informazio linguistikoa `MTok` klasean definitzen da. Kasu honetan ere, informazio linguistikoa kontsultatu ahal izateko RNG eskemetara jo beharko litzateke (ikus 3.1.1.1 atala). Azkenik, aingura `WSeq` motakoa da, tokenen sekuentzia, alegia. Ikus daiteke nola aingura hori eredu abstraktuko `ListAnchor` aingura-klasearen azpiklasea den, izan ere, `ListAnchor` klaseak edozein aingura sekuentzia adierazten baitu, eta, kasu honetan, sekuentzia hori tokenen sekuentzia bat dela zehazten da.

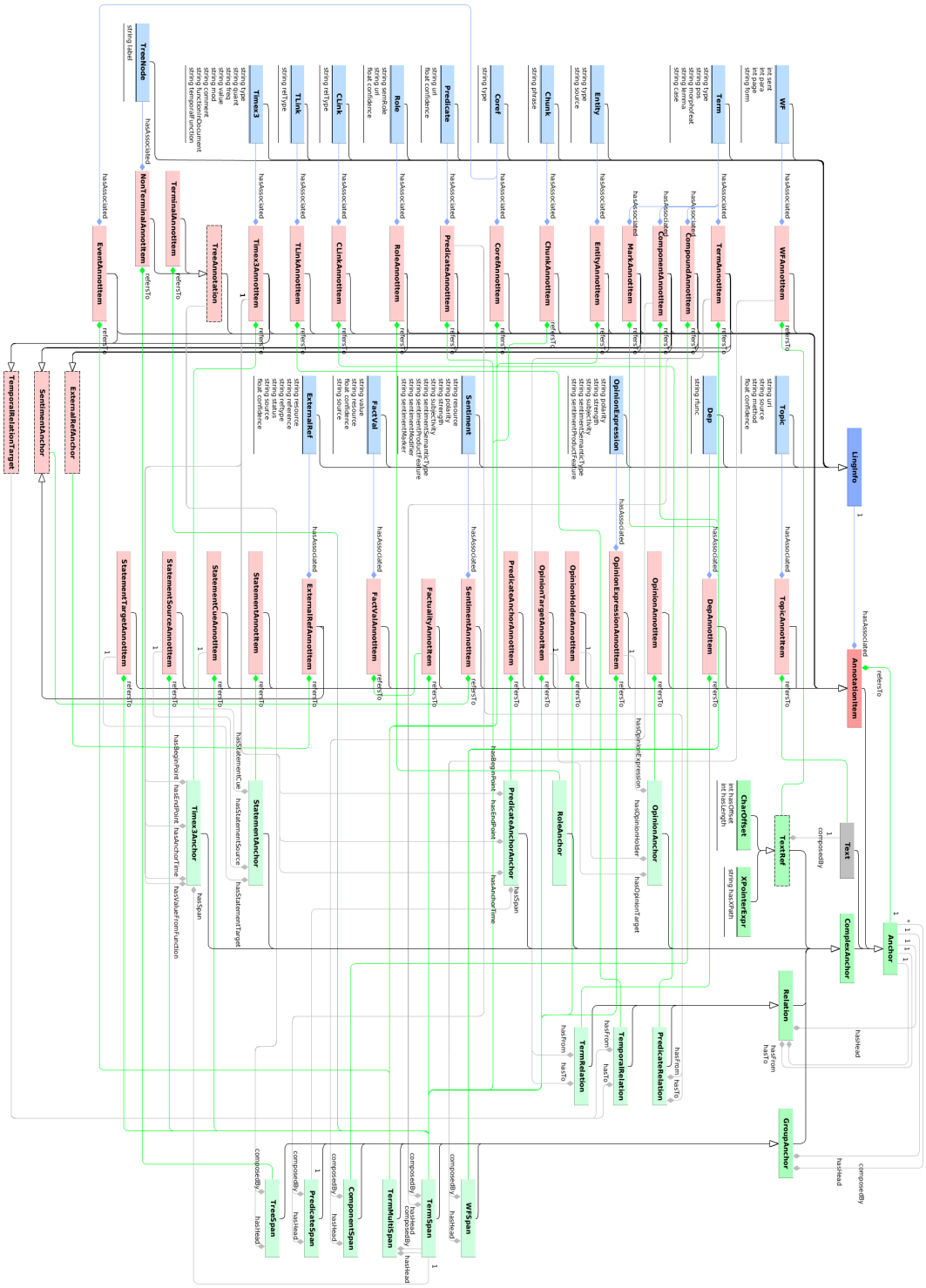
4.2.3.3. NAF eredu-abstraktuaren arabera egokitzen

NAF anotazio-eskema ere eredu abstraktuaren arabera egokitu dugu. NAF ez zen, AWA bezala, aingura, informazio linguistiko eta esteken ereduari jarraituz eraiki. Horregatik, lehenbiziko pausoa NAF gure eredu abstraktura egokitzea da, anotazio motak, aingurak eta informazio linguistikoak identifikatuz eta sailkatuz.

4.9 irudian ikus daiteke NAF eskemaren diagrama eredu abstraktuaren elementuak (anotazioak, aingurak eta informazio linguistikoa) identifikatu eta gero. Anotazio mota asko daudenez NAFen, adibide batzuk aztertuko ditugu jarraian:

WF NAFen *Word form* (WF) bezala ezagutzen dira tokenak. `WFAnnotItem` deitzen diegu WF anotazioei. WFen aingurak bi motatakoak izan daitezke NAFen: *offset* eta *length* atributuekin testuko hasiera-posizio bat eta testu zatiaren luzera defini daitezke, edota XPath espresio batekin aukera daiteke testu zatia bera. Horregatik, diagraman, gure eredu abstraktuaren parte den `TextRef` klasea aukeratu dugu WFen aingura gisa, klase hori `CharOffset` (*offset* eta *length* atributuak) eta `XPointerExpr` (*XPointer* espresioa) klaseen abstrakzioa baita. Behin aingura definituta, NAFeko WF anotazioen atributu guztiak, `offset`, `length` eta `xpath` izan ezik, informazio linguistikoaren parte izango dira: `sent`, `para`, `page` eta `form` atributuak, alegia.

DEP Dependentzia sintaktikoen anotazioak, NAFen, DEP terminoarekin



4.9 irudia: NAF anotazio-eskema eredu abstraktura egokituta

izendatzen dira. DEPek hiru atributu dituzte: `from`, dependentziaren jatorri-terminoa adierazten duen atributua; `to`, dependentziaren helburu-terminoa; eta `rfunc`, dependentziaren erlazio-funtzioa. Aingura identifikatzeko, lehenik, DEP anotazioa zein elementuren gainean definitzen den ulertu behar da. Kasu honetan, bi terminoren arteko erlazioaren gaineko informazioa ematen du anotazioak, eta beraz, `from` eta `to` terminoen arteko erlazioa da DEPen aingura. Eredu abstraktuan badaukagu aingura mota bat erlazioak definitzeko, eta, kasu honetan erlazioa termino-anotazioen artekoa denez, `TermRelation` izeneko azpiklase berri bat sortu dugu, `from` eta `to` termino motako anotazioak izatera behartuz. Azkenik, `rfunc` atributua izango da DEP anotazioen informazio linguistikoa osatzen duena.

Chunk Chunkak azaleko sintaxia egitean identifikatutako zatiak dira. Chunken osagaiak bi dira: Zatia osatzen duten terminoen zerrenda, eta `phrase`, sintagma mota adierazten duen atributua. Kasu honetan, erraza da ikustea anotazioa terminoen zerrenda baten gainean eraikitzen dela. Beraz, aingura termino zerrenda bat izango da. Horrelakoetarako ere badaukagu eredu abstraktuan klase bat erazagututa: `Span`. Hala ere, `Span`-en elementu guztiak termino motakoak direla zehazteko, `TermSpan` izeneko azpiklase bat sortu dugu. Azkenik, terminoaren informazio linguistikoa `phrase` atributuak osatuko du.

Opinion Iritzi-anotazioak, NAFen, `Opinion` elementuen bitartez adierazten dira. Hiru osagai dituzte iritzi-anotazioek: `holder`, iritzia ematen duen aktorea; `target`, iritzia zeri buruzkoa den; eta `expression`, iritzia bera. Gertatzen dena da hirurak ere, NAFen, anotazio osoak direla, beste mota batzuetako anotazioak. Beraz, hiru anotazio horiek iritzi-anotazioen aingura osatzen dute, horien gainean eraikitzen baitira iritzi-anotazioak. Eredu abstraktuko `ComplexAnchor` klasetik eratorri dugu klase berri hau, eta `OpinionAnchor` deitu diogu. Hiru anotazioz osatzen da aingura: `holder`, `target` eta `expression` banaz. Iritzi-anotazioen beste berezitasun bat da ez dutela informazio linguistiko propiorik, informazioa aipatu berri ditugun eta aingura osatzen duten elementuek ematen baitute.

4.2.3.4. AWA eta NAFen arteko elkarreragingarritasunaren bidean

Kapitulu honetan ikusitakoarekin, elkarreragingarritasunaren arazoari aurre egiteko lehen urratsa egin dugu. Horretarako, anotazio-eredu abstraktu bat diseinatu dugu, edozein anotazio-eskemak izan ditzakeen elementu nagusiez osatutakoa. Eskema desberdinak erabiltzen dituzten tresnen arteko elkarreragingarritasuna lortzeko, gure proposamena da, lehenbizi, 4.2.3.1 ataleko pausoak jarraituz eskemak eredu abstraktuarekiko mapatzea. Horrela, bi eskemaren arteko mapaketak egiteko bidearen zati bat eginda legoke.

Aurreko ataletan bi anotazio-eskema, AWA eta NAF, eredu abstraktu beraren azpian berregituratu ditugu, anotazio mota bakoitzaren aingurak eta informazio linguistikoak argi bereiziz. Hori horrela, eredu abstraktua interpretatzen dakien edozein programak, AWAren edo NAFen elementu guztiei buruzko semantikaren zati bat behintzat jakin badaki. Hau da, eredu abstraktua ulertuta soilik, eta honen arabera egokitutako eskema baten anotazioak izanda, jakin daiteke anotazioen aingura nola osatzen den, baita informazio linguistikoa ere.

HPen arteko elkarreragingarritasuna lortzeko bidean pauso garrantzitsua da hori, erabiltzailearentzat ahalik eta generikoena eta transparenteena den soluzio bat lortzeko beharrezkoa baita anotazio-eskemen elementu bakoitza zertan datzan eta bere egitura zein den ahalik eta hobekien ezagutzea.

Egoera horretan, AWAren eta NAFen osagaiak elkarri mapatzea besterik ez da falta. Eskemak eredu abstraktuaren arabera berrantolatzeak mapaketa hori gauzatzen laguntzen du. Behin elementu guztiak (aingurak, estekak eta informazio linguistikoa) mapatutakoan, tarteko geruza berri bat sartu ahal izango genuke tresnen eta anotazioen artean, eta geruza horri esker lor daiteke tresnentzat gardena izatea interpretatu edo sortu behar dituzten anotazioen adierazpidea, elkarreragingarritasunaren arazoari soluzioa emanez.

Tarteko geruza horren nondik norakoak tesi-lan honen irismenetik kanpo gelditu dira. Edonola ere, lan interesgarria iruditzen zaigu etorkizunean heltzeko, horrekin kapitulu honetan landutakoei behar bezalako amaiera bat emango bailitzaieke.

III Atala

TRESNEN PROZESU MAILAKO INTEGRAZIOA

5. KAPITULUA

Datu handien teknikak hizkuntzaren prozesamenduan: arloaren egungo egoera

Kapitulu honetan, datu handiak prozesatzeko sistema ezagunenak azalduko ditugu lehenik. Horien artean, gure arkitekturarako aukeratu dugun MongoDB datu-baseekin hasiko gara, eta prozesaketa banaturako sistemekin jarraituko dugu. Ondoren, hizkuntzaren prozesamenduaren arloan prozesaketa banatuaren inguruan egin diren lanak bildu eta gainetik azalduko ditugu. Horien aurretik, irakurlea errazago koka dadin, ohikoenak diren prozesaketa-ereduak aurkeztetik hasiko gara.

5.1. Prozesaketa-ereduak

Datu handien prozesaketan, bai hizkuntzaren prozesamenduaren arloan eta bai beste arlo guztietan, bi prozesaketa modu dira ezagunenak: batch-prozesaketa eta streaming-prozesaketa.

Batch-prozesaketan datu multzo bat dago prozesatzeko zain, eta datu multzo osoa prozesatu arte emaitzak ez daude eskuragarri. Prozesaketa osoak hasiera eta bukaera bakarra dauka. Helburua datu guztiak lehenbailehen prozesatzea denez, prozesaketa-eredua sinpleagoa da, datu-unitate bakoitza prozesatzen igarotako denborak ez baitauka berebiziko garrantzirik. Horregatik, ingurune banatu batean, datu-unitateak paralelizatu egiten dira nodoen

artean.

Batch eredua gure arlora ekarriz, demagun datu-unitate bakoitza testu-dokumentu bat dela. Dokumentu multzo baten gainean kontsultak egiteko beharrezkoa da dokumentu guztiak prozesatuta izatea. Kasu horretan, ingurune banatu bat izanda, dokumentu bakoitza nodo batera bidaliko genuke, dokumentuak paraleloan prozesatuz. Eredu hau egokia da corpus osoak prozesatzeko.

Streaming-prozesaketan, aldiz, prozesua etengabe dago martxan (Shahrivari, 2014). Normalean ez da datu multzo finko bat prozesatu nahi izaten. Datuak banan-banan edo multzo txikitan doaz sistemara iristen, datu-jario baten forman, eta sistemak iritsi ahala prozesatu behar ditu datuok, emaitzak etengabe eguneratuz. Eredu horretan, garrantzizkoena ez da datu guztiak denbora-tarterik txikienean prozesatzea, datu bakoitza ahalik eta azkarren prozesatzea baizik. Hori dela eta, askotan datu-unitatea bera paralelizatzen da latentzia txikitzeko.

Hizkuntzaren prozesamendura itzuliz, sare sozialen monitarizazioa streaming-kasu baten adibide garbia da. Demagun Twitterreko txio-jario bat prozesatu nahi dugula. Sarrerako datu-unitatea txio bakoitza litzateke, eta txioak prozesatu ahala emaitza eguneratzen joan gaitezke. Kasu horretan, txio bakoitza lehenbailehen prozesatzea da garrantzitsuena.

Azken urteetan arkitektura hibridoak asko erabiltzen dira. Ezagunenetako bat lambda arkitektura da (Marz eta Warren, 2015). Lambda arkitektura datu handiak prozesatzeko arkitektura orokor bat da, eta arazoa hiru geruzatan zatitzen du: abiadura-geruza, zerbitzu-geruza eta batch-geruza. Datu guztiak batch-geruzako biltegian mantentzen dira. Ideia nagusia da kontsultak ez direla behin eta berriz datu guztien gainean exekutatuko. Horretarako, egin nahi izango diren kontsultak identifikatu behar dira lehenik eta behin, eta horien emaitzak kalkulatuko dituzten funtzioak idatzi. Horrela, kontsulta guztiak prekonputatu egingo dira behin eta berriz, zerbitzu-geruza osatuz, eta, kontsulta bat exekutatu behar denean, prekonputatutako datuetatik jasoko dira emaitzak.

Kontsulten emaitzak datu berrienekin eguneratzeko, kontsultak behin eta be-

rriz prekonputatuko dira. Hala ere, iterazio batetik bestera ere orduak pasa daitezke, datu kopurua oso handia izan baitaiteke. Tarte horretan sistemara iritsi diren datuak ere kontsulten emaitzetan islatzeko, sarrerako datu guztiak batch-geruzara bidaltzeaz gain, abiadura-geruzara ere bidaltzen dira. Geruza honetan datu bakoitza berehala prozesatzen da, emaitzak daturik berrienekin eguneratuz. Egoera horretan, kontsulta bat egiten den bakoitzean, datuen zati handiena prekonputatutako zerbitzu-geruzatik jasotzen da, baina daturik berrienak, oraindik prekonputatu gabe egon daitezkeenez, abiadura-geruzatik jasotzen dira. Batch-geruzan, izenak berak dioen bezala, batch-prozesaketa egiten da. Abiadura-geruzan, berriz, streaming-prozesaketa egiten da. Horrela, eredu bakoitzaren alde onei ateratzen die etekina lambda arkitekturak.

5.2. MongoDB datu-baseak

Datu-baseak erabiliak izaten hasi zirenetik, datu-base erlazionalak izan dira gehien erabili direnak. Hala ere, teknologiak aurrera egin ahala, eta digitalizatutako informazio kopurua esponentzialki hazten doan garai hauetan, datu-base erlazionalak, kasu batzuetan, mugatuta gelditzen ari direla ikusi da. Izan ere, datu-base erlazionalak zerbitzari bakarrean biltegiratu eta kudeatzeko diseinatuak izan ziren, eta, gaur egun, datu handien esparruan, sistema banatuak erabiltzea ia ezinbestekoa bihurtu da.

Arazoari aurre egiteko, NoSQL datu-baseak garatu dituzte azken hamarkadan zehar. Datu-base berri horien ezaugarri nagusia da datuak ez direla datu-base erlazionaletan bezain egitura zurrunarekin gordetzen. Egituraren malgutasun horrek, datuen inkontsistentziak agertzeko arriskua handitu arren, erraztu egin du datu-baseen edukia hainbat zerbitzaritan zehar banatzea. Horrela, datu-baseak ere eskalagarri bihurtu dira.

NoSQL motako DBKSen artean erabiliena, DB-Engines[1] webgunearen arabera, MongoDB da 2016ko abenduan, Cassandra[2] eta Redis-en[3] aurretik.

[1] http://db-engines.com/en/ranking (kontsulta: 2017-05-08)
[2] https://cassandra.apache.org (kontsulta: 2017-05-08)
[3] https://redis.io (kontsulta: 2017-05-08)

NoSQL motako beste DBKS ezagunenetako batzuk HBase[4], Memcached[5] eta CouchDB[6] dira.

Gure arkitektura banatuan erabiltzeko, NoSQL datu-base sistema eskalagarri bat behar genuen. Denen artean, MongoDB aukeratu dugu. Testu-dokumentu anotatuak gordetzeko egiturarik egokiena MongoDBk datuak datu-basean gordetzeko erabiltzen duena zela erabaki genuen. Izan ere, NAF anotazioak egitura hierarkikoak dira, eta JSON dokumentuak egokiak dira hierarkiak adierazteko. Gainera, anotazio linguistikoak oso desberdinak izan daitezke elkarren artean, eta ez balioei dagokienez bakarrik, baita egitura aldetik ere, eta horretarako egokiagoa da MongoDBrena bezalako eredu malgu bat, datu-base erlazionalena bezalako eredu zurrun bat baino. Izan ere, datu-base erlazionaletan, datu-mota guztiek egitura finko bat bete behar dute, eta gordeko diren datuen izaera aldakorra bada, balio hutsekin jokatu behar da datuak ereduari egokitzeko. Aldiz, MongoDB datu-baseetan, datu bakoitzak bere egitura propioa dauka, besteekiko independentea.

MongoDB datu-baseak dokumentuz osatzen dira. Dokumentuek BSON formatua jarraitzen dute. BSON JSON ereduaren hedapen bat da, datu-mota berri batzuk gehituta eta lengoaia desberdinekin kodetu eta deskodetzea eraginkorrago egiteko moldaketa batzuekin. Hortaz, datuek BSON eredua jarraitu behar dute datu-basean sartzeko. 5.1 irudian ikus daiteke BSON dokumentu baten adibidea. Ikus daitekeen bezala, BSON dokumentuetan egitura hierarkikoa adieraz daiteke.

Bestalde, MongoDB datu-baseetako dokumentuak bildumatan sailkatzen dira. Dokumentu bakoitza, beraz, datu-baseko bilduma batean sartuko da. Bildumak, besterik gabe, izaera bereko dokumentuak bilduz, datu-basearen antolaketa logikoago egiteko tresna dira. Bildumetan datuen gaineko indizeak ere defini daitezke, bilaketak azkarrago egiteko.

MongoDBko dokumentu bateko atributu baten balioa, bakuna izan ordez egituratua denean, bere osotasunean, beste dokumentu bat bezala ikus daiteke. Hau da, hierarkia osoa dokumentu bakarrean adieraz daiteke, edo hierar-

[4]https://hbase.apache.org (kontsulta: 2017-05-08)
[5]https://www.memcached.org (kontsulta: 2017-05-08)
[6]https://couchdb.apache.org (kontsulta: 2017-05-08)

```
{
    hitz: {
        forma: "teilatuaren",
        morfemak: [
            {
                forma: "teilatu",
                mota: "lema"
            },
            {
                forma: "a",
                mota: "atzizki"
            },
            {
                forma: "ren",
                mota: "atzizki"
            }
        ]
    }
}
```

5.1 irudia: BSON dokumentu baten adibidea. XMLn bezala, egitura hierarkikoak eraiki daitezke.

kiaren maila bakoitza dokumentu batean adieraz daiteke. Hortaz, objektu konplexuak MongoDB datu-baseetan biltegiratzeko bi modu daude. Lehenengo aukera egitura hierarkiko osoa dokumentu bakar batean adieraztean datza. Horri dokumentuen txertatzea deitzen zaio, dokumentu sinpleak konplexuagoetan txertatuz dokumentu konplexuak eraikitzearen ideia jarraitzen baitu. Modu horretan, objektu bakoitza, konplexua izanagatik ere, kontsulta bakarrarekin eskuratuko da. Dokumentu horien gaineko kontsultak oso eraginkorrak dira, beraz. Bigarren aukera da dokumentu sinpleak bereizita gordetzea eta elkarren arteko erreferentziak definitzea. Objektu konplexuak, horrela, elkar erreferentziatzen duten dokumentu sinplez osatzen dira. Bigarren modua datu-base erlazionalek darabilten ereduaren antzekoa da, eta, modu honetan, informazio erredundantea biltegiratzea ekiditen da. Hala ere, objektu konplexu bat eskuratzeko dokumentu bat baino gehiago eskuratu behar dira, kontsulten eraginkortasunean galduz.

Argitu dezagun aurreko paragrafoa adibide batekin. Demagun hitzen informazio morfologikoa biltegiratu nahi dugula MongoDBn. Hitz bakoitzak hainbat morfema izan ditzake. Beraz, 1-N erlazioa daukagunez, hitzak objek-

```
/* teilatuaren */
{
    id: "hitz34",
    hitz: {
        forma: "teilatuaren",
        morfemak: [ "lem5_id", "atz16_id", "atz17_id"]
    }
}

/* teilatu */
{
    id: "lem5_id",
    forma: "teilatu",
    mota: "lema"
}

/* a */
{
    id: "atz16_id",
    forma: "a",
    mota: "atzizki"

}
/* ren */
{
    id: "atz17_id",
    forma: "ren",
    mota: "atzizki"
}
```

5.2 irudia: MongoDBko dokumentu bat erreferentzia bidez adierazia. Izatez, lau dokumentutan gorde da, morfema bakoitzeko bana eta hitz osoarentzat bestea. Ondoren, hitzaren dokumentu nagusian, morfemen erreferentziak gorde dira, identifikadoreen bidez.

tu konplexuak direla esango dugu. Aipatu dugun bezala, hitzen informazioa biltegiratzeko bi aukera daude. Alde batetik, hitz bakoitza, bere morfemak eta guzti, dokumentu bakar batean gorde dezakegu, egitura hierarkiko bat osatuz (ikus 5.1 irudia). Beste aldetik, hitz eta morfema bakoitza dokumentu banatan gorde daitezke, eta hitz bakoitzaren morfemak zein diren adierazteko dokumentuen arteko erreferentziak baliatu (ikus 5.2 irudia). Lehenbiziko moduaren abantaila da hitz bakoitzaren informazio osoa, morfemak barne, kontsulta bakar batekin eskura daitekeela, eta, bigarrenean, berriz, hitza eskuratzeko kontsulta bat egin beharko litzateke, eta morfema bakoitza es-

kuratzeko beste kontsulta bana. Argi dago lehenbiziko aukera, dokumentuak txertatzearena, eraginkorragoa dela. Hala ere, tamaina handiko corpus batekin lanean ari bagara, konturatuko gara morfema asko behin eta berriro errepikatzen direla, eta datu-basean behin eta berriro biltegiratzen ari garela, erredundantzia sartuz. Bigarren modua erabiliko bagenu, morfema bakoitza behin bakarrik biltegiratu genezake, eta morfema hori daukan hitz berri bat biltegiratu behar den bakoitzean morfema horren erreferentzia gorde. Horrela, datu-basearen tamaina murrizteaz gain, morfema baten informazioa aldatu beharko balitz, toki bakarrean aldatzearekin nahikoa izango litzateke.

Datu-baseetatik datuak eskuratzeko kontsulta-lengoaia bat ere eskaintzen du MongoDBk. Horrela, dokumentu mailako kontsulta eraginkorrak egin daitezke. MapReduce ereduan oinarritutako kontsulta-lengoaia bat ere eskaintzen du, bestalde.

MongoDBk izaera banatua dauka, datu kopuru oso handiak modu eraginkorrean kudeatzeko. Horrela, datuak zerbitzari bakarrean gordetzera mugatu ordez, bildumak zatitu eta zerbitzaritan bana daitezke. Horri partizionatze horizontala deitzen diogu. Biltegiratze-arazoak gainditzeaz haratago, datu kopuru handien gaineko kontsultak modu eraginkorrean egitea lortzen da horrekin. Izan ere, exekutatu beharreko kontsulta zerbitzari bakoitzean, paraleloan, exekutatuko da, bakoitzak eskuratutako dokumentuak zerbitzari nagusiari bidaliz. Zerbitzari bakoitzak datu multzo txikiagoa daukanez, kontsultak azkarragoak dira. Gainera, behin partizionatze horizontala behar bezala ezarri eta gero, eragiketak (datuak sartu, datuak eskuratu, indizeak definitu etab.) modu banatuan egitea gardena da erabiltzailearentzat, honek zerbitzari nagusiaren gainean egingo baitu eragiketa, zerbitzari bakarra balego bezala, eta MongoDB sistema bera arduratuko da eragiketa zerbitzari guztietara zabaltzeaz.

Sistema banatua izatearen abantailak ez dira eragiketen eraginkortasunera mugatzen. Partizionatze horizontala datuen erreplikazioarekin konbinatuz, sistema banatuek sendotasunean ere irabazten dute. Datuen erreplikazioa datu-base banatu bateko zerbitzari bakoitzaren edukia zerbitzari gehiagotan sinkronizatuta mantentzean datza. Horrela, zerbitzarietako bat edo batzuk bertan behera geldituta ere, datu guztiek eskuragarri egoten jarraitzen dute.

