Universidad del País Vasco — Euskal Herriko Unibertsitatea

Izen-entitate eta Sentimenaren Analisia Frantsesez, Turismoaren Domeinurako Hurbilpena

Egilea: Andoni Azpeitia
Tutorea: Aitor Soroa

Hizkuntzaren Azterketa eta Prozesamendua

Hizkuntzaren Azterketa eta Prozesamendua Masterreko titulua lortzeko bukaerako proiektua

2014eko Iraila

Sailak: Lengoaia eta Sistema Informatikoak, Konputagailuen Arkitektura eta Teknologia, Konputazio Zientziak eta Adimen Artifiziala, Euskal Filologia, Elektronika eta Telekomunikazioak.

Laburpena

Master bukaerako proiektu honetan, hizkuntza teknologia bidez garatutako hainbat oinarrizko tresnei esker oso aplikazio ahaltsuak nola eraiki daitezkeen azalduko dugu, esate baterako, erabiltzaileek zerbitzu turistikoei buruz duten iritzia ezagutzeko. Aipatutako oinarrizko tresnak tokenizatzaileak, etiketatzaile-morfologikoak, entitate-izenen identifikatzaile eta sailkatzaileak eta polaritate etiketatzailea izan daitezke besteak beste. Horretaz gainera, teknologia horren berrerabilpena eta hedapena kontutan izanik, estandarra den software librearen bidez garatu dira tresna guztiak.

Abstract

In this thesis, we explore the use, extension and combination of Natural Language Processing (NLP) tools for the development of powerful applications, such as tourism-oriented services that help users by extracting useful information from online reviews. The NLP tools we cover include tokenizers, part-of-speech taggers, named entity recognizers and classificators, and polarity taggers. With reuse and possible extensions in mind, all software presented in this work has been developed under an open-source approach.

Gaien aurkibidea

Taulen zerrenda

Irudien zerrenda

1 Proiektuaren definizioa

Sentimenduaren analisia (Sentiment Analysis) eta iritzien erauzketa (Opinion Mining) azken urteotako ikerlerro garrantzitsuak bihurtu dira. Helburua igorlea zein den, zeri buruzko informazioa adierazten den eta noiz idatzi den eta zertarako jakitea da. Horrela, industria berri bat sortu da sare sozialetan zehar sentimendu analisirako zerbitzuak eskaintzen dituena. Nahiz eta zerbitzu gehienek hizkuntza bakarrerako irtenbide orokorrak eskaini, konpainia batzuek turismoaren domeinurako zerbitzu espezifikoak eskaintzen dituzte. Turismoari buruzko online kontsultak oso erabiliak dira bai erabiltzaile eta baita turismo agentzien aldetik.

Hizkuntzaren Azterketa eta Prozesamendua (HAP) master bukaerako proiektu honetan, turismo-zerbitzuei buruz erabiltzaileek idatzitako kritikak analizatu ahal izateko tresnak garatuko dira. Aipatutako arazoa nahiko potoloa izanik frantsesera mugatuko gara.

Orain arte turismoaren domeinuari buruz hitz egin dugu, baina benetako helburua oinarrizko hizkuntza-teknologia garatzea eta lizentzia librepean komunitatearen eskura uztea da. Turismoa aukeratu dugu aplikazio-domeinu bezala gaur egungo merkatuan behar bat dagoelako. Baldintza guztiak bete ahal izateko, funtzionalitate guztiak izaera hedagarri batekin diseinatuko dira. Honek bi ezaugarri eskatzen ditu: funtzionalitateak hizkuntza berrietara moldatzeko aukerak ematea, eta funtzionalitateek nolabaiteko izaera independentea izatea aldi berean ezaugarri komunak eskainiz.

Proiektu honen esparruan garatuko diren tresnak elkarren artean lan egiteko diseinatutako web-zerbitzuak izango dira. Web-zerbitzu bakoitzak oinarrizko LNP (Lengoaia Naturalaren Prozesamendua) modulu bat izango du bere barne sarrerako testua (turismo-zerbitzu bati buruzko iruzkin bat) prozesatu ahal izateko. Web-zerbitzu bakoitza modu independentean exekutatzeko gauza izango da eta baita gainerako web-zerbitzuekin elkarlanean aritzeko. Honi esker bi helburu betetzen dira: oinarrizko LNP tresnak garatzea, eta baita testuetan dagoen informazioa erauztea.