Bestalde, MongoDBren izaera dela eta, badaude datu-basearen egitura diseinatzean kontuan hartu beharreko bi ezaugarri: idazketen atomikotasuna eta dokumentuen hazkuntza. Horietako bakoitza zertan datzan azalduko dugu jarraian.

Idazketen atomikotasuna datu jakin bat bere osotasunean idazteari deitzen zaio, zati bat bidean galdu eta gainontzekoa bakarrik idazteko arriskurik izan gabe. Idazketen atomikotasuna bermatzen ez denean, datuen zati bat idatzitakoan erroreren bat gertatzen bada, baliteke zati hori idatzita gelditzea eta beste zatia idatzi gabe, datuen inkontsistentzia eraginez. MongoDBk idazketen atomikotasuna dokumentu mailan soilik bermatzen du. Horregatik, 1-N motako erlazioetan, aurrez ikusi dugun bezala, dokumentuak erreferentzia bidez adieraztea erabakitzen bada, idazketa bakoitzean hainbat dokumentu biltegiratuz, baliteke dokumentu horietako batzuk datu-basean biltegiratzea eta beste batzuk idatzi gabe gelditzea, ezusteko erroreren bat gertatu delako edo. Horregatik, idazketen atomikotasuna garrantzitsua den kasuetan, datu konplexuak dokumentu bakarrean adieraztea aholkatzen da, dokumentuak txertatzearen modua erabiliz.

Halaber, idazketen atomikotasuna bermatzeak dokumentu konplexuak eraikitzera eraman gaitzakeen arren, dokumentuen hazkuntza ere kontuan hartu beharreko ezaugarria da. Dokumentu bat sortzean, diskoaren zati bat gordetzen da harentzat. Zati hori, normalean, dokumentuak une horretan behar duen byte kopurua baino zerbait handiagoa izaten da. Horrela, etorkizunean dokumentua hazten bada, nahikoa toki izan dezake. Hala ere, dokumentua gehiegi hazten bada, erabilgarri duen espazioa gaindituz, sistemak espazio berri bat hartu behar du diskoan, eta eduki guztia espazio berrira kopiatu. Eragiketa hori, erabiltzailearentzat gardena den arren, garestia izan daiteke maiz errepikatzen bada. Horregatik, behin dokumentuak biltegiratu ondoren editatzea ohikoa den kasuetan, dokumentu konplexuak eraiki ordez, elkarren erreferentziak dituzten dokumentu sinpleak sortzea komeni da, dokumentuen hazkuntza ekidin nahi bada.

Beraz, idazketen atomikotasunaren eta dokumentuen hazkuntzaren arazoak ekiditeko, kasu bakoitzean diseinu-erabaki zuzenak hartzea beharrezkoa da, eta diseinurik egokiena datuen izaeraren menpekoa da. Hortaz, kontsulten

eraginkortasunaren, datuen erredundantziaren, idazketen atomikotasunaren eta dokumentuaren hazkuntzaren arteko oreka bilatu behar da, une bakoitzean erabakiz egitura hierarkikoak dokumentuen barruan sartzea komeni den edo dokumentu sinpleak sortu eta elkarren erreferentziak adieraztea egokiagoa izango den.

5.3. Prozesaketa banaturako teknologiak

Exekuzio jakin bat makina fisiko batean baino gehiagotan banatuta gauzatzea, horretarako espezifikoki garatutako sistemaren baten laguntzarik gabe, lan konplikatua da: makinen arteko komunikazioa bideratu behar da, datuen fluxua kudeatu, hutsegiteen aurrean erantzun egokia eman, nodoak sinkronizatu etab. Horregatik, eta gure prozesaketa-sistema banatua bideratzeko mota horretako sistema bat erabiliko dugunez, eskuragarri dauden sistemen azterketa bat egin dugu. Atal honetan, prozesaketa hainbat nodoz osatutako terminal taldeetan, modu banatuan, gauzatzen laguntzeko sistema erabilienak aurkeztuko ditugu. Prozesaketa gauzatzeko sistemez gain, zeregin horretarako lagungarri diren hainbat teknologia berri ere azalduko ditugu, besteak beste, memoria eta diskoak kudeatzeko sistemak, baliabideen kudeaketarako sistemak eta prozesaketa banaturako programazio-ereduak.

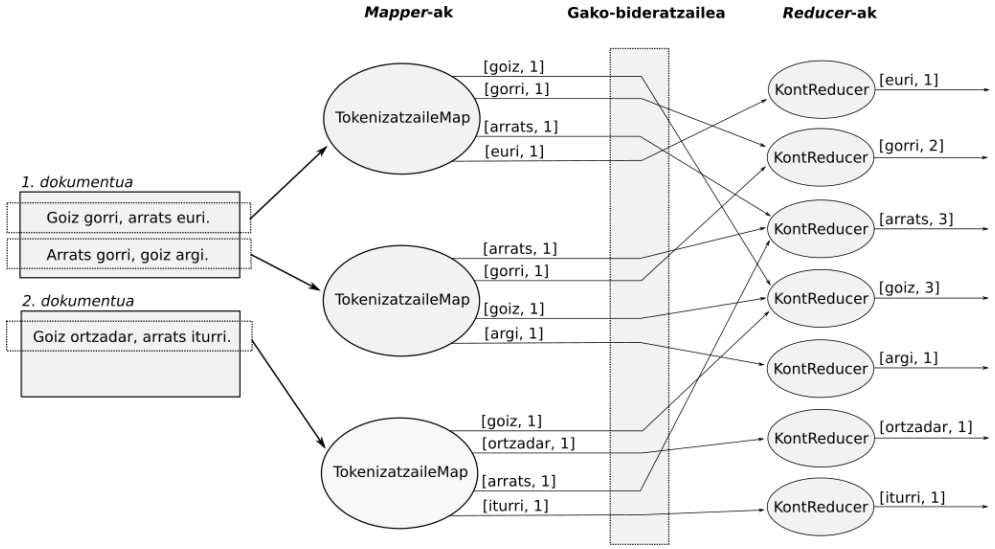
5.3.1. MapReduce

MapReduce (Dean eta Ghemawat, 2008) datu multzo handiak terminal taldeetan, modu banatuan eta paraleloan, prozesatzeko programazio-eredu bat da. Eredua ez ezik, liburutegia ere bada, ereduaren inplementazioak ere existitzen baitira. Googlek garatu eta erabili zuen, nahiz eta dagoeneko utzi dioten erabiltzeari (Sverdlik, 2014). Eredu hau Googleren oinarrizko eragiketa sinpleak datu multzo erraldoien gainean exekutatzeko beharraren ondorioz sortu zen. Hau da, exekutatu beharreko eragiketak oso sinpleak eta azkarrak ziren, baina hainbeste aldiz exekutatu behar ziren, hainbeste daturen gainean, ezen exekuzioa makina sinple askoz osatutako terminal taldeetan gauzatzea ezinbestekoa baitzen. MapReduce ereduari esker, eragiketa horiek makinetan zehar banatzea eta emaitza guztiak berreskuratzea errazten zen.

Prozesaketa terminal talde batean gauzatzean kontuan hartu beharreko alderdirik garrantzitsu eta konplikatuenetakoak honakoak dira, besteak beste: prozesuen paralelizazioa, datuen banaketa, nodoen sinkronizazioa, lan-kargaren banaketa orekatua eta erroreekiko tolerantzia. MapReduce alderdi horiek konpontzea errazteko garatu zuten, erabiltzailea aplikazioaren logika inplementatzean zentra dadin. MapReduceren ezaugarrietako bat da batch-prozesaketarako egokituta dagoela, eta, albo-ondorio bezala, streaming eredua jarraitzen duten aplikazioei ez zaie ondo egokitzen.

MapReduce aplikazio batek *map* eta *reduce* funtzioak inplementatu behar ditu. *Map* funtzioen xedea prozesu osoa zati txikiagotan banatzea da, makinen arteko lanaren banaketa errazteko. Horrela, datu multzo batetik abiatuta, aplikazioak gako-balio bikote bakunak sortu behar ditu. Ondoren, *reduce* funtzioak aurrez banatutako zatiak dagokien moduan elkartzeaz arduratzen dira, prozesatutako datuak emaitza gisa emateko. Horretarako, gako-balio bikoteak gakoaren arabera multzokatuta jaso, prozesatu, eta bikote berriak sortzen dituzte. *Reduce* funtzioaren lana murrizteko, *map* funtzio bakoitzak sortutako bikoteei funtzio konbinatzaile bat aplika dakieke, modu lokalean. Horrela, nolabaiteko *reduce* lokal bat aplikatuz, sarean zehar bikote gehiegi banatzea ekidin daiteke. Askotan, funtzio konbinatzailea *reduce*-ren berdina izan ohi da, baina modu lokalean exekutatuta.

Adibidez, demagun corpus oso bat aztertu eta hitz bakoitzaren agerpen kopurua kalkulatu eta zerrendatu nahi dugula. Dokumentuak banan-banan aztertzen dituen programa arrunt bat idatz dezakegu, baina, corpusa oso handia bada, programa horrek orduak beharko lituzke prozesatzeko. Makina sinplez osatutako terminal talde bat eskuragarri daukagunez, MapReduce programa bat inplementatzea erabaki dugu, 5.3 irudian ikus daitekeen bezala. *Map* funtzioak dokumentuen edukia zatituta jasoko du, lerro bat aldiko. Lerroa tokenizatu, eta hitz bakoitza hurrengo fasera bidaliko du. Gako-balio bikoteak bidali behar direnez, gakoa hitza bera izango da, eta balioa 1 zenbakia (ordura arteko agerpen kopurua). MapReduceren gako-bideratzaileak gako-balio bikoteak jaso eta dagozkien *reduce* funtzioetara bideratuko ditu, gako berbera duten bikoteak *reduce* funtzio berberari bidaliz. *Reduce* funtzioak bikoteak gakoaren arabera multzokatuta jasoko dituenez, funtzioaren



5.3 irudia: Hainbat dokumentu jasota, hitz bakoitzaren agerpen kopurua kontatzen duen MapReduce programa. Irudiko bi dokumentuen artean hiru lerro osatzen dituztenez, *map* funtzioaren hiru exekuzio abiaraziko dira. Adibidean 7 hitz desberdin daudenez, *reduce* funtzioa hainbeste aldiz exekutatuko da.

exekuzio bakoitzak hitz jakin baten agerpen bikote guztiak jasoko ditu. Jasotako bikote kopurua kontatuz, emaitza moduan hitzaz eta haren agerpen kopuruaz osatutako bikotea itzuliko du.

Map funtzioetatik ateratako gako-balio bikoteak sarean bidaltzen dira, *reduce* funtzioak hainbat makinatan exekuta baitaitezke. Bidaliko den bikote kopurua murriztu eta sareko trafikoa arintzeko, badaukagu funtzio konbinatzaileak gehitzea. Adibideko kasuan, testu-lerro bateko hitzen agerpenak bilduko lituzke, hitz bakoitza bere aldetik bidali ordez. Horrela, *map* funtzioek bidalitakoak baino bikote gutxiago bidaliko dituzte funtzio konbinatzaileek, sareko trafikoa murriztuz.

MapReduce programak paralelizazioari esker dira azkarragoak. Hau da, mota horretako programa bat paralelizaziorik gabe exekutatuz gero, exekuzioa ez litzateke azkarragoa ereduari egokitu gabeko programa bat baino.

Liburutegiak erroreekiko tolerantzia ere bermatzen du, bertan behera geldi-

tzen diren nodoak detektatuz eta modu egokian erantzunez. Horretarako, nodo nagusiak seinale bat bidaltzen die nodo guztiei aldian-aldian, eta erantzuna jasotzen du. Nodo baten erantzunik jasotzen ez badu, nodoa bertan behera gelditu den seinale, eta, beraz, nodo horri esleitutako eragiketak beste nodoei esleitzen zaizkie.

Ohitura bezala, gomendagarria da terminal taldeko nodo kopurua baino askoz ere *map* eta *reduce* ataza gehiago edukitzea, lan-kargaren banaketa egokia egin dadin, etekin handiagoa ateratzen baitzaio, horrela, prozesaketa banatuari.

5.3.2. Apache Hadoop

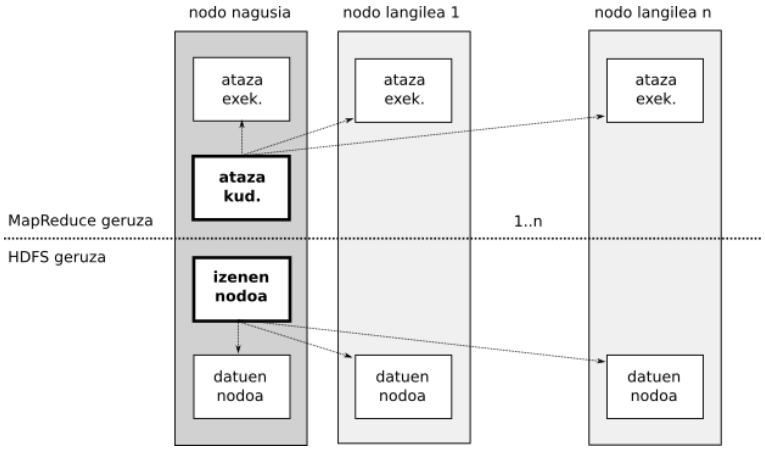
Apache Hadoop[7] MapReduceren inplementazio librea da. Algoritmoaren inplementazioaz gain, Hadoop Fitxategi Sistema Banatua (ingelesez Hadoop Distributed File System, edo HDFS) eskaintzen du, MapReduce programen exekuzioaren datuak gorde, trukatu eta eskuratzeko ingurune izateko inplementatutako fitxategi-sistema banatua (Shvachko *et al.*, 2010).

Hadoop programak terminal taldeetan exekutatzen dira. 5.4 irudian ikus daitekeen bezala, bi motatako nodoak daude terminal talde batean: nodo nagusi bat eta hainbat nodo langile. Nodo guztiek dituzte bai MapReduceren eta bai HDFSren moduluak, nodo bakoitzak bi gauzak egiten baititu: *map* eta *reduce* funtzioak exekutatu, eta datuak HDFSn gorde.

Hadoopen ezaugarri garrantzitsu bat atazak kudeatzeko modu zentralizatua da. Izan ere, 5.4 irudian ikus daiteke nola terminal talde osoan ataza-kudeatzaile bakarra dagoen. Haren lana da, beraz, nodo bakoitzeko ataza-exekutatzaileari zein ataza gauzatu behar duen agintzea. Hori dela eta, terminal taldeko nodo kopurua oso handia denean, eskalagarritasunarekin arazoak sor ditzake (ikus 5.3.5 atala).

Googlek MapReduce lehenbiziko aldiz inplementatu zuenean, Google Fitxategi Sistema (ingelesez Google File System, edo GFS) erabiltzen zuen. Hadoopek, berriz, GFSn oinarrituta Hadoopentzat espezifikoki garatutako HDFS erabiltzen du. Datuen iraunkortasuna eskaintzeaz gain, erabilgarrita-

[7] `https://hadoop.apache.org` (kontsulta: 2017-05-08)



5.4 irudia: Hadoopen arkitektura. Nodo nagusiak, MapReduce funtzioen exekuzioak kudeatzeaz gain (ataza kud.), exekuzioak ere egiten ditu (ataza exek.). HDFSri dagokionez, izenen nodoa, datuen nodo bakoitzean zein datu aurkitzen diren adierazten duena, nodo nagusian aurkitzen da.

suna ere hobetzen du, eta nodo batek huts egiten duenean egoera egonkor batera itzultzen ere laguntzen du. UNIX fitxategi-sistemaren itxura duen arren, GFSren kasuan estandarrak errendimenduaren ordainetan sakrifikatu ziren.

HDFSn nodo nagusi bat (izenen nodoa) eta hainbat datu-nodo daude. Datuak datu-nodoetan idazten dira, eta izenen nodoak datu bakoitza zein nodotan gordeta dagoen adierazten duen informazioa gordetzen du. Gainera, datu bakoitzaren hiru erreplika gordetzen dira, hainbat nodotan. Horrela, nodo batek huts egiten badu ere, datuek erabilgarri egoten jarraitzen dute. Hori horrela izanik, datuak gorde edo eskuratu nahi dituen aplikazioak HDFS bezeroari egin behar dio eskaera, hark izen-nodoari galdetuko dio eskatutako datuak non aurki ditzakeen edo non idatziko dituen, eta, informazio horrekin, datu-nodoetara joko du. Horrela, aplikazioek ez dute HDFSren barne-egitura ezagutu beharrik berau erabiltzeko.

2013 urteaz geroztik Apache Hadoopen parte den beste modulu bat YARN (Vavilapalli *et al.*, 2013) baliabide-kudeatzailea da. Apache Hadoopen hasierako arkitekturak bi arazo nagusi zituen: alde batetik, programazio-ere-

duaren eta baliabideen kudeaketaren arteko lotura estua, eta, beste aldetik, ataza-kudeatzailearen zentralizazioa, zeinak ataza asko exekutatu behar diren aplikazioetan eskalagarritasun-arazoak sor ditzakeen. Bi arazo horiek konpontzeko garatu zuten YARN, programazio-eredua baliabideen kudeaketatik bereiziko zuen modulua.

5.3.3. Apache Apex

Apache Apex[8] Apache Hadoopen gainean inplementatutako datu handientzako prozesaketa-sistema bat da. Datu multzo finkoak eta datu-korronte mugagabeak prozesatzeko gaitasuna dauka, batch eta streaming ereduetara egokituz.

Eskalagarritasun lineala eskaintzen du Hadoop terminal taldeetan zehar, baita segundoko milioika gertakari prozesatu behar diren kasuetan ere. HDFS teknologiari esker, hardwarearen eta prozesuen hutsegiteak berehala antzematen dira. Garatzaileentzako API sinple eta argia eskaintzen du, kode garbia eta berrerabilgarria sor dezaten. Bestalde, kudeaketa-eragiketak eta eragiketa funtzionalak bereizten ditu, eta, gainera, kudeaketaren zati handiena automatikoki egiten du sistemak.

Apex aplikazioen garapena errazteko, hainbat funtzionalitate biltzen dituen Apache Apex Malhar[9] liburutegia ere eskuragarri dago. Besteak beste, datuak Hadoop ingurunera eraman eta, behin prozesatuta, kanpoko biltegiratze-sistemetara eramateko lana errazten du, datuok hainbat datu-base, fitxategi-sistema, ilara-sistema eta sare sozialekin integratzen lagunduz.

5.3.4. Apache Twill

Apache Twill[10] Apache Hadoop YARN teknologiaren gainean eraikitako abstrakzio-geruza bat da. Horri esker, garatzailea aplikazioaren logikan kontzentra daiteke, ingurune banatuen eta paralelizazioaren kontuak YARNen esku utziz. Horrela, ingurune banatuetarako programak garatzeko, Java hariak

[8] `https://apex.apache.org` (kontsulta: 2017-05-08)
[9] `https://github.com/apache/apex-malhar` (kontsulta: 2017-05-08)
[10] `https://twill.apache.org` (kontsulta: 2017-05-08)

kudeatzearen antzeko programazio-eredua eskaintzen du Twillek.

Twill aplikazioak kudeatzeko honako funtzionalitateak ere erabilgarri daude: aplikazioen bizi-zikloaren kudeaketa, prozesu banatuen koordinazioa, hutsegiteekiko tolerantzia eta zerbitzuen kudeaketa.

5.3.5. Facebook Corona

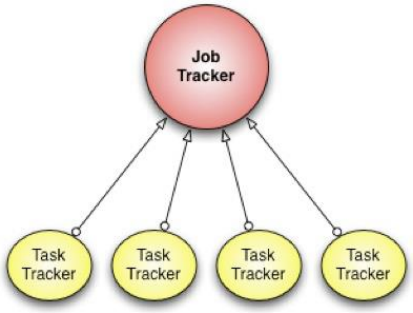
Facebook Corona[11] Facebookek Hadoopen gainean egindako aldaketa batzuen ondorioz sortutako prozesaketa banaturako sistema da. Facebookek, hasiera batean, Hadoop erabiltzen zuen prozesaketa handiak egiteko. Alabaina, Hadoop teknologia ere txiki gelditu zitzaien halako batean, eta hainbat moldaketa egin behar izan zizkioten egun Facebook Corona izenaz ezagutzen den sistema lortzeko.

Facebookeko ingeniariek publikatutako artikulu batean (Facebook, 2012) azaltzen zutenaren arabera, arazo nagusia Hadoopen ataza-kudeatzailean aurkitu zuten. Izan ere, 5.3.2 atalean azaldu dugun bezala, Hadoopek nodo nagusian exekutatzen den ataza-kudeatzaile bakarra erabiltzen du ataza guztiak terminal talde osoko ataza-exekutatzaileen artean banatzeko (ikus 5.5 irudia). Nodo kopurua izugarri handia denean, ataza-kudeatzaile bakar hori ez da nahikoa, eta terminal taldearen prozesatze-ahalmena nabarmen jaisten da.

Hadoopen beste muga bat ere antzeman zuten Facebookeko garatzaileek, hori ere atazen banaketa-ereduarekin lotua. Izan ere, ataza-exekutatzaileek, haien egoeraren berri emateko, seinale bat bidaltzen diote ataza-kudeatzaileari periodikoki. Ataza-exekutatzaile jakin bat zain geldituko da, ezer egin gabe, libratu den unetik hurrengo seinalea bidaltzeko unea iritsi arte, eta, ondorioz, beti dago atzerapen txiki bat edozein ataza abiarazteko unean. Atazak oso laburrak baina ugariak direnean, atzerapena nabaria bihurtzen da.

Azkenik, Hadoopen baliabideen kudeaketa *slot* kopuruaren araberakoa da. Hau da, terminal talde osoak *map* eta *reduce* funtzio kopuru jakin bat exekutatzeko *slot* kopurua dauka, eta horren arabera kudeatzen dira baliabideak.

[11] `https://github.com/facebookarchive/hadoop-20/tree/master/src/contrib/corona` (kontsulta: 2017-05-08)



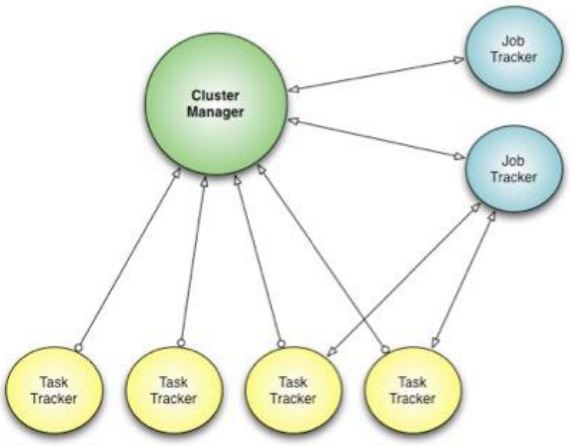
5.5 irudia: Hadoopek ataza-kudeatzaile bakarra ezartzen du terminal taldeko ataza guztiak banatzen eta ataza-exekutatzaile guztiak gainbegiratzen.

Aldiz, batzuetan, baliabideak neurri xeheagoen arabera kudeatzeko beharra izaten da, hala nola, PUZaren erabileraren edo memoria kopuruaren arabera.

Arazo horiei guztiei konponbidea ematen die Facebook Coronak. Horretarako, Coronaren arkitekturan, baliabideen kudeaketa eta atazen kudeaketa bereizi egiten dira. Coronak terminal taldearen kudeatzailearen kontzeptua gehitzen dio bere arkitekturari (ikus 5.6 irudia). Prozesu horren lan bakarra terminal taldeko nodoen eta erabilgarri dauden baliabideen jarraipena egitea da. Gainera, ataza bakoitzak bere kudeatzaile propioa dauka, eta, terminal taldearen kudeatzailearen lana arintzeko, ataza-kudeatzaileek autonomia osoa daukate beren atazak kudeatzeko garaian, terminal taldearen kudeatzaileak ez baitu atazen egoera gainbegiratuko.

Bestalde, ataza-exekutatzaileek ez diete seinale periodikorik bidali behar kudeatzaileei beren egoeraren berri emateko (*pull* eredua), baizik eta libre daudenean modu aktiboan bidaltzen diete abisua ataza-kudeatzaileei. Horrela, ataza-exekutatzailea libratu bezain laster jakinarazten zaio kudeatzaileari, eta honek ataza berri bat esleituko dio denborarik galdu gabe (*push* eredua).

Baliabideen kudeaketa zehatzagoa egiteko funtzionalitatea, teknologia hau deskribatzen duen artikulua publikatu zuten unean, garapen-prozesuan zegoen oraindik. Horrekin batera, softwarearen eguneraketak prozesu guztiak amaitu behar izan gabe egiteko aukera ere garatzen zebiltzan.



5.6 irudia: Coronan, ataza mota bakoitzak bere kudeatzaile propioa dauka. Horrela, terminal taldearen kudeatzailearen lana oso arina da, atazen oso azaleko kudeaketa besterik ez baitu egin behar.

5.3.6. Apache Spark

Apache Spark[12] prozesaketa banaturako kode irekiko beste sistema bat da, hau ere batch-prozesaketara zuzendua (nahiz eta aurrerago streaming-prozesaketarako ere egokitua izan). Kaliforniako Berkeley Unibertsitatean sortua, gero Apache fundazioak hartu zuen ardura bere gain.

MapReduceren erabilera asko zabaldu zen arren, haren desabantailak ere agerian gelditu ziren. Izan ere, prozesaketaren iterazio bakoitzean datuak diskoan idazten dira. Eta ez hori bakarrik, erreplikazioa dela eta, datu bakoitza hainbat nodotan idatzi behar baita. Datuak biltegiratzeko modu haren erruz, Hadoop programen exekuzio-denboraren zati handi bat disko-eragiketei zegokien. Hori ikusita, ikertzaileek sistema berri bat garatu zuten, Hadoop eta MapReduceren oinarri bera zuena, baina datuen garraioa eta biltegiratzea optimizatuko zuena. Horrela jaio zen Apache Spark.

Helburu hori lortzeko, Sparkek *Resilient Distributed Dataset* (RDD) izeneko egitura dinamikoak erabiltzen ditu. RDD bat datu multzo aldaezin bat da, terminal talde osoan erabilgarri dagoena. RDDetako datuak Python,

[12] `https://spark.apache.org` (kontsulta: 2017-05-08)

5.7 irudia: Sparkek ere streaming-prozesaketa egin dezake, Spark Streaming izeneko hedapenari esker.

Java edo Scala lengoaietako edozein objektu izan daitezke. Gakoa da, RDDak diskoan idatzi ordez, memoria dinamikoan mantentzen direla, ahal den neurrian. Horrela, *map*, *reduce* edo bestelako funtzioen artean datuak elkarbanatu behar direnean, RDDen bitartez egiten da, ahal den neurrian memoria dinamikoa erabiliz. Noski, datuak nodoen artean banatu behar badira, RDDak sarearen bitartez banatuko dira.

Datuen banatze dinamikoari esker, Apache Sparken azpiegitura oso egokia da aplikazio iteratiboetarako, hau da, datu multzo handien gainean eragiketa berberak behin eta berriz aplikatu behar diren kasuetarako. Gainera, aplikazio interaktiboetarako ere onurak eskaintzen ditu, datu multzoak kontsulta bidez eskuratu behar direnean ere memorian mantentzen baititu behin eskuratuta, datu horien gaineko hurrengo kontsultak nabarmen arinduz.

Spark Streaming Spark nagusiaren hedapen bat da, eta horri esker, sistemari streaming-prozesaketarako ahalmena ere gehitu zitzaion. Hala ere, moldaketa bat besterik ez da, eta geruza horrek benetan egiten duena honakoa da: Spark Streaming moduluak iturri batetik etengabeko datuak jasotzen ditu, datuak zati edo batchetan biltzen ditu, eta Spark modulu nagusiari batch horiek bidaltzen dizkio, banan-banan, honek ohiko eran prozesa ditzan. 5.7 irudian ikus daiteke azaldu berri dugun prozesua.

5.3.7. S4: stream-konputazio banaturako plataforma

S4 (Neumeyer *et al.*, 2010) ere prozesaketa terminal taldeetan banatzeko kode irekiko sistema bat da, baina streaming-prozesaketara zuzendua. S4-k hutsegiteekiko tolerantzia partziala eskaintzen du, baita datuen erabilgarritasuna eta aplikazioen eskalagarritasuna ere. Hau ere MapReducen oinarrituz garatu

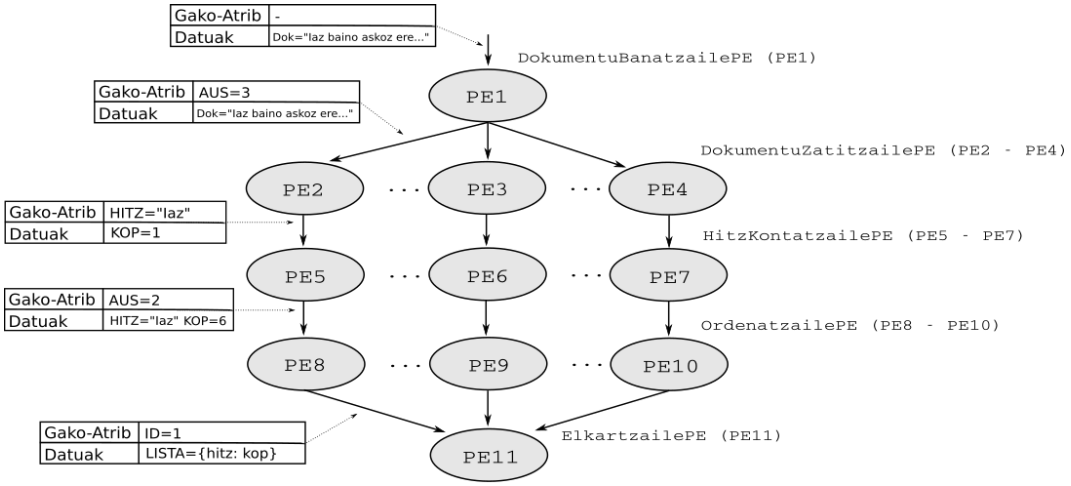
zuten, baina hasieratik streaming ereduarekin bat etortzeko diseinatuz.

S4-ren diseinuaren unitate garrantzitsuenak prozesaketa-elementuak (PE) dira. PE bakoitzak prozesaketaren zati baten funtzionalitatea kapsulatzen du. PE-en artean datuak trukatzeko modu bakarra, batak besteari, zuzenean, mezuak bidaltzean datza. Mezu horiei gertaera deritze S4-n. Eredu horri esker, S4 programek kapsulazio eta gardentasun maila altuak lortzen dituzte, arkitektura konplexuak ekidinez. S4-k ere, Apache Sparken antzera, memoria dinamikoa erabiltzen du nodo mailako datuak gorde eta trukatzeko, disko-eragiketek eragindako botila-lepoak saihestuz. S4-ren arkitektura ez da zentralizatua, ez baitago nodo nagusirik. Nodo guztiak maila berean daude.

PE-en instantzia bakoitza lau ezaugarri hauen arabera identifikatzen da unibokoki: haren funtzionalitatea, kontsumitzen dituen gertaeren mota, gertaera horien gako-atributua eta gako-atributuaren balioa. Alegia, PE mota bakoitzak mota jakin bateko gertaerak kontsumitzen ditu. Zehazki, mota horretako PE-en instantzia bakoitzak, gertaera horien artean, gako-atributuaren balio jakin bat dutenak bakarrik jasotzen ditu. Horrela, gako-atributuaren balio desberdin bakoitzeko, PE instantzia berri bat sortzen da. Kontzeptu horiek hobeto ulertzeko, adibide bat azalduko dugu jarraian:

Demagun gure sistemara dokumentuak bidaltzen dituen datu-korronte bat daukagula. S4 aplikazio bat inplementatu dugu, zeinak dokumentu horiek prozesatuz, aldian-aldian, gehien agertu diren hitzen zerrenda eguneratu bat itzultzen duen. Prozesu osoa bost zatitan banatu dugu, eta horietako bakoitza gauzatzeko PE mota bat sortu dugu:

- `DokumentuBanatzailePE`: Datu-korrontetik dokumentuak jaso eta gako baten arabera banatzen ditu, fluxu paraleloak sortuz.
- `DokumentuZatitzailePE`: `DokumentuBanatzailePE`-k banatutako dokumentuak jaso eta hitzetan zatitzen ditu. Hitz bakoitzarekin gertaera berri bat sortzen du, hitz horrek dokumentuan izan duen agerpen kopuruarekin. Jasotako gertaerak, aldiz, ausaz banatzen dira instantzien artean.
- `HitzKontatzailePE`: Hitz baten agerpen kopuru berri bat iristean (`Do-`



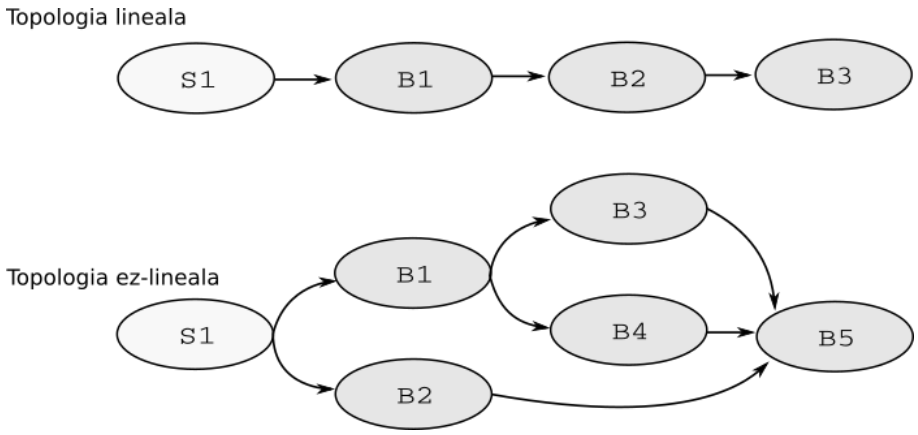
5.8 irudia: S4 programa baten adibidea. Programak, datu-korronte mugagabe batetik, testu-dokumentuak jasotzen ditu, eta dokumentu guztien artean agerpen gehien izan dituzten hitzen zerrenda osatzen du denbora errealean.

`kumentuZatitzailePE`-k dokumentu berri bat prozesatzean), hitz horren agerpen kopurua eguneratu eta informazio eguneratuarekin gertaera berri bat sortzen du. Garrantzitsua da jasotako gertaeren gako-atributua hitza bera izatea, horrela, hitz berari dagozkion agerpen guztiak instantzia berberari iritsiko baitzaizkio.

- `OrdenatzailePE`: Hitzen agerpen kopuru berriak etengabe jasoko dituenez, datu berri bat jasotzen duen bakoitzean gehien agertu diren hitzen zerrenda eguneratu eta gertaera berri batean bidaltzen du informazio eguneratua. Jasotako gertaerak ausaz banatuko dira PE honen instantzien artean.

- `ElkartzailePE`: Agerpen gehien dituzten hitzen zerrendak jasoko ditu etengabe. Bere lana PE instantziek sortutako zerrenda partzialak jaso eta elkartzea da, zerrenda osatua lortuz. PE honen izaera dela eta, instantzia bakarra egongo da, zerrenda partzial guztiak instantzia berberak jaso ditzan.

5.8 irudian ikus daiteke deskribatutako kasua xehetasun gehiagoz. Paraleliza-



5.9 irudia: Storm programen bi topologia posible. Lehena lineala da eta bigarrena ez-lineala.

zioa PE bakoitzaren hainbat instantzia sortuz lortzen da, instantzia bakoitza datu jakin batzuk prozesatzeaz arduratuko baita.

5.3.8. Apache Storm

Apache Storm[13] streaming eredua jarraitzen duen prozesaketa banaturako beste sistema bat da, kode irekikoa hau ere. Eskalagarria eta hutsegiteekiko tolerantea da, eta bidalitako datu guztiak prozesatuak izango direla bermatzen du.

Storm programak topologia baten bidez definitzen dira. Topologiak, era berean, *spout* eta *bolt* motako nodoz osatutako grafo zuzenduak dira (ikus 5.9 irudia). Nodo bakoitza prozesaketa-unitate bat da, eta datuak nodoz nodo pasatzen dira, nodo bakoitzaren irteera hurrengoek sarrera moduan jasoz. Spout nodoak datu-iturriak dira, eta prozesatu gabeko datu gordinak topologian sartzeaz arduratzen dira. Nodo horiek, noski, ez dute beste nodorik beren aurretik. Spoutek edo beste boltek bidalitako datuak jaso eta prozesatzen dituzte bolt nodoek.

Topologiak linealak edo ez-linealak izan daitezke. Topologia lineala da bolt guztiek sarreran beste spout edo bolt bakar bat daukatenean lotuta, eta bes-

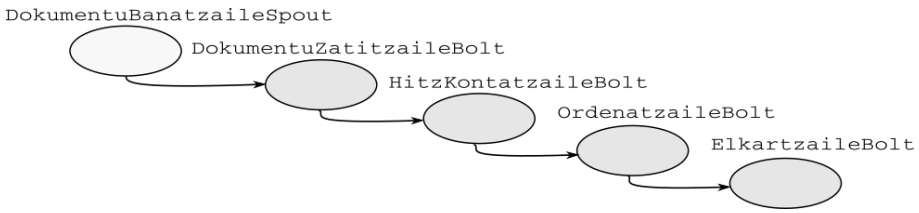
[13] `https://storm.apache.org` (kontsulta: 2017-05-08)

te bakar bat irteeran. Kasu horretan, bolt bakoitzaren instantzia bat baino gehiago jartzea beharrezkoa da paralelizazioa egon dadin. Aldiz, gutxienez boltetako batek sarreran edo irteeran bolt bat baino gehiago konektatuta daukanean, topologia ez-lineala dela esaten dugu. Topologia ez-linealen kasuan, paralelizazioa gerta daiteke bolten instantzia bakarrarekin ere, datu beraren gainean prozesaketa independenteak exekuta baitaitezke aldi berean. 5.9 irudiko bigarren topologian, adibidez, S1 spoutak datu berri bat bidaltzean, B1 eta B2 boltak paraleloan exekutatuko dira. B1-ek bukatzean, B3 eta B4 exekutatuko dira, B1-ek bidalitako datuekin. B2, B3 eta B4-k bukatzean, berriz, B5 exekutatuko da, aurreko hiruren datuekin.

Datu asko prozesatu behar diren kasuetan, terminal taldeko nodo kopurua topologiako nodo kopurua baino askoz ere handiagoa izan daiteke, eta paralelizazio maila egokia lortzeko ezinbestekoa da spout, eta, batez ere, bolt bakoitzaren instantzia ugari sortu eta terminal taldean zehar banatzea. Horrela, instantzia bakoitza datu jakin batzuk prozesatzeaz arduratuko da, modu sinplean arkitektura konplexuak eraikiz.

Topologian zehar bidaltzen diren datuek tupla deituriko egituretan bidaiatzen dute. Tupla batean, gako-balio motako nahi adina bikote sar daiteke. Gainera, datuek edozein mota izan dezakete.

Hitz-kontaketen adibidea hartuz, Apache Stormen kasuan nola egingo genukeen azalduko dugu jarraian. Adibide honetan, kanpoko datu-korronte batetik testu-dokumentuak iristen dira, mugarik gabe, eta Storm programa bat inplementatu nahi dugu dokumentu horiek prozesatu eta, bakoitzarekin amaitzean, dokumentuen artean gehien agertu diren hitzen zerrenda bat eguneratuz. Prozesua, S4 sistemaren adibidean egin bezala, bost zatitan banatu dugu. Lehena spout bat izango da (`DokumentuBanatzaileSpout`), eta datu-korrontetik dokumentuak jaso eta topologiako lehenbiziko boltera bidaliko ditu. Ondorengo laurak boltak izango dira: lehenak (`DokumentuZatitzaileBolt`), dokumentuak jaso eta hitzetan zatituko ditu; bigarrenak (`HitzKontatzaileBolt`), hitzak jaso eta agerpen kopuruak eguneratuko ditu; hirugarrenak (`OrdenatzaileBolt`), boltaren instantzia bakoitzak jasotako emaitzekin zerrenda partzialak eraikiko ditu; azkenik, laugarrenak (`ElkartzaileBolt`), zerrenda partzialak elkartuz, zerrenda osatua eraikiko du. 5.10 irudian ikus



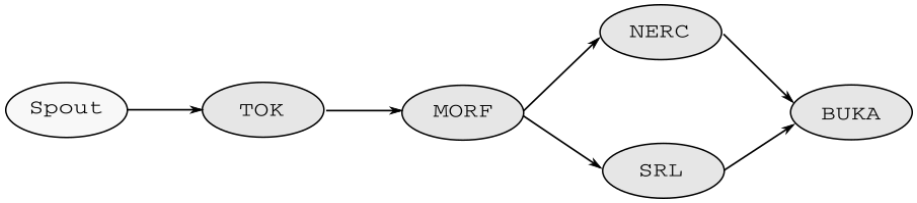
5.10 irudia: Gehien agertu diren hitzak ordenatuta zerrendatzeko adibidearen Storm topologia lineala.

daiteke diseinatutako Storm topologia lineala.

`DokumentuZatitzaileBolt` eta `OrdenatzaileBolt`-en instantzien artean ausaz banatzen dira tuplak. `HitzKontatzaileBolt`-en kasuan, berriz, ezinbestekoa da hitz baten agerpen guztiak boltaren instantzia berberak jasotzea, ezinbestekoa kontaketa behar bezala egin nahi bada. Horretarako, Stormek aukera ematen du, S4-k bezala, tuplak gako-atributu baten arabera multzokatzeko. Kasu honetan, gako-atributua hitza bera izango da, eta, horrela, hitz beraren agerpen guztiak `HitzKontatzaileBolt`-en instantzia berera bideratuko dira. `ElkartzaileBolt`-en instantzia bakarra dagoenez, tuplak ausaz bideratu daitezke.

Adibidean, bolten hainbat instantzia sortuz lortzen da paralelizazioa. Norberaren esku gelditzen da spout eta bolt bakoitzaren instantzia kopurua zehaztea, terminal taldearen tamainaren arabera.

Orain, topologia ez-linealen adibide bat ikusiko dugu. Demagun dokumentuak analisi-kate batekin prozesatu nahi ditugula. Analisi-kateak lau HP ditu: tokenizatzailea (`TOK`), morfosintaxi-analizatzailea (`MORF`), entitate-izenen ezagutzailea (`NERC`) eta rol semantikoen etiketatzea (`SRL`). Gainera, badakigu `MORF`ek `TOK`ek sortutako datuak behar dituela, eta `NERC`ek eta `SRL`k `MORF`ek sortutakoak behar dituztela. HP bakoitza bolt batean bil dezakegu, spout batez eta lau boltez osatutako topologia bat eratuz. Aukera bat topologia lineala osatzea da, bolt bakoitza bestearen atzean kokatuz. Hala ere, kasu honetan topologia ez-lineala ere eraiki dezakegu. Izan ere, badakigu `NERC` eta `SRL`k ez dutela elkarren emaitzen beharrik, eta, hortaz, bi ataza horiek paraleloan exekuta daitezkeela. Horrela, 5.11 irudiko topologia eraiki genezake. Bukaeran gehitu dugun `BUKA` boltak, besterik gabe, paraleloan sortu dituzten



5.11 irudia: Dokumentuen prozesaketa lau HPko analisi-kate batekin egiten duen topologia ez-lineala.

emaitzak jaso eta elkartu egiten ditu. Topologia linealarekin alderatuz, azken honek badauka onura aipagarri bat: dokumentu bakoitzaren prozesaketa azkartu egingo da, lau pausotan prozesatu ordez hiru pausotan prozesatuko baita.

Hurrengo kapituluan aurkeztuko dugun arkitekturaren prozesamendu-kudeaketa garatzeko aukeratu dugun plataforma Storm izan da. Izan ere, bai batch eta bai streaming ereduetara egokitzen den sistema bat behar genuen, eta horretarako egokiena Storm iruditu zaigu. Batetik, topologietan oinarritzen den Storm programen diseinua intuitiboa eta sinplea iruditu zaigu, eta gure beharretara bete-betean egokitzen dela ikusi dugu. Bestetik, sistema hedatuagoa dagoela ikusi dugu, eta horrek Stormen komunitatea sendoagoa izatea ekarri du. Gainera, dokumentazioa ere osatuagoa dago, eta horrek inplementazio-lanak erraztu dizkigu.

5.3.9. Apache Ignite

Apache Ignite[14] prozesaketa banatua gauzatzeko memorian oinarritutako plataforma da. Apache Sparken antzera, diskoa alde batera utzi eta memoria dinamikoaren alde egiten du, abiaduran irabaztearren. Izatez, prozesaketa memorian egitetik haratago, datuen idazketa eta biltegiratzea ere memorian egiten da. Hadoopen eredutik aldentzen da, beraz, Hadoopek prozesatutako datuak diskoan idazten baititu, baita berehala beste prozesu batek datu horiek prozesatu behar dituen kasuetan ere.

Prozesaketa-eredua baino gehiago, prozesaketa ahalbidetzeko plataforma bat

[14] `https://ignite.apache.org` (kontsulta: 2017-05-08)

da. Esaterako, Igniteren MapReduceren inplementazioa Hadoopen interfazearekin bateragarria da. Hau da, printzipioz, edozein Hadoop programak Ignite plataformaren gainean exekutagarria izan behar luke, askoz ere abiadura handiagoak lortuz. Datuak IgniteCache izeneko memorian idazten dira. IgniteCachek JCache-ren (JSR 107) espezifikazioak betetzen ditu. Horrez gain, IgniteCachek datuak atzitzeko aukera zabala eskaintzen du, besteak beste, SQL notazioa erabiltzeko aukera ematen baitu.

Ignitek Java teknologia erabiltzen du. Hortaz, kontuan hartuz plataforma datu kopuru handiekin lan egiteko pentsatuta dagoela, Javaren zabor-biltzaileak eragindako eraginkortasun-arazoak ekiditeko, *on-heap* eta *off-heap* memoriak bereizten ditu. Zabor-biltzaileak Javaren *on-heap* memoriako datuen gainean bakarrik egiten du lan, baina Ignitek datuak zabor-biltzailearen irismenetik kanpo gelditzen den *off-heap* memoria erabiltzeko aukera ematen du. *Off-heap* memoriak cache baten antzera jokatzen du, baina, beti ere, RAM memoria erabiliz.

Ignite ere oso plataforma egokia da datu-korronte mugagabeak streaming ereduari jarraituz prozesatzeko. Ikuspuntu berdin-berdina ez izan arren, Apache Sparkekin alderagarria dela esan daiteke.

5.3.10. Apache Flink

Apache Flink[15] prozesaketa banaturako beste sistema bat da, batch- eta streaming-prozesaketak egiteko ahalmena daukana. Alde horretatik, Apache Sparkekin aldera daiteke, honek ere batch eta streaming ereduak jarraitzen baititu. Batch-prozesaketaren eredua oso antzekoa da bien kasuan. Hala ere, Sparkek streaming-prozesaketa egiteko ere datuak batch txikitan bildu behar izaten ditu, eta, beraz, azpitik batch-prozesaketaren sistema bera erabiltzen du streamingaren kasuan ere. Aldiz, Flinken streaming eredua Apache Stormenaren antzekoagoa da, biek *pipeline*-etan oinarritutako prozesaketa egiten baitute.

Flinkek interfaze erosoa eskaintzen du programatzeko. Besteak beste, *map*, *groupBy*, *join* eta *window* funtzioak eskaintzen ditu, erabiltzeko prest. Era-

[15] `https://flink.apache.org` (kontsulta: 2017-05-08)

giketa horiek oso ohikoak dira prozesaketa banatuaren testuinguruan, eta Stormen kasuan, adibidez, eskuz inplementatu behar ditu garatzaileak. Bestalde, Flinkek datu bakoitza zehazki behin prozesatuko dela bermatzen du. Stormek, esate baterako, gutxienez behin prozesatuko dela bermatzen du, errore jakin batzuk daudenean ezin baitu bermatu ez dela datu bera behin baino gehiagotan prozesatuko.

Flinken beste ezaugarri bat datuak elkarbanatzeko latentziaren kudeaketa da. Datu bat prozesaketa-kateko elementu batek prozesatzen duenean, ez dio unean bertan kateko hurrengo elementuari bidaltzen. Bidalketa sarean zehar egin behar denez, eraginkortasun kontuak direla eta, datuak irteerako buffer batean gordetzen ditu, bufferra betetzen den arte. Hala ere, latentzia txikiak ezinbestekoak diren kasuetarako, datuek bufferrean pasa dezaketen gehienezko denbora ere zehatz daiteke.

5.3.11. Google Cloud Dataflow

Google Cloud Dataflow[16] prozesaketa banatua hodeian egitera bideratutako plataforma da. Abantaila nagusia eskalatze dinamikoa da, hau da, behin erabiltzaileak bere prozesaketa-eredua programatuta, sistemak berak kudeatzen dituela behar dituen baliabideak, automatikoki.

Eskalatze dinamikoa ia beharrezkoa da baliabideen kudeaketa eraginkorra egin nahi bada (Anderson eta Dvorsky, 2016). Baliabideak eskuz esleitu behar direnean, baliabide nahikoa ez erreserbatzeko arriskua dago alde batetik. Kasu horretan, prozesaketa behar baino motelago joango da, nahiz eta baliabide fisiko gutxiagorekin moldatu. Beste aldetik, berriz, kontrakoa ere gerta daiteke, hau da, behar baino baliabide gehiago esleitzea exekuzioari. Kasu horretan, prozesaketa eraginkorra izango litzateke denbora aldetik, baina garestia ekonomikoki. Baliabideen batez besteko beharretara ongi egokituz gero ere, ez litzateke eskalatze dinamikoarekin lor daitekeena bezain kudeaketa eraginkorra lortuko, izan ere, baliabideen beharrak asko alda baitaitezke prozesaketak iraun bitartean. Beraz, beti ez da nahikoa baliabideak prozesaketa abiarazi aurretik kalkulatzea. Baliabideak dinamikoki erreserbatzea eta askatzea da eraginkorrena.

[16] `https://cloud.google.com/dataflow` (kontsulta: 2017-05-08)

Soluzio horri esker, erabiltzaileak prozesaketaren nondik norakoak programatuko ditu, baina ez du baliabideen beharrez kezkatu behar. Sistemak, beharraren arabera, baliabideak erreserbatu edo askatuko ditu, une bakoitzean prozesaketa astunagoa edo arinagoa den ikusita. Kontuan hartu behar da, Googlek erabilitako baliabideen arabera kobratuko duela, eta, beraz, erabiltzaileak ezin izango duela inoiz erabiliko den baliabide kopuruaz zeharo ahaztu.

Horretarako, Googlek hodeiko konputazioan oinarritutako ekosistema konplexua dauka atzetik. Google Cloud Platform azpiegituraren baitan, hodeian oinarritutako zerbitzu ugari jartzen ditu enpresak eskuragarri, besteak beste, konputazioa, biltegiratzea, sarea, segurtasuna eta kudeaketa-aplikazioekin lotutako zerbitzuak.

Programazio-ereduari dagokionez, Apache Beam[17] teknologia erabiltzen da. Hasiera batean Googlek berak garatua, eta, egun, Apache lizentziapean banatua, batch eta streaming prozesaketa-ereduetara behar bezala egokitzen da Beam. Erabiltzailearentzat gardena da datu-iturrien izaera, Beamen kasuan, batch- edo streaming-prozesaketa izateak ez baitu programatzeko era aldatzen. Bestalde, Beamek ez dauka Google Cloud Dataflowekiko dependentzia zuzenik, eta, adibidez, Apache Apex, Apache Flink edota Apache Spark sistemekin integra daiteke. Kasu horietan, Beam erabiliz garatutako kodea beste ingurune horietakoren batean exekutatuko litzateke.

5.4. Hizkuntzaren prozesamendua ingurune banatuetan

Testuen prozesaketa XX. mendean ere egiten zen arren, prozesatu beharreko testu bildumak nabarmen handitu dira XXI. mendean zehar. Testuinguru askotan, ordenagailu arrunt bat ez da nahikoa egin beharreko prozesaketa guztia denbora-tarte onargarri batean burutzeko. Horregatik, XXI. mendeko bigarren hamarkada honetan, teknika berriak garatzen ari dira datu handien prozesaketa gauzatzeko.

[17] `https://beam.apache.org` (kontsulta: 2017-05-08)

2. kapituluan aipatu dugun bezala, testuen prozesaketarako sistema ezagunenetako bat UIMA (Ferrucci eta Lally, 2004) da. UIMA testuak prozesatzeko ingurune bat da, hizkuntza-prozesatzaileak elkartuz analisi-kateak osatzeko lana errazten duena. Elkarreragingarritasunari dagokionez aurrerapauso handia izan zen, eta gaur egun, oraindik, asko erabiltzen da. Hasiera batean, UIMAk ez zuen eskalagarritasuna kontuan hartu. Gerora, UIMA-AS[18] (UIMA Asynchronous Scaleout) modulua gehitu zioten, UIMA aplikazioak eskalagarri egiten zituena. Hala ere, UIMA-ASek ere ez zuen ahalmenik eskaintzen terminal taldeak eta horien atazak, prozesuak eta baliabideak kudeatzeko. Hori konpontzeko, Distributed UIMA Cluster Computing[19] (DUCC) modulu gehigarria erantsi zioten.