Honako hauek dira garatu beharreko web-zerbitzuak:

- **hizkuntza-detektatzailea**. Lehenago esan dugu tresna guztiak frantseserako garatuko direla, hala ere, etorkizunari begira ezinbestekoa da hizkuntza-detektatzaile bat inplementatzea. Hizkuntza-detektatzaileari esker modulu guztiek jakingo dute zein erregela edota modelo estatistiko erabili behar dituzten.

- **Esaldi-banatzailea**. Modulu honi esker, testu dokumentu bat izanda dokumentua hainbat esalditan banatzeko gauza izango gara, eta horrela, moduluaren irteeran lerro bakoitzeko esaldi bat izango dugu.

- **Tokenizatzailea**. Tokenizatzaileari esker sarrerako dokumentua tokenak detektatzeko gauza izango gara. Token guztiak zuriune karakterearen bidez banatuta egongo dira.

- **Etiketatzaile-morfologikoa**. Modulu honetan sarrerako token bakoitzari dagokion kategoria gramatikala ezarriko zaio.
- **Entitate-izenen detektatzaile eta sailkatzailea**. Kategoria gramatikalak identifikatu eta gero, dokumentuan zein motatako izen-entitateak ditugun jakiteko gauza izango gara. Modulu hau oso garrantzitsua izango da, izan ere, dokumentu bateko informazioa erauzteko orduan ez bagara gauza zerri buruz hitz egiten den jakiteko, aurretiko lan guztiak ez du ezertarako balioko. Izen-entitateak hiru motetakoak izan daitezke: pertsonak, lekuak edo erakundeak.
- **Polaritate etiketatzailea**. Modulu honen bidez, dokumentuan azaltzen diren izen-entitateen inguruan iritzi ona, neutrala edo txarra adierazten den jakingo dugu. Modulu honetarako, baliabide semantiko bat garatu beharko da polaritatea adierazten duten hitzak detektatu ahal izateko.

Web-zerbitzu guztiek beraien artean komunikatu ahal izateko web-zerbitzu batetik bestera doan dokumentu bat erabiliko dute, dokumentu horretan web-zerbitzu bakoitzak erauzitako informazio linguistikoa kodetuta egongo da. Web-zerbitzu bakoitzaren barneko oinarrizko LNP modulu guztiak teknologia ezberdinen bidez garatuko denez, dokumentuaren formatua estandarra izateak abantaila handiak ditu, LNP moduluek dokumentuko informazioa *ulertzeko* egokitzapen tresna estandarrak erabili ditzakete eta. Dokumentuaren formatu gisa XML-pean oinarritako KAF[1] (Kyoto Annotation Framework) formatua aukeratu. KAF formatua, LNP (Lengoaia Naturalize Prozesamendua) ezaugarriak kodetzeko bereziki prestatuta dago eta XML bidez idatzita dagoenez, programazio lengoaia guztiak KAF formatua prozesatzeko gai dira, hala ere 3.2 kapituluan sakonago ikusiko dugu nolakoa den.

Proiektu honen beste helburu garrantzitsu bat bere izaera irekia da. Proiektua bukatu eta gero, garatutako tresna guztiak hobetzeko edota beste aplikazio-domeinuetara moldatzeko oso garrantzitsua da estandarra den software librea erabiltzea. Arrazoi horregatik XML edo OpenNLP[2] bezalako teknologia estandarra erabiliko da garapen prozesuan zehar. Proiektu honetarako tresnak *Apache License, Version 2.0*[3] lizentziapean banatzen dira.

1.1 Motibazioa

Interneten aurkitu daitezkeen erabiltzaileen iritziak gero eta garrantzitsuagoak dira produktu eta zerbitzuen ebaluazioan, izan ere, sektore askotan funtsezkoak dira erabiltzaile berrien erabakietan. Azken ikerlanen arabera, erabiltzaileen %30-ak beraien erosketei buruzko iruzkinak zabaltzen dituzte internet bidez, eta hauen %70-ak informazioa bilatu du sarean zehar.

[1] `https://github.com/opener-project/kaf/wiki/KAF-structure-overview`
[2] `http://opennlp.apache.org/`
[3] `http://www.apache.org/licenses/LICENSE-2.0`

Sentimenduaren analisia (Sentiment Analysis) eta iritzien erauzketa (Opinion Mining) azken urteotako ikerlerro garrantzitsuak bihurtu dira. Helburua igorlea zein den, zeri buruzko informazioa adierazten den, noiz idatzi den eta zein helbururekin jakitea da. Horrela, industria berri bat sortu da sare sozialetan zehar sentimendu analisi zerbitzuak eskaintzen dituena. Arrazoi horregatik, konpainia batzuek turismoaren domeinurako zerbitzu espezifikoak eskaintzen dituzte. Hala ere, sektore eta industria berritzaile askotan gertatu ohi den bezala, merkatu horretan parte hartu ahal izateko oinarrizko teknologiak lortu edo garatzea garestia da, eta kasu askotan enpresa txiki eta ertainek ez dituzte baliabide nahikoak inbertsio horri aurre egin ahal izateko. Proiektu honetan erakusten den moduan, enpresek duten behar teknologikoari erantzuteko gai izan gintezke oinarrizko tresnak garatuz eta modu eraginkorrean erabiliz. Proiektu hau testuinguru honetan kokatzen da. Arazoari erantzun egokia eman ahal izateko Open Source komunitatearen barne ezarritako tresna eta teknikak erabiliko ditugu garatutako teknologia hedatu ahal izateko, luzarora begira erabilgarria izan liteke proiektu hau beste aplikazio domeinutan aplikatzea.

1.2 OpeNER

Proiektua dagokien testuinguruan kokatu ahal izateko OpeNER[4] proiektuaren nondik norakoak azaltzea ezinbestekoa da, master bukaerako proiektu honetan azaltzen den lana OpeNER proiektuaren parte baita. OpeNER bi urteko proiektu europarra da (European Commission 7th Framework Programme, grant agreement 296451), eta bertan Donostiako Vicomtech-IK4[5] fundazioa eta IXA[6] EHUko ikerketa taldea, Holandako Olery[7] konpainia eta VU[8] Unibertsitatea, eta Italiako Synthema[9] konpainia eta CNR[10] ikerketa zentroa inplikatuta daude.

OpeNER proiektuaren barne izen-entitateen detektatzaile eta sailkapena (NERC) eta sentimendua analizatu ahal izateko beharrezko baliabide-lexikoak garatu dira, baliabide horiek domeinuaren arabera normalizatuta egonik. Sentimenduaren analisiari dagokionez, OpeNER master bukaerako proiektua baino aurrerago doa: turismoaren domeinuko entitateei egokitutako baliabide-lexikoak eraiki dira, eta baita beste domeinuetara egokitu ahal izateko tresnak. Gainera, detektaturiko polaritateak entitate bakoitzari lotzen zaizkio.

OpeNER-en beste ezaugarri garrantzitsu bat detektaturiko entitate-izenen eta datu-multzoen arteko estekak dira. Demagun etorkizunean izen-entitateen datu-multzoa hizkuntza berri batera itzultzen dugula, jadanik entitateak estekatzeko tresna baldin badugu, nahiko esfortzu txikiarekin garapen handiagoa duten eta ez dutenen hizkuntzen arteko aldea murriztu

[4] http://www.opener-project.org/
[5] http://www.vicomtech.org/
[6] http://ixa.si.ehu.es/Ixa
[7] http://www.olery.com/
[8] www.let.vu.nl
[9] http://www.synthema.it/
[10] http://www.iit.cnr.it/

liteke.

OpeNER proiektuak gaztelania, ingeles, frantsesa, alemana, nederlandera eta italiera landuko ditu. Proiektuaren ikerketa aplikazio-domeinu orokor batera bideratuko da, eta gero, turismoaren sektorera egokitu eta balioztatuko da.

Hauexek dira OpeNER proiektuaren helburuak:

- Gaur egungo hizkuntza-baliabideak berrantolatzea eta kulturalki normalizatutako sentimendu lexikoi eleanitz bat garatzea turismoaren domeinurako hedatuko dena. Baliabideak gaztelania, ingeles, frantses, aleman eta nederlanderarako sortuko dira.
- Izen-entitateen detekzio eta sailkapena aurreko puntuan aipatutako sei hizkuntzetan. Beste hizkuntzetara hedatzea ahalbidetuko da entitateak Wikipedia[11] edo beste baliabide eleanitzekin estekatuz.
- Proiektuaren emaitzetan oinarritutako sentimendu eta iritzi analisirako tresnak garatzea lizentzia librepean.
- Proiektuaren emaitzak zuzentzea batez ere turismoaren domeinurako.
- Proiektuaren emaitzak epe luzaroan iraunkorrak eta ekonomikoki bideragarriak direla bermatuko duten metodoak ikertu eta probatzea.