Testuak prozesatzeko beste ingurune bat Cocytus (Evans *et al.*, 2008) da. Cocytus hasieratik dago ingurune banatuetan exekutatzeko diseinatuta. Horretarako, prozesaketa banaturako sistema bat erabili beharrean, Inferno (Dorward *et al.*, 1997) sistema eragilean oinarritu ziren. Inferno sistema eragile banatua da, eta makinen arteko komunikazioa egiteaz arduratzen da. Horrela, autoreen hitzetan, Cocytus Infernoren gainean eraikitako testuen prozesaketarako geruza sinple bat besterik ez da.

KOSHIK (Exner eta Nugues, 2014) testu eleanitzak prozesatzeko sistema banatua da. Prozesaketa eskalagarria ahalbidetzeko Hadoop erabiltzen du. KOSHIKen berezitasun bat da bere anotazio-eredu propioa diseinatu zaiola. Ereduak ez du aurreko faseetan sortutako anotazio-geruzak aldatzen uzten, geruza berriak gehitu daitezke soilik. Horrek prozesaketa paraleloa bideratzen laguntzen du, modu horretan errazagoa baita geruzen arteko dependentziak ekiditea. Erturk eta Shik (2016) KOSHIK erabiltzen dute hizkuntzaren prozesamendurako, eta haren inguruko balorazio bat egiten dute, alde onak eta txarrak azpimarratuz.

Gamallo *et al.*ek (2014) gaztelaniazko testuak prozesatzeko hainbat modulu aurkezten dituzte, arreta berezia emanez analisi morfosintaktikoa eta entitate-izenen sailkapena egiteari. Moduluek *pipeline* eredua jarraitzen dute, baten irteera hurrengoaren sarrera izan dadin, tartean inolako datu-fitxategirik

[18] `https://uima.apache.org/doc-uimaas-what.html` (kontsulta: 2017-05-08)
[19] `https://uima.apache.org/doc-uimaducc-whatitam.html` (kontsulta: 2017-05-08)

sortu gabe. Prozesaketa web-eskalan egin dadin, moduluei paralelizaziorako teknikak aplikatu zaizkie Hadoop erabiliz. Moduluak oinarri-oinarrizko hizkuntzaren prozesamenduko teknikak erabiliz garatu dituzte, ingurune banatuetara ahal bezain ongi egokitzearren.

Sun eta Gaok (2017) egiturarik gabeko datuentzako prozesaketa- eta biltegiratze-sistema banatu eta eskalagarria aurkezten dute. Lan honen bereizgarri nagusia biltegiratze-sistema eskalagarrian datza. Izan ere, gaur egungo prozesaketa banaturako sistemek memoria dinamikoarekin egiten dute lan gero eta gehiago. Hala ere, lehenago edo beranduago, sortutako datuak biltegiratu eta atzitu egingo dira, eta garrantzitsua da fase hori ere eskalagarria izatea.

Orain arte azaldutako lanik gehienak batch-prozesaketa egiten dute, eta Hadoop sistema darabilte eskalagarritasuna lortzeko. Lin eta Dyer-ek (2010) MapReduce ereduaren arabera pentsatzen ikasten laguntzen dute, beti ere hizkuntzaren prozesamendua oinarri hartuz. Besteak beste, MapReduce aplikazioak garatzeko eredu eta patroi ohikoenak azaltzen dituzte. Bestalde, Andersson-ek (2016), hizkuntzaren prozesamendurako Spark eta Hadoop sistemak konparatzen ditu, arlo horretarako bakoitzak eskaintzen dituen abantailak eta desabantailak azalduz.

Streaming-prozesaketan oinarritutako lanak ere egin dira azken urteetan. Dena *et al.*ek (2013) Stormen oinarritutako sistema bat aurkezten dute, testuen prozesaketa eskalagarria egiteko. Besteak beste, baliabide fisikoen kudeaketari arreta berezia eskaintzen diote, eta horretarako Apache Mesos[20] erabiltzen dute. Hermes (Paris eta Sabena, 2016) ere testuak streaming moduan prozesatzeko beste sistema bat da, baina honek batch-prozesaketa egiteko aukera ere ematen du. Testuak prozesatzeaz gain, biltegiratzen eta kontsultatzen ere laguntzen du.

Hodei-zerbitzuak ere gero eta zabalduago daude hizkuntzaren prozesamenduaren arloan. ILLINOISCLOUDNLP (Wu *et al.*, 2014), esaterako, testu-dokumentuak hodeiean prozesatzeko zerbitzua da. Hainbat HP daude zerbitzuan bertan integratuta, eta horien artean aukera dezake erabiltzaileak

[20] `https://mesos.apache.org` (kontsulta: 2017-05-08)

bere analisi-katea eraikitzeko. HPak norberaren datuekin entrenatzeko aukera ere ematen du, beti ere ILLINOISCLOUDNLP plataforma erabiltzen bada horretarako. Amazon Web Zerbitzuetako Elastic Cloud Computing (EC2) erabiltzen dute oinarri-teknologia gisa. Antzeko beste hodei-zerbitzu bat TextServer (Padró eta Turmo, 2015) da. Hainbat hizkuntzatarako HPak eskaintzen ditu honek, eta batch eta streaming moduko prozesaketa-ereduetara egoki daiteke. Web-interfaze bat eskaintzen du prozesatu beharreko dokumentuak modu erosoan bidali eta jasotzeko. Hodei-zerbitzuekin jarraituz, Yu eta Chen-ek (2013) informazio semantikoaren prozesaketa masiboa hodeian egin ahal izateko teknologien inguruko arloaren egoera lantzen dute.

Prozesaketa banatua hizkuntzaren prozesamenduko beste hainbat atazatan ere landu izan da azken urteetan zehar; esate baterako, galdera-erantzuneko sistemetan. Gaur egun hainbeste erabiltzen diren prozesamendu banaturako Hadoop eta Storm bezalako tresnarik oraindik ez zegoenean, Sonntag-ek (2004) galdera-erantzuneko sistementzako algoritmo banatuak landu zituen JavaSpace teknologia erabiliz. Galdera-erantzuneko sistemek erantzun egokiak bilatzeko ataza prozesamendu aldetik hain garestia izanik, ingurune banatuetan exekutatzeko beharra azpimarratzen zuen dagoeneko. Epstein *et al.*ek (2012) inoiz izan den galdera-erantzuneko sistemarik ezagunena den Watsonen kasuan eskalagarritasuna nola lortu zuten azaltzen dute. Watsonek, hasiera batean, ordubetetik gora behar zuen batez beste galderei erantzuteko. 2011 urtean, ordea, Jeopardy![21] telebistako lehiaketan hartu zuen parte, gizakien aurka lehiatuz, eta erantzun-denbora izugarri hobetu behar izan zuten horretarako. UIMA-AS erabiliz sistema paralelizatu eta milaka PUZez osatutako ingurune banatuan ezarriz, erantzunak batez bestean 3 segundoko atzerapenarekin ematea lortu zuten. Bestalde, Feng *et al.*-ek ere (2016) galdera-erantzuneko sistemetarako prozesaketa banatuaren beharra azpimarratzen dute. Haien esanetan, 48 nodo langileko ingurune batean erantzunak 24 aldiz azkartzea lortu dute.

Informazioaren erauzketaren arloan ere prozesaketa banatuaren beharra azpimarratu dute autore askok. Nesi *et al.*ek (2015) webean zehar hitz-gakoen bilaketa egiteko sistema banatu bat aurkezten dute, GATE plataforman eta

[21] `https://www.jeopardy.com` (kontsulta: 2017-05-08)

HDFS fitxategi-sisteman oinarritua. Mittal *et al.*ek (2015), beren aldetik, lege-dokumentuen artean bilaketa egin eta informazio esanguratsua eraueteko sistema aurkezten dute, bereziki hodei-zerbitzuen Zerbitzu Mailako Akordioen dokumentuetan oinarrituz. Gómez-Pérez *et al.*ek (2016), era berean, nekazaritza-testuetatik informazio baliagarria automatikoki eraueteko sistema aurkezten dute, eskalagarritasuna lortzeko Hadoop erabiliz. Lee *et al.*ek (2013) ere Hadoop erabiltzen dute *E-discovery* delakoa modu eskalagarrian egiteko. *E-discovery* deitzen zaio prozesu legaletan bilaketak formatu digitalean dagoen informazioan egiteari. Twitter-etik informazioa erauzten duten sistema gehienek ere eskalagarritasuna kontuan hartu behar izaten dute. Adibidez, Twittereko mezuetatik gertakarien informazioa denbora errealean erauzten duen sistema aurkezten dute McCreadie *et al.*ek (2013). Streaming ereduari bete-betean egokitzen zaion kasua izanik, Storm erabiltzen dute prozesaketa banatua kudeatzeko.

Dokumentuak web-eskalan multzokatzeko ere egin dira lanak. Adibidez, Hernandez eta Garciak (2016), kosinu-antzekotasunean oinarritutako algoritmo bat garatu dute Hadoop erabiliz. Algoritmoak, pelikulen inguruko balorazio eta kritika sorta bat jasota, pelikulak antzekotasunaren arabera multzokatzen ditu.

Era berean, antzeko prozesaketa-sistema eskalagarriak hizkuntzaren prozesamenduko beste hainbat atazatan ere erabili dira, besteak beste, dokumentuen sailkapenean (Semberecki eta Maciejewski, 2016), bilatzaile semantikoetan (Derivière *et al.*, 2006) edota ikasketa automatikoan (Ravi eta Diao, 2016). Azken lan horretan, bereziki, streaming eredua erabiltzen dute ikasketa automatikorako metodo erdi-gainbegiratuak modu eskalagarrian eta banatuan egiteko.

6. KAPITULUA

Hizkuntzaren prozesamendu masiborako arkitektura bat

6.1. Sarrera eta motibazioa

Edozein esparrutako profesionalek jakintza zabala eta zehatza behar dute erabaki egokienak hartzeko. Gaur egun eskuragarri dagoen informazioaren tamaina erraldoia dela eta, interesgarria den informazio guztia teknologiaz baliatu gabe aurkitu eta prozesatzea ia ezinezkoa da gizakiarentzat. Hori dela eta, hain baliagarria den informazioa zaharkituta gelditzen ari da etengabe, eta horrek profesionalen erabakietan eragin zuzena dauka. Beraz, edozein berrikuntza edo aldaketaren aurrean behar bezala erantzun ahal izateko informazio eguneratua eskuratzeko lehia izugarria da, egungo ia edozein sektoretan. Zeregin horretan lagunduko duen teknologia behar-beharrezkoa da arlo horretan aurrerapauso berriak emateko, eta, arazoa oraindik ebatzita egotetik urrun dagoenez, azken urteetan gorakada nabarmena izaten ari da datu handien tekniken esparrua, honek informazio kopuru handiak ahalik eta denbora-tarterik txikienean prozesatzea baitu helburu.

Hizkuntzaren prozesamenduaren arloan ere bete-betean eragiten du aipatutakoak. Izan ere, informazio gehiena egituratu gabeko dokumentu gisa (webguneak, egunkariak etab.) aurkitzen denez, milioika dokumentu prozesatu behar dira horietatik interesatzen zaigun informazioa erauzteko. Newsreader proiektuan, esaterako, gertakarien ezagutzea egitea da helburua, hizkuntza anitzetan, gertakari bakoitza noiz, non eta nori gertatu zaion erauziz, eta

gertakari berberari dagozkion testuak erlazionatuz, besteak beste. Proiektuan, egunean dozenaka mila dokumentu analisi konplexuen bidez eta ordu gutxiren buruan prozesatzeko ahalmena duen sistema garatu da. Testuen prozesaketaren eskalagarritasunak berebiziko garrantzia dauka proiektuan, beraz.

Hizkuntzaren prozesamenduan egiten diren prozesaketa astunenak, gehienetan, analisi-kate batekin testu-dokumentu kopuru erraldoiak prozesatu behar direnean gertatzen dira. Corpusak milioika dokumentuz osa daitezke, eta horietako bakoitza analisi-kateko hizkuntza-prozesatzaile (HP) bakoitzak prozesatzen du, lortutako irteera kateko hurrengo HPari bidaliz. Hori horrela izanik, dokumentu bakoitza prozesatzeko minutuak behar izan daitezke, eta dokumentu asko prozesatu behar diren kasuetan oso litekeena da denbora-arazoak izatea.

Prozesaketa-denborak murrizteko modu bat konputagailu azkarragoak erostea da, eskalagarritasun bertikalaren alde eginez. Baina soluzio hori, garestia izateaz gain, oso mugatua da, konputagailu bakarrarekin lor daitekeen prozesaketa-ahalmena bera ere oso mugatua baita. Soluzio egokiena, egungo egoeran, prozesaketa hainbat konputagailutan banatuta exekutatzea da, ingurune banatuetan, alegia. Modu horretan eskalagarritasun horizontala lor daiteke, bertikala baino askoz ere merkeagoa eta ahaltsuagoa dena. Horrelako ingurune batean eraginkorrena, beharbada, analisi-kateko HP bakoitza MapReduce bezalako paradigma ezagunaren arabera berrinplementatzea litzateke, HP bakoitzaren exekuzioa bera barrutik paralelizatuz. Hala ere, prozesatzaile bakoitza berrinplementatzea izugarrizko lana litzateke, eta ez oso hedagarria gainera, kateari prozesatzaile berri bat gehitzea lan nekeza bihurtuko bailitzateke. Irtenbide hori baztertuz, edozein analisi-kate ingurune banatu batean ezarri eta prozesaketa paraleloa ahalbidetuko duen sistema bat aurkeztuko dugu kapitulu honetan. Ingurune horretan, HPak paraleloan exekutatuko dira, dokumentu multzo handien prozesaketak modu esanguratsuan azkartuz.

Sistemak murriztapen bakarra ezartzen die prozesatzaileei: testua eta anotazio linguistikoak NAF eredua (ikus 3.2. kapitulua) jarraituz adieraztea. Baldintza hori betetzen duen edozein prozesatzaile integra daiteke hemen

aurkeztu eta eskaintzen dugun arkitekturan, inolako egokitzapenik egin behar izan gabe.

Antzeko sistemak konputagailu anitzez osatutako terminal taldeetan ezarri eta martxan jartzeko prozesua astuna izaten da askotan. Hori saihesteko, makina birtualetan (MB) oinarritu dugu gure sistema. Publikoki eta lizentzia libreekin banatu ditugu MBak automatikoki sortu eta ingurune banatuetan ezartzen dituzten scriptak, sistema hutsetik hasita martxan jartzea edonorentzat lan eroso eta azkarra izan dadin.

Kapitulu hau honela egituratu dugu: Lehenik eta behin, hizkuntzaren prozesamendurako analisi-kateak osatzeko tresna eta prozesatzaile ezagunenak aztertuko ditugu, bereziki gure sisteman integratu ditugunak. Ondoren, hizkuntzaren prozesamendu masiborako diseinatu eta inplementatu dugun arkitektura aurkeztuko dugu. Arkitektura horren kasuan, diseinua xehetasunez azaltzeaz gain, sistemaren erabilgarritasuna eta eraginkortasuna neurtzeko egindako hainbat esperimentu aurkeztuko ditugu, haien emaitzekin batera.

6.2. Hizkuntzaren prozesamendurako analisi--kateak

Gure sistema NAF dokumentuak jaso eta sortzen dituen edozein HP erabiltzeko gai da. Edonork bere HP propioak gehi ditzakeen arren, sisteman IXA pipes tresnak (Agerri *et al.*, 2014b) integratu ditugu lehenetsi gisa. Sisteman integratutako tresnen zerrenda osoa 143. orrialdeko 6.1 taulan ikus daiteke.

IXA pipes-eko HPek honako anotazioak sortzeko gaitasuna dute, euskara, ingeles eta gaztelaniarako: tokenizazioa (taulako TOK HPa), esaldien zatiketa (TOK), kategoria-etiketatzea (POS), lematizazioa (POS), entitate-izenen ezagutzea eta sailkatzea (NERC) eta osagaietan oinarritutako sintaxia (Parse). Gaztelania eta ingeleserako korreferentzia-ebazpena (Coref) ere egiten dute. IXA pipes tresnak kanpoko beste HP batzuekin hedatu dituzte, analisi-kate osatuagoa lortzeko, eta guk ere sartu ditugu gure sisteman. HP horiek NewsReader proiektuaren baitan garatu dituzte, eta ingeleserako honako anotazioak egiteko gaitasuna gehitzen diete IXA pipes tresnei: denbora-espresioen ezagutzea (time, tempRel), hitzen adiera-desanbiguazioa

(WSD), entitate-izenen desanbiguazioa (NED), rol semantikoen etiketatzea (SRL), faktualtasunaren ezagutzea (Fact), sentimendu-analisia (Opinion) eta gertakarien korreferentzia-ebazpena (eCoref).

IXA pipes tresnak HP sinple eta erabilerrazak, eramangarriak, modularrak, eraginkorrak, zehatzak eta lizentzia librearekin banatzeko helburuarekin garatu zituzten. Unix sistema eragileen antzera, IXA pipes-eko HPek ere irteera eta sarrera estandarren bidez lotutako prozesu multzoa osatzen dute. Hau da, HPek irteera estandarrean idazten dute beren emaitza, eta edukia, zuzenean, kateko hurrengo HPari iristen zaio sarrera estandarretik. Datuetan oinarritutako arkitektura horri esker, edozein HP sarrera eta irteerako datuen formatu egokia darabilen beste edozeinekin ordezka daiteke. IXA pipes, oinarrian, edozein instalazio- edo ezarpen-denbora eta esfortzu minimizatzeko diseinatuta dago, eta Apache 2.0 lizentziapean banatzen da. Bestalde, HP guztiek NAF formatuan (ikus 3.2. kapitulua) ematen dituzte beren emaitzak.

6.3. Sistemaren arkitektura

Testuen prozesaketarako gure sistema banatua hainbat makinaz (nodoz) osatzen da. Nodoetako batek sistema osoaren kontrolatzaile-lanak egiten ditu (nodo nagusia), eta gainontzeko guztiek testuen prozesaketa gauzatzen dute (nodo langileak). Dokumentu bakoitza hainbat nodotatik pasatzen da prozesaketan zehar, analisi-kateko faseak hainbat nodotan exekuta baitaitezke paraleloan.

Hauek dira inplementatutako arkitekturaren ezaugarri nagusiak:

- Prozesaketa modu banatuan egiten da, hau da, analisi-kateko pauso bakoitza hainbat nodotan exekuta daiteke. Bai HPak eta bai dokumentuak automatikoki banatzen dira sisteman zehar.
- Dokumentuak edozein unetan irits daitezke sistemara, eta sistema libre egon bezain laster hasten da prozesaketa.
- Sarrera-puntu bakarra dauka sistemak. Dokumentu guztiak, prozesatzera bidaltzean, nodo nagusian kokatzen den sarrerako ilaran gelditzen dira, sistema noiz libratuko zain.

- Era berean, prozesatutako dokumentu guztiak nodo nagusian kokatutako irteerako ilarara bidaltzen dira. Ondoren, beste prozesu batek jasotzen ditu dokumentuak ilaratik eta fitxategi-sistemara pasatzen ditu.

- HPak eta haien arteko dependentziak zehaztuz eraikitako grafoen bidez adierazten dira analisi-kateak. Grafo horiei topologia deritzegu, eta modu deklaratiboan definitzen dira. Horrela, topologiak moldatzea eta berriak sortzea askoz ere erosoagoa eta errazagoa da.

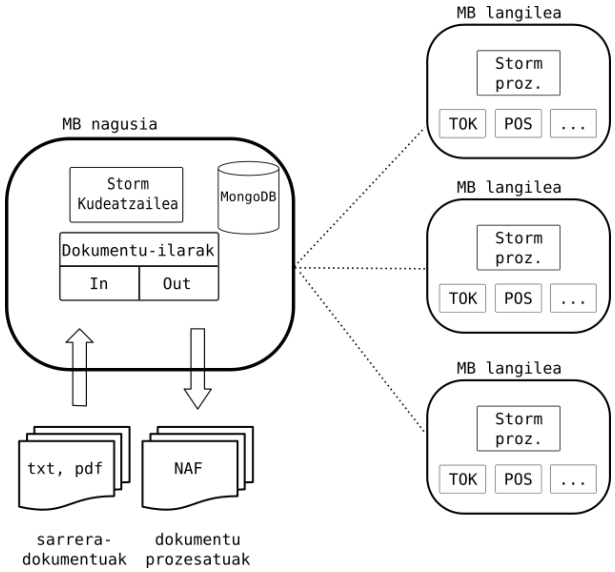
- Topologiak linealak edo ez-linealak izan daitezke. Linealak ez direnean, elkarrekiko dependentziarik ez duten HPak paraleloan exekutatuko dira.

- Sistema oso eskalagarria da. Nodo berriak gehitzea berehalakoa da, eta sistema automatikoki orekatzen da konputazio-ahalmen berrira egokituz.

Exekuzio paralelo eraginkorra ahalbidetzeko, esan bezala, nodo ugaritan banatzen da analisi-katea. Horretarako, makina birtualen (MB) aldeko apustua egin dugu. Izan ere, MBak oso erabiliak dira gaur egun hodei-konputazioan, dependentzia- eta instalazio-arazoak ekiditen laguntzen baitute, erabiltzeko prest dauden makinak zerbitzarietan ezartzea erraztuz. Horrela, HP eta bestelako tresna guztiak instalatuta dauzkaten MBen kopiak automatikoki sortzen ditugu, edozein makinatan muntatu eta erabiltzeko prest.

Hizkuntza-prozesatzaileak terminal taldeetan modu orekatuan banatu eta dokumentuak modu banatuan prozesatzeko Apache Storm erabiltzen dugu (ikus 5.3.8 atala). Stormerako diseinatutako topologian, sisteman parte hartuko duten HPak eta haien arteko dependentziak adierazten dira, eta Storm horretaz baliatzen da HPen banaketa eraginkor eta orekatua modu automatikoan egiteko. HP bakoitzaren hainbat instantzia egon daiteke, beti ere erabilgarri daukagun konputazio-ahalmenaren arabera.

6.1 irudian arkitekturaren diseinu orokorra ikus daiteke. Sistema nodo nagusi batez eta hainbat nodo langilez osatzen da. Jarraian, nodo mota horiek biak aztertuko ditugu.



6.1 irudia: Sistemaren arkitektura adierazten duen irudia. Sistema hainbat nodoz osatzen da, nodo nagusi bat eta hainbat nodo langile. Nodo nagusia sistemaren kudeaketaz arduratzen da, eta nodo langileak prozesaketa gauzatzeaz.

6.3.1. Nodo nagusia

Nodo nagusiak sistema kudeatzen du, nodo guztien arteko zubi-lanak eginez. Hauek dira nodo mota honen zereginik garrantzitsuenak:

- Stormen prozesu guztiak kudeatzeaz arduratzen da.
- Dokumentuen sarrera eta irteerako ilarak bertan kokatzen dira.
- Dokumentuen prozesaketa partzialak biltzen dituen MongoDB datu-basea bertan kokatzen da.

Stormek prozesuak nodo guztietan sortzen dituenez, prozesu nagusi batek, zeinek gainontzeko prozesuak kudeatuko baititu, martxan egon behar du beti. *Nimbus* deituriko prozesu hau nodo nagusian exekutatzen da. Nodo nagusiak erabakitzen du dokumentu bakoitza, uneoro, zein nodotan prozesatuko den. Gainera, analisi-kateko urrats bakoitza hainbat nodotan exekuta daitekeenez,

dokumentu bakoitzak zenbait nodotan zehar bidaiatzen du, nodo nagusiak aginduko baitio, analisi-kateko urrats bakoitzean, zein nodotara jo.

Bestalde, oinarrian streaming eredua jarraitzen denez, sistema etengabe dago dokumentu berriak noiz iritsiko zain. Horretarako, nodo nagusian dokumentuen sarrerako ilara bat eta bakarra dago. Dokumentuak, iritsi ahala, ilara horretan sartzen dira. Aldi berean, libre dauden nodo langileak etengabe ari dira sarrerako ilaran dokumentu berriren bat ote dagoen aztertzen, eta aurkitzean, dokumentua jaso eta prozesatzen hasten dira. Era berean, dokumentu bat prozesatzen bukatzean, nodo nagusiko irteera-ilarara bidaltzen da dokumentua. Nodo nagusian badago beste prozesu bat, irteera-ilaran dokumentu bat antzeman ahala, haren edukiarekin erabiltzaileak aukeratutako direktorioan NAF fitxategi bat sortzen duena. Horrela, beraz, terminal taldean prozesatutako dokumentu guztiak nodo nagusiko direktorio jakin batean aurki ditzake erabiltzaileak. Dokumentuen ilarak inplementatzeko Kafka[1] eta RabbitMQ[2] sistemak erabili ditugu.

Azkenik, analisi-katearen urrats bakoitzean sortutako anotazio linguistiko guztiak etengabe nodo batetik bestera garraiatzen ez ibiltzeko, emaitza partzialak MongoDB datu-base batean gordetzen dira. Datu-base hori nodo nagusian kokatzen da, eta nodo horrek kudeatzen du datu-basea bera ere. 6.3.4. atalean azalduko ditugu datu-basearen nondik norako guztiak.

6.3.2. Nodo langileak

Dokumentuen prozesaketa nodo langileetan gauzatzen da. terminal taldean hainbat nodo langile ezartzeak ahalbidetzen du sistemaren eskalagarritasuna, lana nodo horien artean banatzen baita. Nodo langile bakoitza MB bat da, HP guztiak, beharrezko dependentzia guztiekin batera, instalatuta eta erabiltzeko prest dauzkana.

MBa sortzeko unean, HPak eta bestelako eduki guztia nodo nagusitik ekartzen dira. Sinkronizazioa ez da sorkuntza-garaian bakarrik egiten, etengabe mantentzen baitira eduki jakin batzuk nodo nagusiaren eta langileen artean

[1] `https://kafka.apache.org` (kontsulta: 2017-05-08)
[2] `https://www.rabbitmq.com` (kontsulta: 2017-05-08)

sinkronizatuta. Horri esker, asko errazten da, nodoak sortu ondoren ere, edukiak nodo guztietara eramatea. Demagun, esaterako, HP berri bat sartu dugula analisi-katean, eta dagoeneko ehundik gora nodoko terminal talde bat daukagula erabilgarri. HPa nodo guztietan, banan-banan, eskuz instalatzea izugarrizko lana litzateke. Hori saihesteko, ezarri dugun sinkronizazio-sistemari esker, HPa nodo nagusian besterik ez genuke eskuz instalatu beharko, eta, ondoren, eduki berri hori nodo langile guztietan kopiatzeko agindu. Sistema hau ez da paketeen instalaziora bakarrik mugatzen, konfigurazioak eta bestelako ezarpen konplexuak ere sinkroniza baititzake. Sinkronizazio-sistema ezartzeko Puppet[3] erabili dugu.