OpeNER proiektuaren helburuak lortu ahal izateko gaur egungo teknologia estandarra eta hizkuntza-baliabideak berrerabiliko da, ahal den neurrian lizentzia librea dutenak aukeratuko dira. Funtsezkoa da proiektuaren iraupenetik kanpo teknologia guztiaren erabilpen eta hedapenak ahalik eta esfortzu eta kostu txikiena eskatzea etorkizuneko erabiltzaileei.

LREC 2014 konferentziako *Come Hack with OpeNER*[12] workshop-ean frogatu zen bezala OpeNER-en baitan garatutako tresnekin *jolastuz* epe laburrean hainbat aplikazio interesgarri egin daitezkeela frogatu zen (Azpeitia et al.; Pupi et al., 2014; Cresci et al.; García-Pablos et al.; Atserias et al.).

Irudia 1: OpeNER proiektuaren logoa

[11] http://www.wikipedia.org/
[12] https://www.facebook.com/events/647970071943415/?context=create&source=49

2 Aurrekariak

Atal honetan proiektuaren aurrekariak ikusiko ditugu. Lehendabizi proiektuaren aplikazio-domeinua deskribatuko dugu (2.1. atala); jarraian artearen egoera aztertuko dugu (2.2. atala); eta azkenik proiektu honekin erlazionatutako proiektu Europarrak ikusiko ditugu (2.3. atala).

2.1 Aplikazio-domeinua

Nahiz eta proiektu honetan garatutako teknologia edozein alorretan egokitzeko pentsatuta egon turismoaren sektorea aukeratu da lehendabiziko aplikazio domeinu gisa, proiektu honen ebaluaziorako eta baliagarritasuna frogatzeko oso aukera egokia baita. Bidaia eta turismorako online zerbitzuen hazkunde azkarrak turista eta turismo-agentzietarako negozio aukera berritzaileak eskaintzen ditu, izan ere, turistek ahoz-aho beraien esperientziak ezagutzera emateko internetek aukera ezin hobea eskaintzen die.

Hainbat ikerketek frogatzen duten bezala gero eta turista gehiagok jotzen dute beste turisten iritzietara, horrela erabakitzen dute eskaintza turistiko batek beraien beharrak beteko dituen ala ez. Turistentzako informazio-iturri nagusiak beste turisten iritziak biltzen dituzten online zerbitzuak (TripAdvisor edo WikiTravel), online turismo-agentziek (OTA) eskainitako web-orriak (Expedia), blogak eta sare-sozialak (Facebook, Twitter) eta multimedia edukiak zabaltzeko zerbitzuak (Youtube) dira.

Turismo agentziek beraien produktuak aukeratzeko orduan turisten iritziak garrantzia handia du. Hotel eta jatetxeen jabeentzako bezeroen esperientzia nolakoa izan den jakitea asko axola zaie. Hala ere, sarean zehar dauden milioika multimedia eduki aztertzea ez da posible.

TripAdvisor web-orriak hilabetero 30 milioi bisita ditu, hauen artean iritziak idazten dituzten eta irakurtzen dituzten turistak eta testu bakoitza positibo edo negatiboki markatzen dituztenak ditugu. Informazio guzti hori automatikoki prozesatu ahal izateko beharra dago beraz, eta horretarako proiektu hau oso baliagarria izan liteke.

2.2 Artearen egoera

Atal honetan zehar proiektu honetan garatu beharreko tresnen artearen egoerari buruz arituko gara. Garapen zientifikoari begira oso garrantzitsua da jakitea aurretik zein lan egin diren gure ikerlerro aproposenetan oinarritzeko.