HP bakoitza Stormen bolt baten barruan biltzen da, Storm geruza bat ezarriz HParen gainean. Horrela, HP bakoitza Stormen osagai baten bidez abstraitzen da, terminal taldean zehar haien instantziak banatzea eta elkar komunikatzea Stormen esku geldituz. HPei jartzen diegun murriztapen bakarra da sarrera- eta irteera-datuak NAF formatuan jaso eta bidali behar izatea, sistemaren sarrera eta irteera estandarretatik, hurrenez hurren. Inplementazio-lengoaia, exekutatzeko modua, beharrezko ezarpenak etab. edozein izan daitezke.

Besterik ezean, HPak, iristen zaien dokumentu berri bakoitzarekin hutsetik exekutatzen dira, HPa hasieratu, prozesaketa gauzatu eta prozesua bukatuz. Modu horrek sistemaren erabilerraztasunari egiten dio mesede, exekuzioak ez baitu testuingururik behar. Hau da, dokumentu berri bat iristean, HParen exekutagarriari dei egingo zaio dokumentua sarrera estandarretik bidaliz, besterik gabe. Hala ere, zenbaitetan, HParen hasieratze-denbora gehiegizkoa da behin eta berriz hasieratu behar izateko, sistemaren eraginkortasunari bete-betean eragitea iristeraino. Hori dela eta, hasieratzen denbora gehien behar duten HPak bezero-zerbitzari eredua jarraituz inplementatu eta exekutatzen ditugu. Horrela, HPa behin hasieratzen da, eta memorian kargatuta gelditzen da dokumentu berriak iristeko zain, iritsi ahala prozesatuz. Eredu honen desabantaila nagusia memoriaren erabilera da, zenbait HPk behar duten memoria kantitatea oso handia baita. Esaterako, IXA pipes tresnaren entitate-izenen desanbiguatzaileak 10 GB inguru hartzen ditu hasieratu

[3] `https://puppet.com` (kontsulta: 2017-05-08)

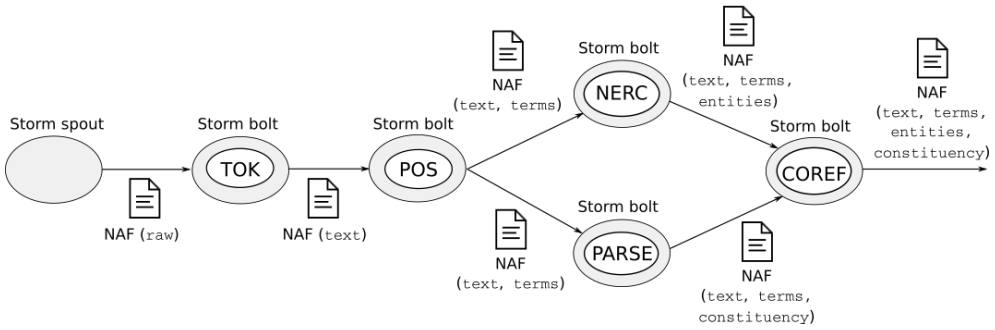
ondoren. Memoriaren arazoaren eragina murrizteko, nodo langile dedikatuak sortzeko aukera gehitu diogu sistemari. Horrela, HP jakin bat bakarrik ezar daiteke nodo langile batean, nodoaren memoria osoa harentzat utziz; hori aproposa da memoria asko behar duten HPak bezero-zerbitzari bezala exekutatzeko. Nodo dedikatuak topologiarekin batera definitzen dira, modu deklaratiboan (ikus 6.3.5 atala). Sistema bera arduratzen da, topologia terminal talde osoan zehar banatu behar duenean, nodo dedikatuei dagozkien HPak bakarrik esleitzeaz.

6.3.3. Datuen fluxua nodoetan zehar

5.3.8. atalean azaldu dugu Storm aplikazioak garatzeko prozesu osoa spout eta boltetan zatitu behar dela, eta horiek elkarren artean modu egokian lotu, topologiak osatuz. Sistema osoa Stormen gainean inplementatu dugunez, spout eta bolt egokiak identifikatu behar izan ditugu, eta topologia osatu. Gure kasuan, HP bakoitza bolt batean bildu dugu, eta spout bakarra dago, dokumentuak sarrera-ilaratik eskuratzen dituena. Bolten arteko loturek beren arteko dependentziak adierazten dituzte, eta bi HPren arteko lotura dagoenean, lehenbizikoaren irteera bigarrenak jasotzen du sarrera gisa. Bolten artean partekatzen diren datuak, berriz, sarrerako testu-dokumentuei dagozkien partzialki anotatutako NAF dokumentuak dira.

Analisi-kateko HPak nodo langileen artean banatuta daudenez, dokumentuek nodoen artean bidaiatzen dute. Dokumentu baten prozesaketa osoa analisi-kateko HPetan zatitzen da. Beraz, prozesaketa-unitate txikiena HP baten exekuzioa dela esan daiteke. HP bat exekutatu, haren irteera analisi-kateko hurrengo HPa daukan nodoari bidali, nodo horretan dagokion HPa exekutatu, jasotako datuak sarrera estandarretik pasatuz, eta prozesu hori guztia analisi-kateko HP bakoitzarekin errepikatzea zeregin konplexua da. Storm lan horren zati handi bat egiteko gauza denez, horri etekina atera diogu. Horretarako, Stormek ezagutza jakin bat behar du: HPen topologia.

Topologiaren ezagutzarekin, Stormek HPen egitura ezagutzen du, eta, ondorioz, gure aplikazioaren datuen fluxua ere bai. Horrela, Stormek badaki HP baten emaitza zein beste HPri bidali behar dion. Izan ere, batzuetan, HPen arteko lotura ez da bakuna, topologia ez-linealak eraiki baitaitezke. Horre-



6.2 irudia: Datuen fluxua horrelakoa litzateke datu-baserik erabiliko ez bagenu. HP bakoitzak, berak sortutako geruzaz gain, jasotakoak ere bidaliko lizkieke hurrengoei, bestela bidean galduko lirateke eta.

lakoetan, baliteke HP baten emaitza beste hainbat HPri aldi berean bidali behar izatea, HP horiek paraleloan exekuta daitezen. Ondoren, paraleloan exekutatu diren HPen emaitzak beste HP batek jasoko ditu, eta haren esku geldituko da jasotako datu guztiak elkartu eta NAF bakarra eraikitzea.

Datuei dagokienez, lehenik eta behin, azalduko dugu datu-baserik erabiliko ez bagenu bolten artean zein datu mota partekatuko genituzkeen. Izatez, anotazio linguistikoz hornitutako testu-dokumentuak bidaliko genituzke bolt batetik bestera: NAF dokumentuak. NAF dokumentuak XML dokumentuak direnez, string batean bidaltzen da dokumentu bakoitza.

6.2 irudiko adibidean ikus daiteke orain arte azaldutakoaren arabera datuen fluxua nolakoa litzatekeen.

Dena dela, jarraian azalduko dugun bezala, badago datuen bidalketa modu eraginkorrago batean egiterik. Izan ere, NAF dokumentuak oso handiak izan daitezke gehitu zaizkien anotazio-geruzen arabera. Gainera, NAF dokumentua hazten doa HP batetik bestera, HP bakoitzak anotazio-geruza berri bat gehitzen baitio dokumentuari. Hori horrela, datuen transmisioa oso garestia izan daiteke analisi-katean aurrera egin ahala, beti ere jatorrizko dokumentuen tamainaren edo analisi-kateko HP kopuruaren arabera. Alabaina, HP guztiek ez dituzte zertan aurreko HP guztiek sortutako analisi guztiak jaso beren lana behar bezala egiteko. Izatez, HP bakoitzak HP gutxi batzuekiko dependentzia izan ohi du, eta horiek sortutako anotazio-geruzak besterik

ez ditu erabiltzen. Esate baterako, rol semantikoen etiketatzea egiten duen HPa analisi-katearen amaieran ager daiteke, eta bere aurretik beste HP asko izan, baina tokenak eta terminoak besterik erabiltzen ez dituenez, aski luke lehenbiziko bi HPen emaitzak jasotzearekin. Horregatik, HP bakoitzak sortutako anotazio-geruza, topologiako hurrengo HPei zuzenean bidali ordez, datu-base zentral batean gordetzen da, eta hurrengo HPak, anotazio-geruza guztiak jaso ordez, datu-basetik beharrezko dituenak besterik ez ditu jasotzen. Horretarako, HP bakoitzak zein anotazio-geruza irakurriko dituen eta zein sortuko dituen zehaztu behar da. 6.3.5. atalean azalduko dugu HPen arteko dependentziak nola adierazten diren.

Soluzio horrek sistemaren datu-trafikoa murrizten du, baina baita datuen fluxuari konplexutasuna gehitu ere. Izan ere, lehen, Stormek tuplatan bidaltzen zituen, bolt batetik bestera, datu guztiak. Orain, ordea, Stormek ez ditu garraiatuko prozesaketan erabiliko diren datuak. Aldiz, Stormen tupletan unean uneko NAF dokumentuaren identifikadorea da bidaliko den datu bakarra, eta HP bakoitza barruan hartzen duen boltaren esku gelditzen da, identifikadore horretaz baliatuz, behar dituen NAF dokumentuaren geruzak datu-basetik jasotzea. Era berean, sortutako anotazio-geruza datu-basean idazteaz ere bolta bera arduratuko da. 6.3 irudian, adibide baten bidez ikus daiteke azaldutakoaren irudikapena.

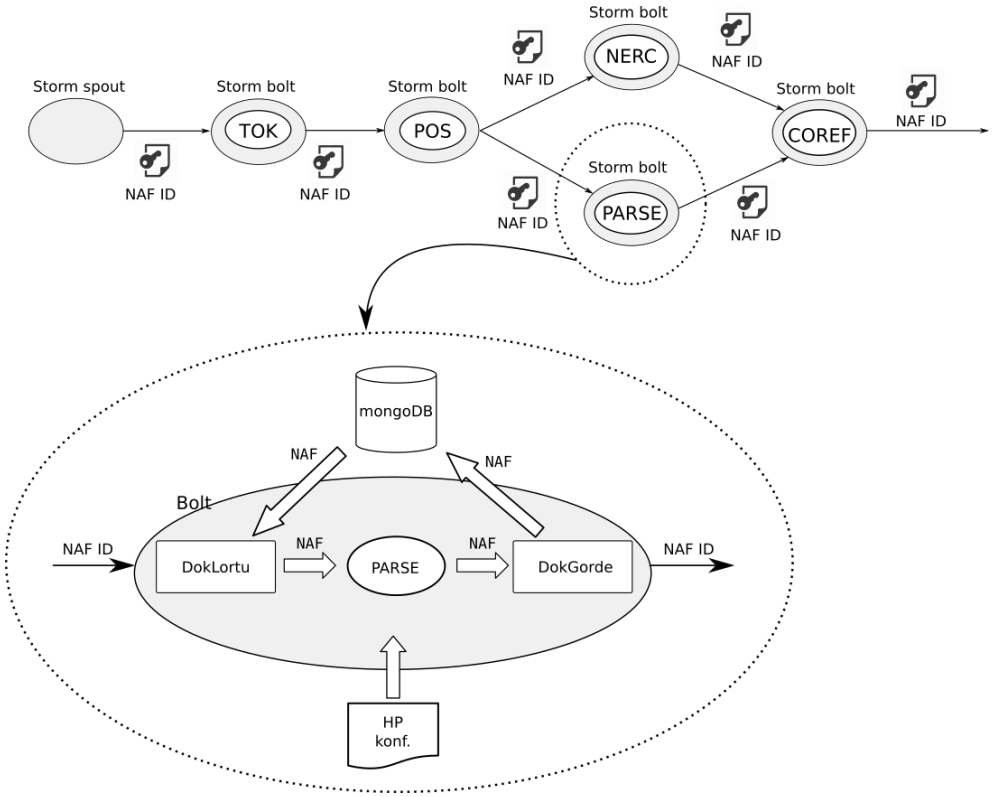
Laburbilduz, bolt bakoitzak honako pauso hauek jarraituko ditu tupla bat jasotzean:

1. Tuplatik NAF dokumentuaren identifikadorea jaso.

2. Jasotako identifikadorearekin, eta HParen espezifikazioak ezagututa, NAF dokumentu horretatik behar dituen anotazio-geruzak bakarrik eskuratu.

3. Kargatutako NAF dokumentua sarrera estandarretik bidaliz, bolt horri dagokion HPa exekutatu.

4. Amaitzean, irteerako NAF dokumentutik berria den anotazio-geruza erauzi eta datu-basean gorde.



6.3 irudia: Datuen fluxua, MongoDB datu-basea gehitu ondoren. HP batetik bestera bidaltzen den bakarra NAF dokumentuaren identifikadorea da. Horrekin, HPek datu-basetik eskuratzen dituzte beharrezko dituzten NAF geruzak. HP bakoitzak zein geruza behar dituen kanpoko fitxategi batetik (HP konf.) irakurtzen du.

5. Dokumentuaren identifikadorea hurrengo HPei bidali.

Modu horretan, sareko trafikoa murriztu dugu. Azter ditzagun berriro 6.2. eta 6.3 irudietako adibideak. Lehenbiziko kasuan, datu-baserik erabili gabe, HP bakoitzak sortutako geruzez gain, jasotako guztiak ere bidali behar dizkie hurrengoei, bestela bidean galduko bailirateke. Modu horretan, adibidean, 16 geruza transmitituko lirateke sarean osotara. Aldiz, datu-basea erabiliz, datu-basetik beharrezko geruzak besterik ez ditu ekartzen bolt bakoitzak. Horrela, adibidean, 9 geruza besterik ez genituzke bidaliko. Adibideko anali-

si-kateak 4 HP besterik ez dituela kontuan harturik, onura askoz ere gehiago nabarituko litzateke analisi-kate errealaren kasuan (15 HP).

6.3.4. Datu-basearekiko integrazioa

6.3.3. atalean azaldu den bezala, NAF dokumentuen tamaina handia dela eta, terminal taldeko trafikoa murriztearren, partzialki prozesatutako dokumentuen anotazioak datu-base zentral batean gordetzen dira. Horrela, anotazioak behar direnean bakarrik eskuratzen dira, behin eta berriz nodo batetik bestera bidali behar izan gabe. Datu-baseko eragiketek sistema moteldu ez dezaten, gehien errepikatzen diren eragiketak modu eraginkorrean egin behar dira. Gure arkitekturaren ezaugarriak kontuan hartuta, MongoDB datu-baseen kudeaketa-sistema (DBKS) aukeratu dugu. Atal honetan, MongoDB datu-basea gure arkitekturan nola integratu dugun zehaztuko dugu.

Tesi-lan honetan aurkeztutako testuen prozesaketa masiborako sisteman, etengabe bidaltzen dira testu anotatuak HP batetik bestera. HPak makina desberdinetan aurki daitezkeenez, testu anotatuen bidalketa garestia izan daiteke. Testu-dokumentu asko prozesatu behar direnean, trafiko handiegia sortu eta komunikazioa botila-lepo bihur daiteke. Hori saihesteko, MongoDB datu-base bat ezarri dugu terminal taldearen nodo nagusian.

Datu-basea ezarrita ere, nodo nagusian ezarriko denez, anotazioak behin eta berriz nodo nagusira bidali eta bertatik jaso beharko dira. Datu-basea ezartzeak abantaila nabarmen bat eskaintzen du, ordea. Izan ere, datu-base zentralik erabili ezean, analisi-kateko HP bakoitzak sortutako anotazioak ondorengo HP guzti-guztiei bidali behar zaizkie. Demagun A, B eta C HPek, hurrenkera horretan, osatzen dutela analisi-katea. Ak sortutako anotazioak Ck bakarrik erabili beharko balitu ere, Ak Bri bidali beharko lizkioke, eta Bk Cri. Datu-basea erabiliz, Ak datu-basean idatziko lituzke anotazioak, Bk ez lituzke jasoko, eta Ck zuzenean eskuratuko lituzke datu-basetik. Analisi-katea konplexua denean, trafikoa nabarmen murritz daiteke horrela.

HPen arteko dokumentu anotatuen bidalketa bakoitza datu-basearen bitartez egingo denez, beharrezkoa da datu-basearekiko eragiketak modurik eraginkorrenean egitea. Horretarako diseinatu dugun eredua aurkeztuko dugu atal

honetan.

Lehenik eta behin, maiz errepikatuko diren eragiketak zein diren aztertuko dugu:

- Anotazio-geruza osoak datu-basean gordetzea: HP batek bere lana bukatzean, jatorrizko testu-dokumentuari dagokion anotazio-geruza berri bat sortzen da gehienetan. Beste batzuetan, geruza berri bat sortu ordez, aurrez sortutako bateko anotazioak aldatzen dira. Soluzio ahalik eta orokor eta malguena lortzearren, aurrez existitzen den geruza aldatzen den kasuetan ere, geruza osorik idatziko da datu-basean, aurrez zegoena gainidatziz. Horrela, idazketa-eragiketak geruza osoak idaztera mugatuko dira. Kasuren batean geruza bat baino gehiago idatzi behar bada ere, geruza bakoitzeko idazketa-eragiketa bana egingo da.

- Anotazio-geruza osoak datu-basetik eskuratzea: HPek beste geruzetako anotazioak erabiltzen dituzte. Tokenizatzailearena da kasu bereziena, anotazioak erabili ordez, jatorrizko testua erabiltzen baitu. Hala ere, gure sistemaren anotazio-eredua NAF denez, jatorrizko testua ere anotazio-geruza bat bezala tratatzen da: `raw` geruza. Zenbaitetan, NAF dokumentua osorik eskuratu nahi izango da, baina, kasu horietan ere, geruza bakoitzeko kontsulta bat egitea erabaki dugu. Horrela, datu-basearen gainean egin beharreko kontsulta mota bakarra anotazio-geruza osoak eskuratzea da.

Eragiketa guztiak geruza mailakoak da. Ez da inoiz anotazio mailako eragiketarik egiten. Horretaz baliatuz, eredurik egokiena hurrengoa dela ondorioztatu dugu: testu-dokumentu bakoitzaren anotazio-geruza bakoitza MongoDBko dokumentu batean sartzea. Adibidez, tokenizatzaileak testu-dokumentu bat prozesatzen duenean, token guztiak dokumentu bakarrean bilduta sartuko dira datu-basean. Beste geruzen kasuan zertxobait konplikatzen da eredua. Izan ere, terminoak, adibidez, tokenez osatzen dira. Kasu horretan, tokenak dagoeneko datu-basean leudekeenez, testu-dokumentuari dagozkion termino guztiak ere beste dokumentu batean bilduta sartuko lirateke, baina termino horien tokenak berriro gorde ordez, tokenen erreferentziak gordez.

Era berean, entitateak terminoz osatzen direnez, entitateen geruza gordetzean, terminoen erreferentziak gordeko lirateke, terminoak dagoeneko datu-basean baileudeke. Horrela, eredu eraginkor bezain garbi bat lortu dugu. Jarraian, MongoDB datu-baseak diseinatzeko orduan kontuan hartu beharreko puntuak nola ebatzi ditugun aipatuko dugu:

- Kontsulten eraginkortasuna: Kontsulta guztiak geruza osoak eskuratzeko dira. Geruza bakoitzaren informazio espezifikoa dokumentu bakar batean bildu da, nahiz eta beheragoko mailetako geruzetarako erreferentziak eduki gehienetan. Edozein kasutan, geruza bat osoa eskuratzeko, kontsulta gutxi batzuk besterik ez da egin beharko. Zehazki, geruza baten anotazioak osatzeko erreferentziatu beharreko beheragoko mailetako geruza bakoitzeko kontsulta bat egin beharko da.

- Datuen erredundantzia: Anotazio konposatuak dauzkagunean, adibidez, entitateak, geruza horretako maila goreneko anotazioak besterik ez da gordetzen. Hau da, entitateak terminoz osatu arren, eta terminoak tokenez, entitateak gorde behar direnean, entitateen terminoak erreferentzia bidez gordetzen dira. Horrela, termino eta tokenak behin eta berriz geruza guztietan gordetzea ekiditen da. Beraz, diseinuari esker, datuen erredundantziaren arazoa saihesten da.

- Idazketen atomikotasuna: Aurrez esan dugu idazketak, dokumentu mailan, atomikoak direla MongoDBn. Gure ereduaren arabera, idazketak anotazio-geruzak gordetzera mugatzen dira. Geruza bakarra gorde behar denean, idazketa dokumentu bakarrean egingo da, beraz, modu atomikoan egingo da. Geruza bat baino gehiago aldi berean gorde behar direnean, kontuan izan behar da idazketa osoa ez dela atomikoa izango, eta arriskua dagoela, erroreren bat gertatuz gero, geruzetako batzuk bakarrik gordetzeko. Hala ere, gure sisteman integratu ditugun HP guztiek geruza bakarra idazten dute datu-basean.

- Dokumentuen hazkuntza: Dokumentuak behin datu-basean sartu eta gero aldatzen badira, arriskua dago esleituta duten disko-espazioa gainditu eta diskoan birkokatu behar izateko. Gure diseinuaren izaera dela

```
<topology>
    <cluster componentsBaseDir="/home/worker/components"/>
    <module name="EHU-tok" runPath="EHU-tok.v21/run.sh"
            input="raw" output="text"
            procTime="1"/>
    <module name="EHU-pos" runPath="EHU-pos.v21/run.sh"
            input="text" output="terms"
            procTime="2" source="EHU-tok"/>
    <module name="EHU-nerc" runPath="EHU-nerc.v21/run.sh"
            input="terms" output="entities"
            procTime="11" source="EHU-pos"/>
    <!-- ... -->
</topology>
```

6.4 irudia: Topologia zati baten definizioaren adibidea.

eta, MongoDBko dokumentuak ez dira inoiz aldatuko, geruzako anotazioak denak batera gordetzen baitira, eta, sortu eta gero, anotazioak ez baitira inoiz aldatzen. Izatez, HP batek dagoeneko esistitzen den geruza bat aldatu behar badu, geruza hori ezabatu eta bertsio berria geruza berri baten moduan gordeko du.

6.3.5. Topologien definizioa

Sisteman integratutako topologia 15 HPk osatzen dute. 6.1 taulan (143. orr.) ikus daitezke topologiaren xehetasunak. Zehazki, HP bakoitzaren izena eta oinarrizko deskribapen batekin batera, behar dituen sarrerako NAF geruzak eta sortzen dituenak adierazten dira.

Dokumentuak tesi-lan honetan aurkeztutako sisteman prozesatzeko, topologia definitu behar da lehenik eta behin. Nodo nagusian, lehenetsitako bi topologia aurki daitezke. Bata batch eredurako egokituta dago, eta horren arabera HP guztiak segidan exekutatzen dira, bata bestearen atzetik. Kasu horretan, paralelizazioa sarrerako dokumentu osoak makinen artean banatuz lortzen da. Aldiz, beste topologia streaming eredurako egokituta dago, eta bertan, HP bakoitza, behar dituen geruzak sortzen dituzten HPek bukatu bezain laster exekutatzen da, topologia ez-lineala lortuz.

Erabiltzaileak bere topologia propioak ere eraiki ditzake, lehenetsitakoez gain. Jarraian, topologiak nola definitzen diren azalduko dugu. Adibide

bezala, 6.4 irudian topologia baten zati baten definizioa ikus daiteke. Definizioa XML dokumentu batean gauzatzen da. Erro-elementua `<topology>` izanik, barruan `<cluster>` elementu bat eta hainbat `<module>` elementu izan ditzake. `<cluster>` elementuak HPak aurkitzen diren direktorioaren erroa adierazten du, ondoren, HPen definizioak errazteko. `<module>` elementu bakoitzak, berriz, topologiako HP bat definitzen du. HP bakoitzak honako atributu hauek izan ditzake:

- `name`: HParen izena.
- `runPath`: HParen exekutagarria zein helbidetan (*path*) aurkitzen den. Balio hau `<cluster>` elementuaren `componentsBaseDir` atributuan zehaztutako direktorioarekiko erlatiboa da.
- `input`: HPak zein NAF geruza behar dituen sarrera gisa.
- `output`: HPak zein NAF geruza sortzen dituen irteera gisa.
- `procTime`: HParen prozesatze-denbora erlatiboa.
- `source`: Analisi-katean HP honen aurretik aurkitzen den HPa (bat baino gehiago izan daitezke, topologia ez-linealen kasuan). Kateko lehenbiziko HPak ez dauka honelako atributurik.
- `vm-type`: HPa nodo dedikatu batean exekutatu nahi bada, nodo dedikatuaren identifikadorea adierazi behar da atributu honen bidez. Nodo dedikatuei Stormen konfigurazioan ezartzen zaie identifikadorea (informazio xehetuagoa sistemaren dokumentazioan aurki daiteke).

HP bakoitzaren `source` balioekin osatzen da topologia. Erabiltzaileak HPen `input` eta `output` atributuetan adierazitako geruzak zein diren kontuan hartu beharko du topologia osatzeko garaian, atributu horiek adierazten baitute HPen arteko dependentziak zein diren. Bestalde, `procTime` atributuak HParen exekuzio-denbora erlatiboa adierazten du. Sistemak balio hori erabiliko du terminal taldean HP horren zenbat instantzia ezarriko diren kalkulatzeko. Izan ere, HP motelenen instantzia kopurua handiagoa izango da HP azkarrenena baino, paralelizazioa handiagoa izan dadin, eta, horrela, HP guztien

exekuzio-denbora errealak (paralelizazioa kontuan hartuta) ahalik eta gehien murrizteko. `procTime` balioak kalkulatzeko paralelizaziorik gabeko exekuzio bat egin beharko da lehenbizi, dokumentu gutxi batzuk prozesatuz. HP bakoitzak behar izan duen denbora neurtuz, `procTime` balioak lortzen dira.

HP bakoitzaren (i) instantzia kopurua (p_i) honako formula honen bitartez kalkulatzen da automatikoki:

$$p_i = \left\lceil \frac{t_i \cdot N}{T} \right\rceil \tag{6.1}$$

non t_i i HParen exekuzio-denbora erlatiboa den (`procTime` atributuaren balioa), $T = \max(t_i)$ HP motelenaren exekuzio-denbora den, eta N nodo langileen kopurua den. Formularen arabera, HP baten exekuzio-denbora handiagoa den heinean, esleituko zaion instantzia kopurua ere handiagoa izango da. Gainera, HP motelenaren exekuzio-denborarekin (T) zatituz, gehienezko instantzia kopurua nodo langileen kopurura mugatzen da, nodo berean HP bakoitzaren instantzia bat baino gehiago edukitzea ekidinez. Formularen emaitza zatidura gorantz biribilduz lortzen da, HP azkarrenek ere gutxienez instantzia bat badutela ziurtatzeko.

6.3.6. Sistema hutsetik ezartzen

Eraikitako arkitektura nahiko konplexua denez, makina birtualetan oinarritu gara erabilgarritasuna errazteko asmoz. Nodo nagusia eta nahi adina nodo langile sortzea oso erraza da publikoki eskuragarri jarri dugun softwarearekin[4]. Bertan, hainbat script eskaintzen ditugu, sistema hutsetik hasita lehenbiziko prozesaketak egiterainoko bidea oso erraza egiten dutenak.

Besteak beste, paketeak honakoa eskaintzen du:

- Nodo nagusi batez eta nahi adina nodo langilez osatutako terminal talde eskalagarria sortzeko scriptak.

[4] `https://github.com/ixa-ehu/vmc-from-scratch` (kontsulta: 2017-05-08)

- Nodo bakoitzaren makina birtualari esleitutako memoria eta CPU kopurua ezartzeko aukera.
- NewsReader proiektuan diseinatutako eta erabiltzeko prest dagoen 19 HPz osatutako analisi-kate ahaltsua.
- Kanpotik ekarritako HPak lehenetsitako analisi-katean integratzeko edo analisi-kate berriak sortzeko aukera.
- HP bakoitzaren instantzia kopuru konfiguragarria. Besterik ezean, automatikoki kalkulatuko da HP bakoitzaren eraginkortasunaren arabera, baina erabiltzaileak kopuru zehatzak ezar ditzake.
- Memoria asko behar duten HPentzat nodo dedikatuak sortzeko aukera.
- Nodo nagusitik nodo langile guztietara modu errazean softwarea edo beste edozein baliabide sinkronizatzeko azpiegitura.
- Dokumentuen sarrera-ilarara dokumentuak bidali eta irteera-ilaratik dokumentuak eskuratzeko scriptak.

Funtzionalitate guztien xehetasunak proiektuaren dokumentazioan aurki daitezke, adierazitako helbidean.

6.3.7. Esperimentuak eta emaitzak

Atal honetan, aurkeztutako sistema banatuaren abiadura eta eskalagarritasun maila neurtzeko egindako hainbat esperimentu azalduko ditugu. Esperimentu horiekin, dokumentu asko prozesatzean sistema nola dabilen aztertu nahi izan dugu, prozesaketak hainbat nodoz osatutako terminal talde batean eginez.

Esperimentuen helburu nagusia sistemaren eskalagarritasuna neurtzea izan da. Hau da, dokumentuak prozesatzen emandako denbora neurtzetik haratago, nodo berriak gehitu ahala, sistemaren prozesamendu-ahalmena nola igotzen den aztertu nahi izan dugu. Kasurik onenean, abiadura linealki hobetuko litzateke. Hau da, makinak bikoiztuz, dokumentu kopuru jakin bat prozesatzeko behar den denbora erdira jaitsiko litzateke. Hala ere, sistema

banatuetan beti dago nodoen eta trafikoaren kudeaketak eragindako kostu gehigarri bat, eta, beraz, hobekuntza lineala izatea praktikan ezinezkoa denez, ahalik eta gehien hurbiltzea da helburua.

Esperimentuekin sistemaren bi ezarpen nagusi aztertu ditugu: batch- eta streaming-ezarpenak. Batch-prozesaketan dokumentu multzo osoa hasieratik bidaltzen da prozesatzera. Streaming-prozesaketan, aldiz, dokumentuak iturri jakin batetik edo gehiagotatik etor daitezke, noiz iritsiko diren jakin gabe. Hori dela eta, streaming moduan, sistema martxan jartzen da eta dokumentuen zain gelditzen da, iritsi orduko prozesatuz. Prozesuak ez dauka, berez, amaierarik. Hortaz, erabiltzaileak exekuzioa aktiboki amaitzen duenean amaitzen da prozesua. Batch- eta streaming-prozesaketak optimizatzearren, atal honetan aurrerago azalduko ditugun ezarpen jakin batzuk esleitu dizkiogu bakoitzari. Batch-prozesaketaren optimizazioaren helburua dokumentu multzo osoa ahalik eta azkarren prozesatzean datza (*throughput*), eta streaming-prozesaketarenean, berriz, dokumentu bakoitza ahalik eta azkarren prozesatzean.

HPen topologiak ere bi motatakoak izan daitezke: linealak eta ez-linealak. Topologia linealetan HP guztiak bata bestearen atzetik exekutatzen dira. Topologia ez-linealetan, aldiz, elkarren artean dependentziarik ez duten HPak paraleloan exekutatzen dira. Batch- eta streaming-ezarpenen eta topologia lineal eta ez-linealen artean badago lotura estu bat. Izan ere, topologia linealak egokiagoak dira batch-prozesaketarako, eta ez-linealak egokiagoak dira streaming-prozesaketarako. Hori dena aurrerago azalduko dugu sakonago.

Ingeleserako egindako esperimentuetan erabilitako analisi-katea 6.1 taulan zerrendatutako HPek osatzen dute. NewsReader proiektuan zehar garatutako HPak erabili ditugu, bai ingeleserako eta bai gaztelaniarako egindako esperimentuetan.

6.2 taulan, esperimentuetan erabilitako dokumentu multzoak azaldu ditugu, bakoitzaren hitz eta esaldi kopuruak zehaztuz. Dokumentuak automobilen industriaren domeinuko berriei dagozkie, eta NewsReader proiektutik hartuak izan dira. Dokumentu multzoak aipatzen hasita, esango dugu en100 dokumentu multzoa (en1K-ren azpimultzoa) garapenean zehar erabili dugula, emaitza horietan oinarrituz parametro eta konfiguraziorik egokienak

HP	Deskribapena	Sarrera (NAF geruza)	Irteera (NAF geruza)
TOK	Tokenizatzailea, Esaldi-zatitzailea	Testu gordina (`raw`)	Tokenak (`text`)
POS	Kategoria--etiketatzailea	Tokenak (`text`)	Lemak, kategoriak (`terms`)
NERC	Entitate-izenen ezagutzailea	Lemak, kategoriak (`terms`)	Entitate-izenak (`entities`)
Parse	Sintaxi-analisia	Tokenak (`text`), kategoriak (`terms`)	Sintaxi-zuhaitzak (`constituency`)
Coref	Korreferentzia-ebazpena	Entitateak, sintaxi-zuhaitzak (`entities, constituency`)	Korreferentzia-erlazioak (`coreferences`)
Opinion	Sentimendu-analisia	Entitateak, sintaxi-zuhaitzak (`entities, constituency`)	Iritziak (`opinions`)
WSD-ukb	Hitzen adiera-desanbiguazioa	Lemak, kategoriak (`terms`)	Synset-ak (`terms`)
WSD-ims	Hitzen adiera-desanbiguazioa	Lemak, kategoriak (`terms`)	Synset-ak (`terms`)
NED†	Entitate-izenen desanbiguazioa	Tokenak, lemak, kategoriak (`text, terms, entities`)	Entitate desanbiguatuak (`entities`)
SRL	Dep. erauzlea, rol semantikoen etiketatzea	Lemak, kategoriak (`terms`)	Dependentziak, rol semantikoak (`deps, srl`)
time	Denbora-espresioen ezagutzea	Lemak, entitateak, sintaxi-zuhaitzak (`terms, entities, constituency`)	Denbora-espresioak (`timeExpressions`)
eCoref	Gertakarien korreferentzia	Lemak, rol semantikoak (`terms, srl`)	Gertakarien korreferentziak (`coreferences`)
tempRel	Denbora-erlazioak	Lemak, entitateak, sintaxi-zuhaitzak, korref., rol sem., denbora-espresioak (`terms, entities, constituency, coreferences, srl, timeExpressions`)	Denbora-erlazioak (`temporalRelations`)
causalRel	Erlazio kausalak	(`terms, entities, constituency, coreferences, srl, timeExpressions, temporalRelations`)	Erlazio kausalak (`causalRelations`)
Fact	Faktualtasuna	Lemak (`terms`)	Faktualtasun-anotazioak (`factualityLayer`)

6.1 taula: Esperimentuetan erabilitako HPen zerrenda, bakoitzaren sarrera eta irteerako anotazio-geruzak ere adieraziz. † ikurra daukaten HPek bezero-zerbitzari eredua jarraitzen dute.

		Hitzak			Esaldiak		
Izena	Dok. kopurua	N	μ	σ	N	μ	σ
en70K	70.000	$16,7 \times 10^6$	238,1	191,9	688.692	9,8	8,5
en1K	1.000	874.799	874,8	86,4	35.536	35,5	10,0
en100	100	92.305	923,1	75,7	3.494	34,9	10,8
sp	1.873	989.168	528,1	383	28.331	15,1	11,6

6.2 taula: Esperimentuetan erabilitako ingelesezko eta gaztelaniazko dokumentu multzoak. Dokumentu multzo bakoitzaren hitz eta esaldi kopurua, batezbestekoa (μ) eta desbideratze estandarra (σ) adierazi ditugu.

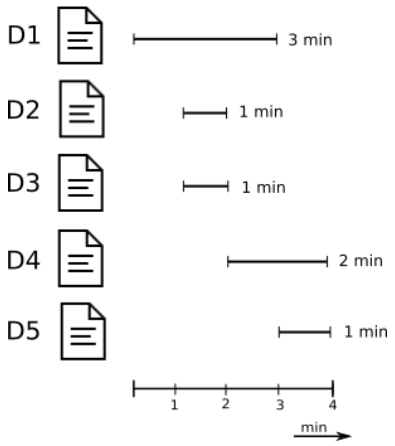
aukeratzeko. Behin konfigurazio optimoa aukeratuta, dokumentu multzo handiagoekin egin ditugu esperimentuak, sistemaren benetako abiadura eta eskalagarritasuna neurtzeko. Multzo horretako dokumentuen batezbesteko esaldi kopurua 35 da. en70K dokumentu multzoak, berriz, 70.000 dokumentu ditu, eta dokumentu txikiak eta ertainak ditu, 9,8 esaldikoak batez beste. en1K multzoa en70K-ren 1.000 dokumentuko azpimultzo bat da, bertako dokumentu handienetakoekin osatua, eta batez beste 35,5 esaldi dituzte bertako dokumentuek. Dokumentu handiagoekin ere egin ditugu probak, baina HP batzuek arazoak eman dituzte dokumentu handienekin, eta esperimentu horiek bertan behera utzi ditugu.

Esperimentuetarako erabili ditugun makina birtualak 16 PUZ-nukleo (E5--2640 2.00GHz) eta 128GB RAM dituen makina batean ezarri ditugu. Bestalde, dokumentuen sarrera eta irteerako ilarak Apache Kafka erabiliz kudeatu ditugu.

6.3.7.1. Neurketak egiteko erabilitako metrikak

Ohiko lau metrika erabili ditugu esperimentuen emaitzak neurtzeko: latentzia, throughputa, azkartze maila eta errendimendua.

Aipatutako lau metrikak ulertzeko, lehenik eta behin, ezinbestekoa da ulertzea denbora neurtzeko bi modu erabiltzen ditugula. Kontuan izanik exekuzioak sistema banatuetan egiten ditugula, eta, beraz, paralelizazioa egon daitekeela, igarotako denbora eta prozesamendu-denbora bereizten ditugu.



6.5 irudia: Exekuzio paraleloa dela eta, prozesamendu-denbora eta igarotako denbora desberdinak dira. Irudiko exekuzioaren prozesamendu-denbora 8 minutukoa da, dokumentu bakoitza prozesatzen emandako denborak metatu egiten baitira. Igarotako denbora, berriz, 4 minutukoa da, pasatutako denbora erreala baita horretarako kontuan hartzen dena.

Igarotako denbora, prozesaketa hasten den unetik amaitzen den uneraino pasatutako denbora erreala da. Prozesamendu-denbora, berriz, nodo bakoitzak prozesatzen emandako denborak batuz kalkulatzen da, eta adierazten duena da, prozesaketa nodo bakarrean, inolako paralelizaziorik gabe, egin izan balitz, prozesaketa egiten behar izango litzatekeen denbora erreala. 6.5 irudian igarotako denbora eta prozesamendu-denbora ulertzen laguntzeko adibide bat ikus daiteke.

Latentzia, datu-unitate bakoitza (dokumentu, esaldi, hitz...) prozesatzen batez beste igarotako denbora da. Dokumentuen kasuan, 6.2 formulan ikus daitekeen bezala, D dokumentu multzoa izanik, bakoitza prozesatzen igarotako denbora erreala kalkulatu behar da lehenik (T_d), eta denen arteko batezbestekoa kalkulatu. Horrela, adibidez, bost dokumentu prozesatzeko bost minutu behar izan direla jakitea ez da nahikoa dokumentuen batez besteko latentzia zein izan den kalkulatzeko. Hori ulertzeko, aztertu 6.6 irudia. Bertan bi prozesaketa desberdinen eskemak ageri dira. Bietan bost dokumentu prozesatu dira, eta, bietan, igarotako denbora bost minutukoa izan

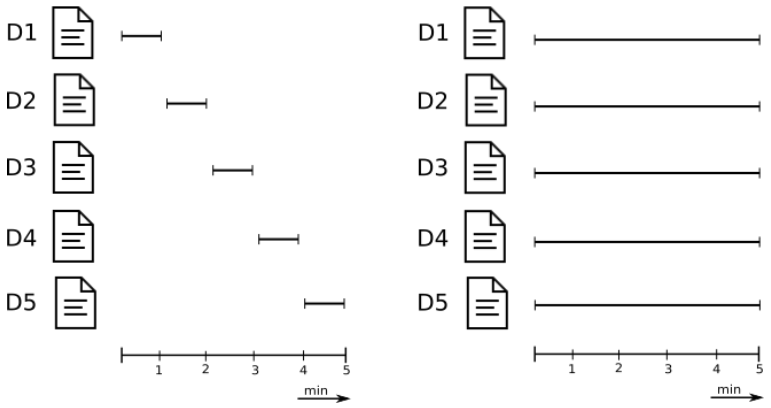
da. Hala ere, irudian ikus daiteke nola eskuineko kasuan bost dokumentuak paraleloan exekutatu diren, eta nola, benetan, bakoitzak bost minutu behar izan dituen. Ezkerreko kasuan, aldiz, prozesaketa seriean gauzatu da, eta dokumentu bakoitzarekin igarotako denbora erreala minutu bakarrekoa izan da. Hortaz, eskuinekoan latentzia 5 min/dokumentu izan da, eta ezkerrekoan 1 min/dokumentu.

$$latentzia = \frac{\sum_{d \epsilon D} T_d}{|D|} \tag{6.2}$$

Throughputak denbora-unitate batean prozesatutako datu-unitate kopurua adierazten du. Horretarako, dokumentuen kasuan, 6.3 formulan ikus daitekeen bezala, prozesatutako dokumentu kopuru totala ($|D|$) eta prozesaketa osoa egiten igarotako denbora (T_D) ezagutu behar dira. Kasu honetan, exekuzioaren paralelizazio maila ez da kontuan hartzen balioa kalkulatzeko. Horregatik, 6.6 irudiko bi kasuetan throughputa berbera da: 1 dokumentu/minutu. Dokumentuen prozesaketa paraleloa dela eta, irudiko eskuineko kasuan gertatzen den bezala, latentzia eta throughputa ez daude zuzenean erlazionatuta. Latentziatik ezin da throughputa eratorri, ez eta throughputetik latentzia ere. Paralelizaziorik ez dagoenean, irudiko ezkerreko adibidea kasu, throughputaren alderantzizkoa da latentzia, eta latentziaren alderantzizkoa throughputa.

$$throughputa = \frac{|D|}{T_D} \tag{6.3}$$

Latentziak eta throughputak sistemak dokumentuak zein abiaduran prozesatzen dituen adierazten dute. Hala ere, gure sistemaren ezaugarri garrantzitsuena eskalagarritasuna da, hau da, nodo berriak gehitu ahala, sistema osoaren abiadura ahalik eta gehien igotzeko gaitasuna. Horretarako, latentziak eta throughputak ez dute, beren horretan, informazio esanguratsurik ematen. Izan ere, latentzia eta throughputa erabiliz eskalagarritasuna neurtzeko, nodo kopuru desberdinekin egindako esperimentuen latentzia eta throughputak



6.6 irudia: Bost dokumenturen bi prozesaketa posible denboran zehar nola exekutatu diren adierazten duten bi eskema.

konparatu beharko genituzke. Badaude exekuzio bakarrean sistemaren eskalagarritasuna neurtzeko adierazgarriagoak diren bi metrika: azkartze maila eta errendimendua.

Azkartze mailak prozesamendu-denbora (T_p) eta igarotako denbora (T_i) erlazionatzen ditu (ikus 6.4. formula). Horrela, prozesaketa nodo bakarrean egitetik nodo kopuru jakin batean egitera igarotako denbora zenbat aldiz murriztu den adierazten du. 6.5 irudiko (145. or.) exekuzioan, suposatuz bost nodo daudela eta dokumentu bakoitza nodo banatan prozesatu dela, azkartze maila 2koa da, prozesamendu-denbora igarotakoaren bikoitza baita. Gehienezko azkartze maila, beraz, nodo kopuruaren adinakoa da. Adibidez, bost nodoko sistema banatu batean lor daitekeen azkartze mailarik altuena 5 da, horrek esan nahiko bailuke prozesaketa bost nodotan egitea nodo bakarrean egitea baino bost aldiz azkarragoa izan dela. Azkartze mailak, dagoeneko, eskalagarritasunari buruzko informazioa ematen du.

$$azkartzea = \frac{T_p}{T_i} \tag{6.4}$$

Hala ere, ezin dira, besterik gabe, nodo kopuru desberdinez osatutako sistemetan lortutako azkartze mailak elkarrekin alderatu, azkartze maila nodo

kopuruaren menpekoa baita. Horretarako, lortutako azkartze maila, azkartze maila maximoarekin (nodo kopuruarekin) zatituz, errendimendu-balioa lortzen da (ikus 6.5. formula). Errendimendua 0 eta 1 balioen artean kokatzen den azkartze maila besterik ez da. Horrela, edozein sistemaren errendimenduak konparagarriak dira. Balioa ehuneko moduan ematea ere ohikoa da. 6.5 irudiko (145. or.) exekuzioaren errendimendua, berriro bost nodo daudela eta dokumentu bakoitza nodo batean prozesatu dela suposatuz, 0,4 da, edo, ehuneko moduan emanda, % 40. Balioa % 100dik urrun egotearen arrazoia da nodoei ez zaiela etekin handirik atera, nodo guztiak egon baitira hainbat minutuz geldirik.

$$errendimendua = \frac{azkartzea}{N} \tag{6.5}$$

6.3.7.2. Batch-prozesaketa

Batch moduan prozesatzean, dokumentu multzo osoa hasieratik bidaltzen da prozesatzera, dena batean. Prozesaketa modu hau egokia da corpus bat prozesatu behar denean, esate baterako. Horrelakoetan, normalean, prozesaketa osoa bukatu arte emaitzak ez dira erabilgarriak; beraz, dokumentu multzo osoa ahalik eta lasterren prozesatzea da helburua. Horregatik, throughputa ahalik eta altuena izan dadin nahi dugu.

Sistema batch moduan probatzeko, 7 MBez osatutako terminal talde bat ezarri dugu, horietako bat nodo nagusia izanik eta beste 6ak nodo langileak. MB bakoitzak PUZ-nukleo bakarra dauka esleituta, eta 12 GB RAM memoria.

Hainbat modu daude HPak terminal taldean antolatzeko. Nodo langile bakoitzean HP bakoitzaren kopia bana ezar daiteke, edo HPrik motelenen instantzia gehiago ezar daitezke abiadura konpentsatzearren. Ezarpenik egokiena aukeratzeko, hainbat ezarpen diseinatu eta probatu ditugu, ondoren emaitzak aztertzeko:

- Oinarri-lerroa: HP bakoitzaren instantzia bakarra terminal talde osoan. Modu honetan, ez dago inolako paralelizaziorik.

- ALL_6: HP bakoitzaren 6 instantzia. Ez da ziurtatzen dokumentu bakoitza zein nodotan prozesatuko den analisi-katearen fase bakoitzean. Kateko pauso bakoitzean, dokumentuen banaketa nodoetan zehar ausazkoa denez, baliteke nodo batek bi dokumentu aldi berean prozesatzea, beste nodoren bat libre dagoen bitartean.

- SRL_6: SRL HParen 6 instantzia, nodo langile bakoitzean bana, eta gainontzeko HPen instantzia bakarra. SRL paralelizatzea erabaki dugu, prozesaketa-denbora gehien behar duena delako.

- p_6: HP bakoitzaren instantzia kopurua 6.1 formularen arabera (140. or.) kalkulatzen da. Formulak HP motelenei instantzia gehiago esleitzen dizkie azkarrenei baino.

- MONO: Kasu honetan, nodo bakoitzean HP guztien instantzia bana dago, eta dokumentu bakoitza nodo berean prozesatzen da osorik. ALL_6 ezarpenaren antzekoa da, baina dokumentuak ez dira nodo batetik bestera aldatzen. Horrela, nodo berean HP bakarra ari daiteke prozesatzen aldi berean, nodo langile bat ez baita dokumentu berri bat prozesatzen hasiko aurrekoa osorik prozesatzen bukatu arte.

6.3 taulan, en100 dokumentu multzoko 100 dokumentuak ezarpen bakoitzarekin prozesatzen igarotako denborak ikus daitezke. Paralelizaziorik gabe, 4 ordu eta erdi eskas behar izan dira prozesaketa amaitzeko. Emaitzei erreparatuz, ikus daiteke emaitza hobeak eman dituela SRLren instantzia kopurua bakarrik handitzeak dokumentuen fluxua kontrolatu gabe HP guztien instantzia ugari sortzeak baino. ALL_6 moduan dokumentuen fluxurik kontrolatzen ez denez, horrek nodo batzuen gainkarga eragin dezake, sistema osoa desorekatuz eta abiadura motelduz. Instantzia kopuruak formularen bidez kalkulatzeak gainkarga horiek txikitzen ditu, baina, hala ere, emaitza SRL_6-rena baino 5 puntu okerragoa da. Emaitzarik onena dokumentuen fluxua kontrolatzean lortzen da, MONO ezarpenarekin. Kasu horretan, CPU guztiak etengabe ari dira lanean, batch eredua izanik beti egongo baitira dokumentu berriak zain, denak bukatzen diren arte behintzat. Gainera, CPU bakoitzak ez ditu inoiz bi dokumentu aldi berean prozesatuko. Horrela, paralelizazio maila optimoa lortzen da.

Ezarpena	Igarotako denbora	Irabazia
Oinarri-lerroa	260,67	–
ALL_6	81	% 69,74
SRL_6	68,05	% 74,58
p_6	73,22	% 72,65
MONO	55,3	**% 79,34**

6.3 taula: en100 dokumentu multzoa prozesatzen emandako denborak 6 nodo langilez osatutako terminal talde batean. Batch ereduari jarraituz egin dira prozesaketa guztiak. Hainbat ezarpen probatu ditugu batch-prozesaketarako portaerarik onena zeinek daukan ikusteko. Denborak minututan neurtuta daude.

Datuak	Igarotako denb.	Throughputa	Proz.-denb.	Azkartzea	Errend.
en1K	719,4	1,38	6.480,22	9,01	% 90,1
en70K	38.159,47	1,83	31.328,13	8,21	% 82,1

6.4 taula: Batch eredua jarraituz egindako prozesaketen estatistikak. Denborak minututan neurtuta daude.

Emaitza horiek aintzat hartuta, en1K eta en70K dokumentu multzoak prozesatu ditugu. Horretarako, nodo nagusi bat eta 5 nodo langile erabili ditugu, baina nodo bakoitzari bi PUZ-nukleo esleituz, eta, hartara, 10eko paralelizazio maila ezarriz. Erabilitako ezarpena MONO izan da.

Emaitzak 6.4 taulan daude ikusgai. Aurrekoarekin alderatuz, dokumentu kopurua asko hazi da, 1.000 eta 70.000 dokumentu prozesatu baitira oraingo honetan. Igarotako denboraz gain, prozesaketa-denbora ere gehitu dugu taula honetan. Datu horiekin, azkartze maila eta errendimendua ere kalkulatu ditugu. 70.000 dokumentuak nodo bakarrean exekutatzeko 217 egun beharko genituzke, eta 5 egun 1.000 dokumentuak prozesatzeko. Adierazitako terminal taldean exekutatuz, 26 egunera eta 10 ordutara jaitsi ditugu denborak, hurrenez hurren, 8,2 eta 9,01eko azkartze mailak lortuz (balio maximoak 10 izanik), eta % 82 eta % 90,1eko errendimenduak lortuz.

Emaitza oso onak izan arren, gogoeta egin dugu ulertzeko zergatik ez garen balio maximo teorikora are gehiago hurbildu. Alde batetik, ikusi dugu nodo bakoitzean prozesatutako azkeneko dokumentuak ez direla une berean buka-

tzen. Hori dela eta, prozesua bukatutzat jo baino lehen, nodo batzuk zain daude lanik egin gabe. Horrek azkartze mailaren eta errendimenduaren jaitsiera dakar. Horrekin lotuta, eta hau da eragin handiena izan duen puntua, prozesaketa martxan jartzeko unean, dokumentu guztiak nodo bakoitzaren sarrerako ilaren artean banatu behar dira. Nodo bakoitzari dokumentu kopuru berdina esleitzen zaion arren, dokumentuen tamainak ez dira kontuan hartzen. Hori dela eta, nodo batzuek prozesaketa besteek baino lehenago bukatzen dute, eta denbora-tarte bat geldirik pasatu behar izaten dute, besteek bukatuko zain. Hori Apache Kafkaren ilarak partizionatzeko beharrak eragindako arazoa da, kontsumitzaile bakoitzeko ilara-partizio bat sortzera behartzen baitu.