HAP masterra

2.2.1 Esaldi-banatzaileak eta tokenizatzaileak

Testu bat tokenizatzeak bertan agertzen diren hitzak prozesamendurako oinarrizko unitate lexikoetan deskonposatzea esan nahi du. Hona hemen adibide bat:

```
J'aime voyager à Paris.
```

Tokenizatu ondoren:

```
J' aime voyager à Paris .
```

Lengoaia naturalaren prozesamenduan edozein tresnek sarrerako testuak esaldika banatuta egotea espero du, eta modu berean esaldi bakoitzak tokenizazio egoki bat eskatzen du. Mundu errealeko testuek ez dute horrelako egiturarik eta segmentazio lanetarako ohikoa da heuristikoak erabiltzea. Berrien domeinuan eskuz zenbait patroi definitu ziren (Grefenstette eta Tapanainen, 1994). Tokenizazioari dagokionez karaktere-kateak hartzen dira karaktere berezi bat aurkitu arte (normalean puntuazio ikurrak, parentesiak, komatxoak, ...). Erregela horiek egokiak izaten dira domeinu orokorretan, baina biomedikuntzan adibidez, erroreak daude (karaktere berezi asko anbiguoak dira entitate-izenen laburduretan eta erreferentzia bibliografikoetan formula kimikoetan eta abarretan agertu litezke (Grover et al., 2006)).

Erregela konplexuak idatzi beharrean bada beste hurbilpen bat azaldutako arazoak saihesteko balio duena, gainbegiratutako ikasketa automatikoa (ML). Ikasketa automatikoari esker beste hizkuntza edo domeinuetarako metodo berbera erabil daiteke, behar den gauza bakarra etiketaturiko corpus bat da. Gainera, metodo hauen errendimendua erregeletan oinarritutakoekin lortzen dena baina altuagoa izan daiteke (Palmer eta Hearst, 1997). Hala ere ikasketa automatikoak baditu bere ahuleziak, kontuan izan behar da tresna on bat garatu ahal izateko beharrezkoa dela detektatu nahi ditugun ezaugarri guztiak corpusean agertzea, bestela ez gara gai izango esaldi edota token banaketa zuzen bat egiteko.

Corpus egokia lortzeko arazoak direla medio, proiektu honetan itzulketa automatikorako Moses[13] tresnaren barne dauden esaldi-banatzaile eta tokenizatzaileak erabiliko ditugu. Domeinu orokorretarako eta hamasei hizkuntzetarako aproposak diren erregelak erabiltzen ditu, eta horretaz gainera, laburdura lista handiak eskaintzen ditu hizkuntza gehienetarako. Proiektuaren hedagarritasunari begira beste abantaila nabarmen bat erregelak gure beharretara egokitzeko erraztasuna da.

2.2.2 Etiketatzaile-morfologikoa

Etiketatzaile-morfologikoek testuaren analisi-morfologikoa egiten dute token bakoitzari dagokion etiketa ezarriz. Analisi morfologikoa nahi adina konplikatu liteke, kategoria-gramatikalarekin nahikoa izan dezakegu edo kategoria ez ezik pertsona eta numeroa jakitea

[13] `http://www.statmt.org/moses/`

garrantzitsua izan liteke.

Azken urteotan etiketatzaile-morfologikorako sistema ugari proposatu dira hizkuntza ezberdinetarako. Sistema onenen artean ikasketa gainbegiraturako metodoak inplementatzen dituztenak ditugu (Manning eta Schütze, 1999). Ingelera eta beste hizkuntza europarretarako gizakion ahalmenetik gertu dauden tresnak garatu dira, gainera tresna gehienek ez dute aparteko baliabide lexikorik erabiltzen, nahikoa dute entrenamendu corpusaren bidez ikasitako hiztegi batekin (Hajič, 2000). Gure kasuan badugu entrenamendurako corpusa, French TreeBank[14] (Abeillé et al., 2003) delakoa, beraz ikasketa gainbegiratua oso metodologia aproposa dirudi.

Azken urteotan MaxEnt (Maximum Entropy) sekuentzia baldintzatu etiketatzaileak oso erabiliak dira etiketatzaile-morfologikoak entrenatzeko (Søgaard, 2010; Tsuruoka et al., 2005; Dalal et al., 2006). MaxEnt modeloen abantaila nagusia elkarren artean gainjartzen diren ezaugarri ezberdinen konbinatzeko ahalmena da sailkatzaileen arteko dependentzia galdu gabe. Beste abantaila bat entrenamendu abiadura altua da.