6.3.7.3. Streaming-prozesaketa

Prozesaketa streaming ereduari jarraituz egiteko ere optimizatu nahi izan dugu sistema. Batch ereduarekin alderatuz, desberdintasun nagusia da streaming ereduan sistema etengabe dagoela martxan, dokumentuak noiz iritsiko zain, eta dokumentuak edozein unetan irits daitezkeela, banaka edo multzoka. Streaming-prozesaketan, edozein unetan dira ordura arte lortutako prozesaketaren emaitzak erabilgarri. Adibidez, demagun historian zehar izandako gertakarien erregistro bat mantendu nahi dugula (NewsReader-en egin den antzera), iturri desberdinetatik lortutako artikulu eta berriak prozesatuz, eta gertakari berberari dagozkienak elkarrekin bilduz. Denborak aurrera egin ahala, gertakari berriak etengabe iritsiko dira sistemara streaming-ingurune batean, eta prozesua ez da inoiz amaituko, etorkizuneko gertakariak ere bildu nahi izango baititugu. Hala ere, edozein unetan, ordura arte prozesatutako dokumentuekin lortutako erregistroa baliagarria izango da, denbora-tarte jakin bateko gertakariei buruzko informazio osatua emango baitu. Horregatik, dokumentu berri bat iritsi ahala, ahal bezain laster prozesatzea da helburua. Dokumentu multzo baten prozesaketa optimizatu ordez, dokumentu bakarrarena optimizatuko dugu. Batch-prozesaketan garrantzitsuena throughputa igotzea zen bezala, kasu honetan dokumentuen latentzia hobetzea dugu helburu.

Streaming-ingurune bat simulatzeko asmoz, Poissonen banaketan oinarritu-

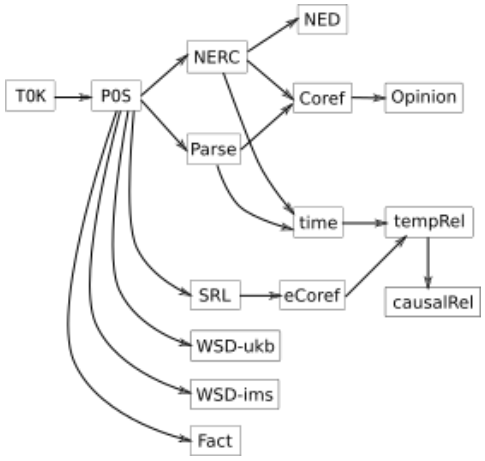
Ezarpena	Datuak	Proz. denb.	Latentzia (dok/esaldi/hitz)
Lineala	en1K	4.620,90	5,00 / 0,14 / $5{,}78\times10^{-3}$
Ez-lineala	en1K	5.421,33	2,78 / 0,08 / $3{,}13\times10^{-3}$

6.5 taula: Streaming-prozesaketako esperimentuen prozesaketa-denbora eta latentziak, topologia lineal eta ez-linealekin. Denborak minututan neurtuta daude.

tako dokumentu-bidaltzaile bat inplementatu dugu. Prozesuak, aurresan ezin daitekeen denbora-tarte bat pasatzean, dokumentu berri bat bidaliko du prozesatzera. Denbora-tartea, aurresan ezin daitekeen arren, kontrolatua da nolabait. Izan ere, oinarri-erritmo bat zehaztuz, batez beste erritmo horretan bidaliko ditu dokumentuak, baina ez beti abiadura berean. Hau da, dokumentu bakoitza denbora-tarte aldakor bat pasatu eta gero bidaliko da, baina dokumentu asko bidali direnean denbora-tarte guztien batezbestekoak oinarri-erritmorantz joko du. Batch esperimentuen emaitzak ikusita, badirudi dokumentu bakoitzak, batez beste, minutu bat eta minutu eta erdi artean behar duela prozesatzen. Hortaz, Poissonen banaketaren oinarri-erritmo bezala 1,000 dokumentu 33 ordutan prozesatzeko abiadura ezarri dugu (dokumentu bat bi minuturo). Horrela, batzuetan dokumentu bat baino gehiago aldi berean prozesatu behar izango dira, baina sistema gainkargatu gabe.

Oraingoan, instantzia kopuruak 6.1 formularen (140. or.) bitartez kalkulatu ditugu. Badakigunez sistema ez dela batch-prozesaketan bezainbeste kargatuko, ez dugu PUZ-nukleo bakoitza gehiegi kargatzeko beldurrik izan. en1K dokumentu multzoa modu horretan prozesatzean lortutako emaitzak 6.5 taulako lehenbiziko lerroan daude ikusgai. Prozesaketa-denbora totala jaitsi egin da, sistemaren kargatze maila baxuagoa dela eta. Azkartze maila eta errendimendua ez ditugu erakutsi taula honetan, sistemak dokumentuen zain denbora asko pasa dezakeenez, neurri horiek ez baitute informazio esanguratsurik ematen streaming-prozesaketan.

Batch-prozesaketan ez dauka zentzu handirik topologia ez-linealak (ikus 6.7 irudia) ezartzeak, bestela ere CPU guztiak etengabe lanean ari baitira. Kasu honetan, aldiz, dokumentu bakarra hainbat makinaren artean prozesatzeko,

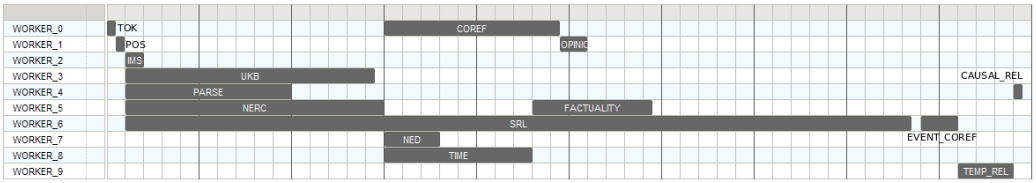


6.7 irudia: Esperimentuetan erabilitako ingeleserako topologia ez-lineala.

analisi-kateko HP bakoitza makina desberdinetan, aldi berean, prozesa dezakegu, dokumentu bakar hori prozesatzen igarotako denbora murriztuz. 6.7 irudiko topologiari esker, TOK eta POS HPak bata bestearen atzetik exekutatuko diren arren, NERC, Parse, SRL, WSD-ukb, WSD-ims eta Fact HPak paraleloan exekuta daitezke, bakoitza nodo edo PUZ-nukleo batean, ez baitute besteen emaitzen beharrik beren lana gauzatzeko. Topologia ez-lineala erabiliz, 6.5 taulako bigarren lerroan ageri diren emaitzak lortu ditugu. Ikus daitekeenez, prozesaketa-denbora igo egin den arren, latentzia asko hobetu da, ia erdira jaitsiz. Kontuan hartzen badugu MB guztiak makina fisiko bakarraren gainean muntatu ditugula, espero daiteke prozesaketa-denbora igotzea, sistema gehiago kargatzen baita paralelizazioa handitu ahala. Izatez, topologia ez-linealen kasuan, paraleloan exekutatzen ari diren HPen kopurua terminal taldeko nodo kopurua baino handiagoa izan baitaiteke.

HP guztien exekuzio-denborak berdinak edo antzekoak balira, 6.7 irudiko topologiarekin % 50ekoa baino hobekuntza hobea lortuko genuke, analisi-katea 15 fase izatetik 6 izatera pasatzen baita. Hala ere, HPen arteko desoreka oso handia da, SRLk berak dokumentu baten prozesaketaren denboraren % 60 hartzen baitu. Ondorioz, beste nodoetako HPek lehenago bukatu arren, SRLk bukatu arte zain egon behar dute hurrengo fasearekin hasteko, SRLren

emaitza beharrezkoa baita hurrengo faseko HPentzat. 6.8 irudian ikus daiteke nola banatzen den dokumentu baten analisi-kate osoaren exekuzio-denbora terminal taldeko nodoetan zehar. Irudian garbi ikus daiteke SRLren exekuzioak nola eragiten duen dokumentuaren latentzian, gainerako nodo guztiak berak bukatzeko zain gelditzen baitira une batetik aurrera.



6.8 irudia: Dokumentu bat topologia ez-linealarekin prozesatuta, prozesaketa-denboraren banaketa HPen artean nolakoa izan den adierazten duen eskema.

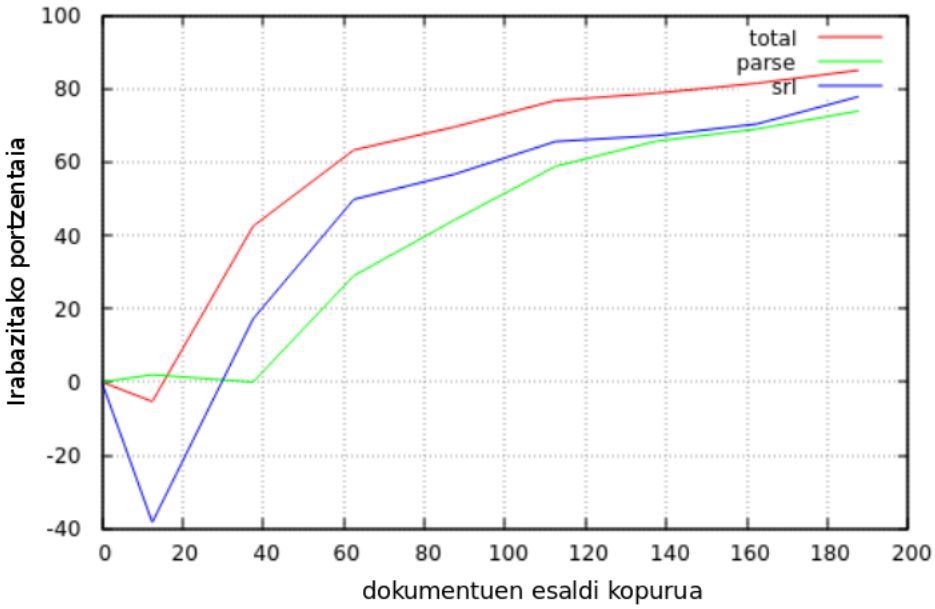
6.3.7.4. Dokumentuen granularitatea ustiatuz

6.8 irudian ikusi dugu nola analisi-kateko HP baten exekuzio-denbora luzea denean, haren irteera behar duten HPek zain egon behar izaten duten, prozesaketaren latentzia handituz. Hori konpontzeko beste optimizazio bat dokumentuak esalditan zatitu eta esaldiak nodoen artean banatzea izan daiteke. Ideia da PUZ-nukleo bakoitzak dokumentu osoa prozesatu ordez, esaldi edo paragrafo bat soilik prozesatzea, sarrerako datuen granularitatea aldakor bihurtuz eta latentzia nabarmen txikituz. Horrek PUZ-nukleo gehiagoren beharra eskatzen du, paralelizazio maila handituko baita. Horrelakorik egin gabe, kasu ideala aldi berean prozesatu beharreko dokumentu adina PUZ-nukleo edukitzea bazen, esaldi mailako granularitatearekin aldi berean prozesatu beharreko esaldi adina PUZ-nukeo eskuragarri edukitzea litzateke ideala.

Baldintza horiek betetzen direla suposatuz, azterketa teoriko bat egin dugu. Horretarako, 50 dokumentuko 8 multzo prozesatu ditugu rol semantikoen etiketatzailearekin eta analizatzaile sintaktikoarekin, multzo bakoitzean esaldi kopuru-tarte bateko dokumentuak sartuz. Multzo bakoitzean honako esaldi kopuruak dituzten dokumentuak sartu ditugu: [0-25), [25-50), [50-75),

[75-100), [100-125), [125-150), [150-175) eta [175-200).

Azterketa teorikoa egin dugula diogu ez baitugu ingurune erreal batean egin prozesaketa. Aldiz, kasu ideala simulatzeko, prozesaketa paralelizaziorik gabe egin dugu, testuak banan-banan prozesatuz. Gero, azterketa bat egin dugu suposatuz denbora horiek lortu ditugula testuak paraleloan prozesatuta. Horregatik, emaitza hauek islatzen dutena da zenbaterainoko hobekuntza lortuko genukeen latentzian PUZ-nukleo kopurua dokumentu multzo bakoitzeko esaldi kopurua bezainbestekoa balitz, gutxienez.



6.9 irudia: Egindako azterketa teorikoaren arabera, grafika honetan ikus daitezkeen hobekuntzak lortuko genituzke PUZ-nukleo kopuru mugagabearekin, sarrerako testuak esalditan zatituko bagenitu. Grafikan ikus daiteke prozesatzen igarotako denbora nola hobetzen den dokumentuen batezbesteko esaldi kopuruak gora egin ahala.

Prozesaketa bi modutan egin dugu, dokumentuz dokumentu lehenik, eta esaldiz esaldi ondoren. Behin prozesaketa-denborak aterata, konparatu dugu zein izango litzatekeen igarotako denbora bi kasuetan, suposatuz sarrerako

testu guztiak paraleloan prozesatzeko adina PUZ-nukleo daudela. Emaitza 6.9 irudiko grafikan ikus daiteke. Esaldi gutxiko dokumentuetan denboren hobekuntza ez da adierazgarria, baina dokumentuen esaldi kopuruak gora egin ahala, denborek hobetzeko joera hartzen dute.

Azterketa honetatik bi ondorio atera ditugu. Batetik, testuen granularitatea aldakorra izatea onuragarri izan dadin, PUZ-nukleoen kopuruak oso handia izan behar duela, dokumentuen esaldi kopuruak ere handiak izan ohi baitira. Bestalde, kontuan izan behar da HP askok denbora asko pasatzen dutela datu eta modelo linguistikoak kargatzen. Dokumentuak osorik prozesatzean, hasieraketa dokumentu bakoitzeko behin egin behar da. Esaldiak prozesatzen badira, ordea, hasieraketa osoa esaldi bakoitzeko egin behar da, eta horrek exekuzio-denborak asko okertuko lituzke esaldi askoko dokumentuetan. Beraz, optimizazio hori HPek bezero-zerbitzari moduan funtzionatzen badute bakarrik izan daiteke mesedegarri, kasu horretan HPen hasieraketa behin bakarrik egin beharko bailitzateke.

IV Atala

ONDORIOAK

7. KAPITULUA

Ondorioak eta etorkizuneko lanak

Kapitulu honetan tesi-lan honetan aurkeztutako ekarpenak laburtu eta ateratako ondorioak aurkeztuko ditugu. Horrekin batera, tesian zehar landutako kontzeptu guztiak elkarren artean nola uztartzen diren ere azalduko dugu.

7.1. Ekarpen nagusiak

Tesi-lan honek lau ekarpen nagusi izan ditu, ondoren laburbildu ditugunak:

- AWA anotazio-eskema hobetu dugu. Eskemari anotazio maila berriak gehitu dizkiogu, eta formalizazio lana egin dugu, arloaren egoeran aztertutako ekarpen teorikoak eskemari aplikatuz.

- NAF anotazio-eskema aurkeztu dugu. Helburu orokorreko anotazio linguistikoen eskema hau tamaina handiko proiektuetan erabilia izan da dagoeneko, bere erabilgarritasuna eta egokitasuna berretsiz.

- Anotazio-eskemen arteko elkarreragingarritasuna lortzeko bidean, lehenbiziko pausoak eman ditugu. Anotazio-eredu abstraktu bat aurkeztu dugu, anotazioak adierazpide batetik bestera bihurtzeko zeregina erraztuko duen formalismo gisa.

- Testu-dokumentu kopuru handiak modu eskalagarrian prozesatzeko sistema bat aurkeztu dugu. Arkitektura horrek gaur egun datu handien

prozesaketa egiteko erabiltzen diren teknika eta tresna ezagunenetakoak erabiltzen ditu. Gainera, analisi-kateak modu deklaratiboan sortzeko aukera ematen du, jatorri desberdinetako HPak modu erosoan integratzea ahalbidetuz. Sistema hasieratik bukaeraraino inplementatu dugu eta publiko jarri dugu, edonork erabil dezan.

7.2. Ondorioak

Anotazio linguistikoak adierazteko bi eskema aurkeztu ditugu tesi-lan honetan: Anotazio-Amaraunen Arkitektura (AWA) eta NLP Annotation Format delakoa (NAF).

AWA euskarazko testuak prozesatzeko HPentzako anotazio-eskema izan dadin diseinatu dugu. Euskararen ezaugarriak kontuan hartuz garatu dugu, besteak beste, egitura morfologikoaren aberastasuna eta anbiguotasun maila altua. Hala ere, izaera orokorreko eskema izanik, edozein hizkuntza eta erabilpen-kasutarako egokia izatea izan du hasieratik helburu. Izaera orokor hori indartzeko, datu-eredu abstraktu bat diseinatu dugu lehenik, eta, ondoren, eskema zehatza eraiki dugu horren gainean.

AWAren datu-eredua hiru elementu nagusik osatzen dute: aingurek, informazio linguistikoak eta horien arteko estekek. Informazio linguistikoa TEIk proposatutako ezaugarri-egitura motatuen bidez definitu dugu. Anotazio mota bakoitzarentzat ezaugarri-egitura motatu bat diseinatu dugu, anotazioak zein informazio mota adierazi behar duen kontuan hartuz.

Datu-eredua edozein anotazio-eskemarentzat baliagarria dela pentsatzen dugu, anotazio guztiek amankomunean izan dezaketen egitura biltzen baitu. AWA bera datu-eredu horren gainean eraikitako eskema da. Anotazio mota bakoitzarentzat aingura eta informazio linguistikoaren egitura jakin bana zehaztu dugu, baina, datu-ereduari esker, eskemaren elementu guztiak hezurdura komun baten gainean eraikita daude, geruza guztietako anotazioen arteko koherentzia ziurtatuz. Horrek sendotasuna eman dio eskemari, elementu bakoitzaren izaera garbi definituta gelditu baita. Horren ondorioz, hainbat ezaugarri betetzen ditu gure anotazio-eskemak:

- Anbiguotasuna modu egokian eta naturalean adierazteko gaitasuna

dauka. Izan ere, nahikoa da aingura jakin bati, esteka batez baino gehiagoz baliatuz, informazio linguistiko bat baino gehiago esleitzea.

- Ezaugarri-egitura motatuei esker, informazio linguistiko konplexuak adieraz daitezke. Malgutasun hori ezinbestekoa izan da euskararen kasuan, adierazi beharreko informazio morfologikoa, esaterako, oso konplexua baita.

- Datu-ereduaren ahalmena, besteak beste, interpretazio-aingura deritzogun egiturek islatzen dute. Zenbaitetan, anotazio multzo bat identifikatu nahi izaten dugu, multzo horri informazio linguistiko zehatz bat esleitzeko. Esaterako, termino baten gainean sortu diren anotazio guztien artean azpikategoria morfologiko jakin batekoak bakarrik bil ditzakegu. Horrelakoetan, nahikoa da aingura berri bat definitzea anotazio horiek multzokatzeko, eta aingura berri horri informazio linguistikoa lotuz, anotazio berri bat sortzea. Horregatik, anotazio multzo horri interpretazio deitzen diogu, eta sortutako aingurari interpretazio-aingura.

NAF anotazio linguistikoak adierazteko beste eskema bat da. NAFek ere, izaera orokorreko anotazio-eskema izanik, maila linguistiko anitzetako anotazioak adierazteko gaitasuna eskaintzen du. NAF erabilerraza eta sinplea izateko helburuarekin diseinatu dugu. Inplizituki bada ere, NAFen ere AWAren datu-ereduan identifikatutako hiru elementuak aurki daitezke: aingurak, informazio linguistikoa eta estekak.

NAF datu estekatuen (*linked data*) kontzeptuarekin oso lotuta egon da hasieratik. Helburua NAF anotazioak ezagutza-baseekin lotzea da. Horretarako, aingura mota berezi bat dago NAFen, `externalRef`, anotazio bat kanpoko baliabideetako elementu batekin edo hainbatekin lotzea ahalbidetzen duena.

NAFen egokitasuna eta erabilgarritasuna agerian gelditu dira, NewsReader bezalako neurri handiko proiektuetan arrakastaz erabilia izan baita. Eskema bereziki egokia da ingurune banatuetan erabiltzeko, izan ere, maila linguistiko bakoitzeko anotazioak geruza berri batean ematen baitira, aurrez sortutako geruzak aldatu ordez. Horrela, dokumentu bera bi HPk paraleloan prozesa dezakete, bakoitzak anotazio-geruza oso bat lortuz, eta ondoren, azken

prozesu batek geruza guztiak elkar ditzake, jatorrizko testu-dokumentuari dagokion NAF dokumentu osoa lortuz.

AWA eta NAF eskemekin batera, tesi-lan honetan HPen arteko elkarreragingarritasuna ere landu dugu. Horretarako, HP guztiak anotazio-eskema berbera erabiltzera derrigortu ordez, eskemen arteko bihurketak egitearen alde egin dugu. Bihurketak egiteraino iritsi ez garen arren, bihurketak egin ahal izateko eman behar diren baldintzak jorratu ditugu. Chiarcosen sailkapenean (Chiarcos, 2012b) oinarritu gara, bi elkarreragingarritasun maila nagusi bereiziz: elkarreragingarritasun estrukturala eta elkarreragingarritasun kontzeptuala.

Tesi-lan honetan elkarreragingarritasun estrukturala landu dugu batez ere. Horretarako, helburu orokorreko anotazio-eskemek amankomunean dituzten elementuak identifikatu eta anotazio-eredu abstraktu bat diseinatu dugu. AWAren datu-eredutik abiatu gara eredu abstraktua lortzeko. Behin eredu abstraktua definituta, bi anotazio-eskemaren arteko bihurketa egiteko, lehenik eta behin, eskema horiek eredu abstraktuaren arabera egituratzea proposatzen dugu. OWL teknologiaz baliatuz bi anotazio-eskemak hezurdura komun horren arabera egituratuta, bien arteko mapaketa egiteko bidea asko errazten dela ikusi dugu. Adibide bezala, tesi-lan honetan aurkeztutako AWA eta NAF eskemak eredu abstraktuaren arabera adierazteko ariketa egin dugu.

Soluzio hori Chiarcosen OLiAren soluzioarekin konbinatuz, anotazio-eskema desberdinak erabiltzen dituzten HPen arteko elkarreragingarritasuna lortzea askoz ere hurbilago dagoen ataza dela irizten dugu.

Tesi-lan honen bigarren blokean hizkuntzaren prozesamendurako sistema banatu eta eskalagarria eraiki dugu. Sistemak analisi-kateak nodoen artean modu orekatuan banatzeko gaitasuna dauka, ondoren, jasotako dokumentuak modu banatuan prozesatzeko.

Eskalagarritasun bertikala baztertu eta eskalagarritasun horizontalaren alde egin dugu. Horrela, terminal taldeari nodo berriak gehitzea aski da sistemaren prozesamendu-ahalmena modu ia-linealean hobetzeko. Gure sistemari esker, testu-dokumentu kopuru erraldoiak denbora-tarte mugatuan prozesa

daitezke, sakabanatuta eta modu ez-egituratuan dauden informazio-iturrietatik beharrezko informazioa denbora errealean erauztea ahalbidetuz.

Teknologia ugari erabiliz eraiki dugu sistema. Garrantzitsuenak aipatuz, prozesaketa banatua kudeatzeko Storm erabili dugu, eta, anotazio linguistikoen tarteko-biltegi gisa, MongoDB datu-baseak. Gainera, analisi-katea martxan jartzeko, beharrezkoa izan daiteke hizkuntza-prozesatzaile asko instalatu eta ezarri behar izatea. Gure sistemaren ezarpena eta erabilera errazak izan daitezen, makina birtualetan oinarritutako arkitekturaren alde egin dugu. Gainera, makina birtualen erabilerak errazten du gure sistema gaur egun hain erabiliak diren Amazon Web Service[1] bezalako hodei-konputazioko zerbitzuekin integratzea ere. Publikoki eskuragarri jarri ditugu makina birtual berriak automatikoki sortzeko scriptak[2].

Analisi-kateko prozesatzaileek bete beharreko baldintza bakarra da bai sarrera eta bai irteera NAF anotazio-eskemaren arabera jaso eta ematea. Erabiltzaileak berak baldintza hori betetzen duen bere gustuko edozein analisi-kate ezar dezakeen arren, eskuragarri jarri ditugun makina birtualetan IXA pipes katea ezarri dugu. IXA pipes-en ingeleserako analisi-katea, esperimentuez gain, NewsReader proiektuan erabili dugu. Ixa pipes-ek euskararako katea ere eskaintzen du, eta erabilia izan da, adibidez, QTLeap[3] proiektuan.

Gure sistemaren ezaugarri bereizgarrienetako bat da batch eta streaming ereduetara egokitzen dela. Lehenbiziko kasuan topologia linealak erabiltzen dira, eta bigarrenean ez-linealak. Prozesu mailako integrazioari arreta berezia jarri diogunez, testu-fitxategi bat editatuz topologiak modu deklaratiboan adierazteko gaitasuna eskaintzen du sistemak. Horrela, topologia lineal edo ez-linealak eraikitzea lan erraza bihurtzen da. Hori oso garrantzitsua dela uste dugu, egindako esperimentuetatik ateratako ondorioetako bat izan baita batch-prozesaketarako askoz ere egokiagoak direla topologia linealak, eta, streaming-prozesaketarako, berriz, topologia ez-linealak. Hau da, ondorioztatu dugu topologiaren arkitektura oso garrantzitsua dela erabili nahi den prozesaketa ereduaren arabera.

[1] `https://aws.amazon.com/es` (kontsulta: 2017-05-08)
[2] `https://github.com/ixa-ehu/vmc-from-scratch` (kontsulta: 2017-05-08)
[3] `http://qtleap.eu` (kontsulta: 2017-05-08)

Egindako esperimentuei esker, ikusi dugu sistema oso eskalagarria dela, eta behar bezala egokitzen dela batch eta streaming ereduetara.

Streaming-prozesaketarekin lotuta, sarrerako testuen granularitatea aldakorra izanda latentziak nola hobetuko liratekeen ikusteko azterketa teoriko bat egin dugu. Bi ondorio atera ditugu horren inguruan. Batetik, testuen granularitatea aldakorra izatea onuragarri izan dadin, PUZ-nukleoen kopuruak oso handia izan behar duela, dokumentuen esaldi kopuruak ere handiak izan ohi baitira. Bestalde, kontuan izan behar da HP askok denbora asko pasatzen dutela datu eta modelo linguistikoak kargatzen. Dokumentuak osorik prozesatzean, hasieraketa dokumentu bakoitzeko behin egin behar da. Esaldiak prozesatzen badira, ordea, hasieraketa osoa esaldi bakoitzeko egin behar da, eta horrek exekuzio-denborak asko okertuko lituzke esaldi askoko dokumentuetan. Beraz, optimizazio hori HPek bezero-zerbitzari moduan funtzionatzen badute bakarrik izan daiteke mesedegarri, kasu horretan HPen hasieraketa behin bakarrik egin beharko bailitzateke.

Lehenago ere esan dugun bezala, tesi-lan honetan zehar bi bloke nagusi landu ditugu: anotazio-eskemen mailako integrazioa eta prozesu mailako integrazioa. Azter dezagun sakonago nola uztartzen diren landutako bi blokeak, eta zein izan diren lan honetatik ateratako ondorio nagusiak.

Testu kopuru handien prozesaketa ingurune banatuetan egiteko aurkeztu dugun sistema, izatez, ahalik eta orokorrena egiten saiatu gara. Hau da, sistemak ez du erabiltzailea analisi-kate edo HP jakin batzuk erabiltzera behartzen. Hala ere, bada sistemak ezartzen duen muga bat: HP guztiek NAF anotazio-eskema ulertu eta produzitu behar dute. Izan ere, elkarreragingarritasunaren arazoa oraindik konponduta egotetik urrun dagoen honetan, oso zeregin nekeza da eskema desberdinak darabiltzaten bi HP elkarrekin martxan jartzea. Horregatik, 3. eta 4. kapituluetan landutakoek arazo horri irtenbidea ematen laguntzea dute helburu.

Tesi-lan hau ez da nahikoa izan elkarreragingarritasunaren arazoa behin betiko ebazteko, eta aurkeztutako testuen prozesaketarako sistemak NAF eskema erabiltzen duten HPak bakarrik onartzen jarraitzen du, baina elkarreragingarritasunaren arazoari errotik heldu diogu, eta aurrez beste batzuek jarritako oinarria are gehiago sendotu eta aurrerapauso bat egin dugu.

Elkarreragingarritasunaren arazoa ebazte aldera proposatutako bidea ez ezik, tesi-lan honetan beste hiru ekarpen nagusi ere egin ditugu: AWA eta NAF anotazio-eskemak, batetik, eta testu kopuru erraldoiak denbora-tarte mugatuan prozesatzeko sistema eskalagarria, bestetik. Esan dezakegu hirurak ekarpen teoriko hutsa baino zerbait gehiago izan direla, izan ere, hirurak ari baitira dimentsio handiko proiektuetan erabiltzen.

7.3. Etorkizuneko lanak

Atal honetan etorkizunean egiteko gelditu zaizkigun lanak zerrendatuko ditugu.

Elkarreragingarritasunaren arazoari helduz, ez dugu arazoa bere osotasunean ebatzi. Tesi-lan honen helburuen artean elkarreragingarritasunaren arazoari heldu eta lehenbiziko pauso batzuk finkatzea zegoen. Helburu hori beteta, pauso horiekin jarraitu eta aurrera egitea gelditu zaigu etorkizunerako. Elkarreragingarritasun estrukturalari eskaini diogu, lan honetan, arreta gehien, eta elkarreragingarritasun kontzeptuala landu gabe utzi dugu. Elkarreragingarritasun kontzeptuala ebazteko, Chiarcosek berak OLiA delako ontologia komun baten erabilera defendatzen du (Chiarcos, 2012c). Ontologia horretan urteetan zehar sortu diren sailkapen ezagunenak integratu dituzte, elkarren arteko mapaketak egitea erraztuz. Horri helduz, gure proposamenei jarraituz bi anotazio-eskema guk aurkeztutako eredu abstraktuaren arabera egituratu ditugularik, interesgarria litzateke OLiA ontologiak erabiliz mapaketarekin aurrera egitea, anotazio linguistikoak eskema batetik bestera pasatzea lortu arte.

Bestalde, testuen prozesaketarako sistema bere osotasunean inplementatu dugun arren, horren inguruan ere badaude azter daitezkeen beste hainbat lerro. Batetik, saiatu gara dokumentu osoak prozesatu ordez, dokumentuak granularitate maila desberdinetan zatitu eta paralelizazio maila handitzearen ideia lantzen, baina oinarrizko esperimentuak baino ez ditugu egin. Etorkizunean, horretan sakondu eta granularitatez aldatzearen onurak sakonago aztertu nahi genituzke.

Prozesaketa-sistemarekin jarraituz, Storm erabili dugu prozesaketa banatu-

rako sistema bezala. Tesi-lan hau hasi zen unetik hona, arlo horrek eboluzio handia jasan du, eta gaur egun badaude Storm baino arrakasta handiagoa lortu duten sistemak. Horien artean Apache Spark da, agian, ezagunena. Beraz, gustatuko litzaiguke Spark gure sisteman integratu eta zer moduz dabilen aztertzea.

Azkenik, makina birtualetan oinarritu gara gure sistema terminal taldeetan banatzea errazteko. Arlo horretan ere aurrerapen handiak izan dira tesi-lan honek iraun duen bitartean, eta gaur egun oso erabiliak dira Docker eta Kubernetes sistemak (Bernstein, 2014). Makina birtualen exekuzioa astuna izan ohi da, sistema eragile osoa, paketeak eta liburutegi guztiak baitoaz makina birtual bakoitzean. Docker eta Kubernetes bezalako sistemek edukiontzien kontzeptua zabaldu dute. Edukiontzi batean, gure aplikazioaz gain, beharrezko liburutegiak eta sistema eragilearen zatirik garrantzitsuena besterik ez doaz. Horrela, makina birtualen antzeko soluzioa lortzen da, baina edukiontzi bakoitza askoz ere arinagoa da ohiko makina birtual oso bat baino.

Bibliografia

Aduriz I., Agirre E., Aldezabal I., Arregi X., Arriola J., Artola X., Gojenola K., Maritxalar A., Sarasola K., eta Urkia M. Morfeus: Euskararako analizatzaile morfosintaktikoa, 1999.

Aduriz I. eta Díaz de Ilarraza A. Morphosyntactic disambiguation and shallow parsing in computational processing of basque. *Anuario del Seminario de Filología Vasca*, 1–21, 2013.

Agerri R., Agirre E., Aldabe I., Altuna B., Beloki Z., Laparra E., López De Lacalle M., Rigau G., Soroa A., eta Urizar R. Newsreader project. *Procesamiento del Lenguaje Natural*, 53:155–158, 2014a.

Agerri R., Bermudez J., eta Rigau G. Ixa pipeline: Efficient and ready to use multilingual NLP tools. *LREC*, 2014 lib., 3823–3828, 2014b.

Alegria I., Artola X., Sarasola K., eta Urkia M. Automatic morphological analysis of basque. *Literary and Linguistic Computing*, 11(4):193–203, 1996.

Anderson E. eta Dvorsky M. Comparing cloud dataflow autoscaling to spark and hadoop, 2016. URL `https://cloud.google.com/blog/big-data/2016/03/comparing-cloud-dataflow-autoscaling-to-spark-and-hadoop`.

Andersson L. Natural Language Processing In A Distributed Environment: A comparative performance analysis of Apache Spark and Hadoop MapReduce, 2016.

Aranzabe M.J., Díaz de Ilarraza A., eta Gonzalez-Dios I. First approach to automatic text simplification in basque. *Proceedings of the Natural Language Processing for Improving Textual Accessibility (NLP4ITA) Workshop (LREC 2012), Istanbul, Turkey*, 1–8, 2012.

Artola X., Díaz de Ilarraza A., Soroa A., eta Sologaistoa A. Dealing with complex linguistic annotations within a language processing framework. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(5): 904–915, 2009.

Banko M., Cafarella M.J., Soderland S., Broadhead M., eta Etzioni O. Open Information Extraction from the Web. *IJCAI*, 7 lib., 2670–2676, 2007.

Bechhofer S. OWL: Web Ontology Language. *Encyclopedia of Database Systems*, 2008–2009. Springer, 2009.

Beloki Z., Rigau G., Soroa A., Fokkens A., Verstoep K., Vossen P., Rospocher M., Corcoglioniti F., Cattoni R., Verhoeven S., *et al.*. NewsReader System Design, Final Deliverable D2.3. 2016.

Berners-Lee T., Hendler J., eta Lassila O. The semantic web. *Scientific american*, 284(5):28–37, 2001.

Bernstein D. Containers and cloud: From LXC to Docker to Kubernetes. *IEEE Cloud Computing*, 1(3):81–84, 2014.

Bird S., Day D., Garofolo J., Henderson J., Laprun C., eta Liberman M. ATLAS: A flexible and extensible architecture for linguistic annotation. *Proceedings of the Second International Conference on Language Resources and Evaluation (LREC 2000)*, 2000.

Bird S. eta Liberman M. A formal framework for linguistic annotation. *Speech Annotation and Corpus Tools*, 33(1-2):23–60, 1999.

Bosma W., Vossen P., Soroa A., Rigau G., Tesconi M., Marchetti A., Monachini M., eta Aliprandi C. KAF: a generic semantic annotation format. *Proceedings of the GL2009 Workshop on Semantic Annotation*, 1–8. Citeseer, 2009.

Boulton D. eta Hammersley M. *Analysis of unstructured data*, 143–259. Sage, 2006.

Brickley D. eta Guha R. RDF Schema 1.1. Barne-txostena, W3C, 2014. URL `https://www.w3.org/TR/rdf-schema`.

Brugman H. eta Wittenburg P. The application of annotation models for the construction of databases and tools: Overview and analysis of MPI work since 1994. *IRCS Workshop on Linguistic Databases*, 2001.

Buneman P., Davidson S., Fernandez M., eta Suciu D. *Adding structure to unstructured data*, 336–350. Springer Berlin Heidelberg, Berlin, Heidelberg, 1997. ISBN 978-3-540-49682-3.

Bunt H. A methodology for designing semantic annotation languages exploiting semantic-syntactic isomorphisms. *Proceedings of the Second International Conference on Global Interoperability for Language Resources (ICGL2010)*, 29–46, 2010.

Carletta J., Kilgour J., O'Donnell T., Evert S., eta Voormann H. The NITE object model library for handling structured linguistic annotation on multimodal data sets. *Proceedings of the EACL Workshop on Language Technology and the Semantic Web (3rd Workshop on NLP and XML, NLPXML-2003)*, 2003.

Carlotto T., Beloki Z., Artola X., eta Soroa A. Interoperability of Annotation Schemes: Using the Pepper Framework to Display AWA Documents in the ANNIS Interface. *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC 2016)*. European Language Resources Association (ELRA), 2016. URL `http://dblp.uni-trier.de/db/conf/lrec/lrec2016.html#CarlottoBAS16`.

Chiarcos C. A generic formalism to represent linguistic corpora in RDF and OWL/DL. *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC 2012)*, 3205–3212, 2012a.

Chiarcos C. Interoperability of corpora and annotations. *Linked Data in Linguistics*. Springer, 2012b.

Chiarcos C. Ontologies of linguistic annotation: Survey and perspectives. *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC 2012)*, 303–310, 2012c.

Chiarcos C., Dipper S., Götze M., Leser U., Lüdeling A., Ritz J., eta Stede M. A flexible framework for integrating annotations from different tools and tagsets. *Traitement Automatique des Langues*, 49(2):271–293, 2008.

Chiarcos C., Hellmann S., eta Nordhoff S. Towards a Linguistic Linked Open Data cloud: The Open Linguistics Working Group. *Traitement automatique des langues*, 52(3):245–275, 2011.

Chowdhury G.G. Natural language processing. *Annual review of information science and technology*, 37(1):51–89, 2003.

Cunningham H., Maynard D., Bontcheva K., eta Tablan V. GATE: an architecture for development of robust HLT applications. *Proceedings of the 40th annual meeting on association for computational linguistics*, 168–175. Association for Computational Linguistics, 2002.

Dean J. eta Ghemawat S. MapReduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

Dena D., Bucicoiu M., eta Bardac M. A managed distributed processing pipeline with Storm and Mesos. *Networking in Education and Research, 2013 RoEduNet International Conference 12th Edition*, 1–6. IEEE, 2013.

Derivière J., Hamon T., eta Nazarenko A. A scalable and distributed NLP architecture for web document annotation. *Advances in Natural Language Processing*, 56–67. Springer, 2006.

Dipper S. XML-based Stand-off Representation and Exploitation of Multi-Level Linguistic Annotation. *Berliner XML Tage*, 39–50, 2005.

Dipper S., Faulstich L., Leser U., eta Lüdeling A. Challenges in modelling a richly annotated diachronic corpus of German. *Workshop on XML-based richly annotated corpora*, 21–29, 2004.

Dipper S. eta Götze M. Accessing heterogeneous linguistic data-generic XML-based representation and flexible visualization. *Proceedings of the 2nd Language and Technology Conference 2005*, 23–30, 2005.

Dorward S.M., Pike R., Presotto D.L., Ritchie D.M., Trickey H.W., eta Winterbottom P. The Inferno operating system. *Bell Labs Technical Journal*, 2(1):5–18, 1997.

Dutoit T. *An introduction to text-to-speech synthesis*, 3 lib. Springer Science & Business Media, 1997.

Eberle K., Eckart K., Heid U., eta Haselbach B. A Tool/Database Interface for Multi-Level Analyses. *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC 2012)*, 2912–2916, 2012.

Epstein E.A., Schor M.I., Iyer B., Lally A., Brown E.W., eta Cwiklik J. Making Watson fast. *IBM Journal of Research and Development*, 56(3.4): 15–1, 2012.

Erturk E. eta Shi H. Natural Language Processing using Hadoop and KOSHIK. *arXiv*, 2016.

Evans N., Asahara M., eta Matsumoto Y. Cocytus: parallel NLP over disparate data. *TAL*, 49(2):271–293, 2008. URL `http://www.atala.org/IMG/pdf/TAL-2008-49-2-10-Evans.pdf`.

Evert S., Carletta J., O'Donnell T., Kilgour J., Vögele A., eta Voormann H. The NITE object model. Barne-txostena, NITE project, 2003. URL `http://www.ltg.ed.ac.uk/NITE/documents/NiteObjectModel.v2.1.pdf`.

Exner P. eta Nugues P. KOSHIK- A Large-scale Distributed Computing Framework for NLP. *Proceedings of the 3rd International Conference on Pattern Recognition Applications and Methods*, 463–470, 2014. ISBN 978-989-758-018-5.

Facebook. Under the Hood: Scheduling MapReduce jobs more efficiently with Corona, 2012. URL `https://www.facebook.com/notes/facebook-engineering/`

under-the-hood-scheduling-mapreduce-jobs-more-efficiently-with-corona/10151142560538920/.

Farrar S. eta Langendoen D.T. A linguistic ontology for the semantic web. *Glot International*, 7(3):97–100, 2003.

Feng M., Xiang B., eta Zhou B. Distributed deep learning for question answering. *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, 2413–2416. ACM, 2016.

Ferrucci D., Brown E., Chu-Carroll J., Fan J., Gondek D., Kalyanpur A.A., Lally A., Murdock J.W., Nyberg E., Prager J., *et al.*. Building Watson: An overview of the DeepQA project. *AI magazine*, 31(3):59–79, 2010.

Ferrucci D. eta Lally A. UIMA: an architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering*, 10(3-4):327–348, 2004.

Fokkens A., Soroa A., Beloki Z., Ockeloen N., Rigau G., van Hage W.R., eta Vossen P. NAF and GAF: Linking linguistic annotations. *Proceedings 10th Joint ISO-ACL SIGSEM Workshop on Interoperable Semantic Annotation*, 9–16, 2014.

Gamallo P., Pichel J.C., Garcia M., Abuín J.M., eta Pena T.F. Análisis morfosintáctico y clasificación de entidades nombradas en un entorno Big Data. *Procesamiento del Lenguaje Natural*, 53:17–24, 2014.

Gómez-Pérez P., Phan T.N., eta Küeng J. Agricultural Knowledge Extraction from Text Sources Using a Distributed MapReduce Cluster. *Database and Expert Systems Applications (DEXA), 2016 27th International Workshop on*, 29–33. IEEE, 2016.

Grishman R. TIPSTER Text Architecture Design. Barne-txostena, New York University, 1998.

Hellmann S., Lehmann J., eta Auer S. Linked-data aware uri schemes for referencing text fragments. *Knowledge Engineering and Knowledge Management*, 175–184. Springer, 2012.

Hellmann S., Lehmann J., Auer S., eta Brümmer M. Integrating NLP using linked data. *International Semantic Web Conference*, 98–113. Springer, 2013.

Hernandez A.F.R. eta Garcia N.Y.G. Distributed processing using cosine similarity for mapping Big Data in Hadoop. *IEEE Latin America Transactions*, 14(6):2857–2861, 2016.

Ide N., Bonhomme P., eta Romary L. An XML-based Encoding Standard for Linguistic Corpora. *Proceedings of the Second International Conference on Language Resources and Evaluation (LREC 2000)*, 825–830, 2000.

Ide N. eta Bunt H. Anatomy of annotation schemes: mapping to GrAF. *Proceedings of the Fourth Linguistic Annotation Workshop*, 247–255, 2010.

Ide N. eta Pustejovsky J. What does interoperability mean, anyway? Toward an operational definition of interoperability for language technology. *Proceedings of the Second International Conference on Global Interoperability for Language Resources*, 2010.

Ide N., Pustejovsky J., Calzolari N., eta Soria C. The SILT and FLaReNet international collaboration for interoperability. *Proceedings of the Third Linguistic Annotation Workshop*, 178–181, 2009.

Ide N. eta Romary L. A common framework for syntactic annotation. *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, 2001.

Ide N. eta Romary L. Standards for language resources. *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC 2002)*, 2002.

Ide N. eta Romary L. International standard for a linguistic annotation framework. *Natural Language Engineering*, 10(3-4):211–225, 2004a.

Ide N. eta Romary L. A registry of standard data categories for linguistic annotation. *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC 2004)*, 135–138, 2004b.

Ide N. eta Romary L. Representing linguistic corpora and their annotations. *Proceedings of the 5th International Conference on Language Resources and Evaluation Conference (LREC 2006)*, 2006.

Ide N. eta Suderman K. GrAF: A graph-based format for linguistic annotations. *Proceedings of the Linguistic Annotation Workshop*, 2007.

Ide N. eta Suderman K. Bridging the gaps: interoperability for GrAF, GATE, and UIMA. *Proceedings of the Third Linguistic Annotation Workshop*, 27–34, 2009.

Kemps-Snijders M., Windhouwer M., Wittenburg P., eta Wright S.E. ISOcat: Corralling Data Categories in the Wild. *Proceedings of the 6th International Conference on Language Resource and Evaluation (LREC 2008)*, 2008.

Koehn P., Hoang H., Birch A., Callison-Burch C., Federico M., Bertoldi N., Cowan B., Shen W., Moran C., Zens R., *et al.*. Moses: Open source toolkit for statistical machine translation. *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, 177–180. Association for Computational Linguistics, 2007.

Lee T., Kim H., Rhee K.H., eta Shin S.U. Implementation and performance of distributed text processing system using hadoop for e-discovery cloud service. *Journal of Internet Services and Information Security (JISIS)*, 4 (1):12–24, 2013.

Lin J. eta Dyer C. Data-intensive text processing with MapReduce. *Synthesis Lectures on Human Language Technologies*, 3(1):1–177, 2010.

Ma X., Lee H., Bird S., eta Maeda K. Models and tools for collaborative annotation. *Proceedings of the Third International Conference on Language Resources and Evaluation Conference (LREC 2002)*, 2002.

Maeda K., Bird S., Ma X., eta Lee H. Creating annotation tools with the Annotation Graph Toolkit. *Proceedings of the Third International Conference on Language Resources and Evaluation Conference (LREC 2002)*, 2002.

Manning C. Understanding human language: Can NLP and deep learning help? *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, 1–1. ACM, 2016.

Manning C.D., Schütze H., *et al.*. *Foundations of statistical natural language processing*, 999 lib. MIT Press, 1999.

Manning C.D., Surdeanu M., Bauer J., Finkel J.R., Bethard S., eta McClosky D. The Stanford CoreNLP natural language processing toolkit. *ACL (System Demonstrations)*, 55–60, 2014.

Manyika J., Chui M., Brown B., Bughin J., Dobbs R., Roxburgh C., eta Byers A.H. Big Data: The next frontier for innovation, competition, and productivity. Barne-txostena, McKinsey Global Institute, 2011.

Marz N. eta Warren J. *Big Data: Principles and best practices of scalable realtime data systems*. Manning Publications Co., 2015.

McCrae J., Montiel-Ponsoda E., eta Cimiano P. Integrating WordNet and Wiktionary with Lemon. *Linked Data in Linguistics*, 25–34. Springer, 2012.

McCrae J., Spohr D., eta Cimiano P. Linking lexical resources and ontologies on the semantic web with lemon. *The semantic web: research and applications*, 245–259. Springer, 2011.

McCreadie R., Macdonald C., Ounis I., Osborne M., eta Petrovic S. Scalable distributed event detection for twitter. *Big Data, 2013 IEEE International Conference on*, 543–549. IEEE, 2013.

Mendes P.N., Jakob M., García-Silva A., eta Bizer C. DBpedia Spotlight: shedding light on the web of documents. *Proceedings of the 7th international conference on semantic systems*, 1–8. ACM, 2011.

Meystre S.M., Lee S., Jung C.Y., eta Chevrier R.D. Common data model for natural language processing based on two existing standard information models: CDA+GrAF. *Journal of Biomedical Informatics*, 45(4):703–710, 2012.

Miller G.A. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995.

Mittal S., Joshi K.P., Pearce C., eta Joshi A. Parallelizing natural language techniques for knowledge extraction from cloud service level agreements. *Big Data (Big Data), 2015 IEEE International Conference on*, 2831–2833. IEEE, 2015.

Nesi P., Pantaleo G., eta Sanesi G. A Distributed Framework for NLP-Based Keyword and Keyphrase Extraction From Web Pages and Documents. *DMS*, 155–161, 2015.

Neumann A., Ide N., eta Stede M. Importing MASC into the ANNIS linguistic database: A case study of mapping GrAF. *Proceedings of the seventh linguistic annotation workshop (LAW)*, 98–102, 2013.

Neumeyer L., Robbins B., Nair A., eta Kesari A. S4: Distributed stream computing platform. *2010 IEEE International Conference on Data Mining Workshops*, 170–177. IEEE, 2010.

Nivre J., de Marneffe M.C., Ginter F., Goldberg Y., Hajic J., Manning C.D., McDonald R., Petrov S., Pyysalo S., Silveira N., *et al.*. Universal dependencies v1: A multilingual treebank collection. *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*, 1659–1666, 2016.

Otegi A., Ezeiza N., Goenaga I., eta Labaka G. A Modular Chain of NLP Tools for Basque. *International Conference on Text, Speech, and Dialogue*, 93–100. Springer, 2016.

Padró L. eta Stanilovsky E. Freeling 3.0: Towards wider multilinguality. *Proceedings of the 8th International Conference on Language Resources and Evaluation Conference (LREC 2012)*, 2012.

Padró L. eta Turmo J. TextServer: Cloud-based multilingual natural language processing. *Data Mining Workshop (ICDMW), 2015 IEEE International Conference on*, 1636–1639. IEEE, 2015.

Pareja-Lora A. eta de Cea G.A. Ontology-based interoperation of linguistic tools for an improved lemma annotation in Spanish. *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC 2010)*, 2010.

Paris M. eta Sabena G. Hermes: A Distributed-Messaging Tool for NLP. *Machine Learning, Optimization, and Big Data: Second International Workshop, MOD 2016, Volterra, Italy, August 26-29, 2016, Revised Selected Papers*, 10122 lib., page 402. Springer, 2016.

Rabiner L.R. eta Juang B.H. Fundamentals of speech recognition. 1993.

Ravi S. eta Diao Q. Large scale distributed semi-supervised learning using streaming approximation. *Proceedings of AISTATS*, 2016.

Schuurman I. eta Windhouwer M. Explicit semantics for enriched documents. What do ISOcat, RELcat and SCHEMAcat have to offer. *2nd Supporting Digital Humanities conference*, 2011.

Semberecki P. eta Maciejewski H. Distributed classification of text documents on Apache Spark platform. *International Conference on Artificial Intelligence and Soft Computing*, 621–630. Springer, 2016.

Shahrivari S. Beyond batch processing: towards real-time and streaming big data. *Computers*, 3(4):117–129, 2014.

Shvachko K., Kuang H., Radia S., eta Chansler R. The hadoop distributed file system. *2010 IEEE 26th symposium on mass storage systems and technologies (MSST)*, 1–10. IEEE, 2010.

Sonntag D. Distributed NLP and machine learning for question answering grid. *Proceedings of the workshop on Semantic Intelligent Middleware for the Web and the Grid at ECAI*, 2004.

Sperberg-McQueen C. eta Burnard L. TEI P4: Guidelines for Electronic Text Encoding and Interchange–XML-compatible version. 2001, 2001.

Sun D. eta Gao S. Scalable-DSP: a high scalable distributed storage and processing system for unstructured data in big data environments. *Proceedings of the Australasian Computer Science Week Multiconference*, page 41. ACM, 2017.

Sverdlik Y. Google Dumps MapReduce in Favor of New Hyper-Scale Analytics System, 2014. URL `http://www.datacenterknowledge.com/archives/2014/06/25/google-dumps-mapreduce-favor-new-hyper-scale-analytics-system/`.

Teich E., Hansen S., eta Fankhauser P. Representing and querying multilayer corpora. *Proceedings of the IRCS Workshop on Linguistic Databases*, 228–237, 2001.

Unger C., McCrae J., Walter S., Winter S., eta Cimiano P. A lemon lexicon for DBpedia. *Proceedings of the 2013th International Conference on NLP & DBpedia*, 1064 lib., 2013.

Van Gompel M. eta Reynaert M. FoLiA: A practical XML Format for Linguistic Annotation. A descriptive and comparative study. *Computational Linguistics in the Netherlands Journal*, 3:63–81, 2013.

Vavilapalli V.K., Murthy A.C., Douglas C., Agarwal S., Konar M., Evans R., Graves T., Lowe J., Shah H., Seth S., *et al.*. Apache Hadoop YARN: Yet another resource negotiator. *Proceedings of the 4th annual Symposium on Cloud Computing*, page 5. ACM, 2013.

Windhouwer M. RELcat: a Relation Registry for ISOcat data categories. *Proceedings of the 8th International Conference on Language Resources and Evaluation Conference (LREC 2012)*, 3661–3664, 2012.

Windhouwer M., Kemps-Snijders M., eta Wright S.E. Referencing ISOcat data categories. *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC 2010)*, 2010.

Windhouwer M. eta Wright S.E. Linking to linguistic data categories in ISOcat. *Linked Data in Linguistics*, 99–107. Springer, 2012.

Wu H., Fei Z., Dai A., Sammons M., Roth D., eta Mayhew S.D. ILLINOISCLOUDNLP: Text analytics services in the cloud. *LREC*, 14–21, 2014.

Yu W. eta Chen J. The state-of-the-art in web-scale semantic information processing for cloud computing. *arXiv preprint arXiv:1305.4228*, 2013.

Zajac R., Casper M., eta Sharples N. An open distributed architecture for reuse and integration of heterogeneous NLP components. *Proceedings of the fifth conference on Applied natural language processing*, 245–252. Association for Computational Linguistics, 1997.

Zeldes A., Lüdeling A., Ritz J., eta Chiarcos C. ANNIS: A search tool for multi-layer annotated corpora. *Proceedings of Corpus Linguistics 2009*, 2009.

Zipser F. eta Romary L. A model oriented approach to the mapping of annotation formats using standards. *Workshop on Language Resource and Language Technology Standards (LREC 2010)*, 2010.