MaxEnt algoritmoak inplementatzen dituzten aplikazioak badaude eskura lizentzia librepean, horien artean OpenNLP dugu. OpenNLP-ren bidez modeloak entrenatzea eta erabiltzea oso erraza da, eta etorkizunari begira frantseserako tresna garatu eta gero beste hizkuntzetara moldatzeko modelo berriak entrenatzea da egin beharreko lan bakarra.

2.2.3 Entitate-izenen detektatzaile eta sailkatzailea

Entitateen-izenak detektatu eta sailkatzea testu batean azaltzen diren pertsona-izenak, erakundeak, kokalekuak, orduen adierazpenak, kopuruak, balio monetarioak, ehunekoak, eta abar bezalako kategoria aurredefinituetan elementu atomikoak aurkitzea da.

Azken urteotan ikerlan asko egin da entitate-izenen detekzio eta sailkapenaren (NERC) inguruan. Izen-entitateak bilatzeko lau teknika nagusi erabili izan ohi dira: ezagutzan oinarritutakoak, ikasketa gainbegiratua eta ikasketa erdi-gainbegiratuan oinarritutako sistemak.

Ezagutzan oinarritutako sistemak izan ziren hasierako ikerketetan garatutakoak. Funtsean automata finitu deterministetan eta erregeletan oinarritzen dira. Ikerlan hauek batez ere MUC-6 eta MUC-7 konferentzietan eta erlazionatutako ondorengo lanetan (Appelt et al., 1995; Weischedel, 1995; Krupka eta Hausman, 1998; Aone et al., 1998; Mikheev et al., 1998, 1999) egin ziren ingelerarako. Naiz eta ezagutzan oinarritutako metodoak lehenak izan oraindik gaur egun erabiltzen dira, horren isla 2003. urtean Budi eta Bressan-en lanean proposatu zuten metodoa da informazioaren erauzketan erabiltzen dena (Budi eta Bressan, 2003). Normalean metodo hauekin lortutako errendimendua altua da, baina lan nekeza da ezagutzaren inplementazioa eta ez da erraza sistemak beste hizkuntzetara egokitzea.

[14] `http://www.llf.cnrs.fr/Gens/Abeille/French-Treebank-fr.php`

Ikasketa gainbegiratua da gaur egun gehien erabiltzen de metodologia. NERC teknika honen barruan badira zenbait hurbilpen ezberdin, HMM-ak (Hidden Markov Model) (Bikel et al., 1997), Decision Tree-ak (Sekine, 1998) eta Maxent (Maximum Entropy) modeloak Borthwick et al. (1998); Borthwick (1999) dira arrakasta handiena dutenak. Metodo hauen arazo nagusia etiketatutako corpus baten beharra da, garatutako tresnaren zuzentasuna neurri handi batean erabilitako corpusaren menpe dago eta corpusaren domeinutik kanpo gure sistemaren kalitateak behera egin dezake.

Erdi-gainbegiratutako sistemek lehendabizi sailkatzaileak ikasteko metodo gainbegiratuak erabiltzen dituzte, eta gero sailkatzaile horiek hobetzen dira etiketa gabeko datumultzoekin. Ildo honetatik egindako lan arrakastatsuenetarikoak ezaugarri linguistikoetan oinarritutakoak (Collins eta Singer, 1999) edo aldi berean NE tresna ezberdinak entrenatuz lortzen direnak dira (Cucerzanand eta Yarowsky, 1999; Collins, 2002). Bootstrapping metodoak erabiltzea ere nahiko ohikoa da (Riloff eta Jones, 1999; Cucchiarelli eta Velardi, 2001), sistema batzuek beste sailkatzaile baten irteera erabiltzen dute, beste batzuek berriz eskuz etiketatutako hasierako entitate lista baten bidez garatzen dituzte NERC modeloak.

Proiektu honetarako badugu etiketatutako corpus zabal bat, beraz, corpus horretan oinarrituta gainbegiratutako NERC tresna bat garatzea oso aproposa da, nahiko epe laburrean sistema on bat lortzea posible izango da. 4.4. kapituluan aztertuko dugu gure NERC moduluaren berezitasunak.

2.2.4 Sentimenduaren analisia eta hizkuntza-baliabideak

Sentimenduaren analisia eta iritziaren azterketaren garrantzia gero eta handiagoa da zerbitzu, enpresa edo marka bati buruzko ospea eta kontsumitzaileen iritzia jakin ahal izateko. Sarean zehar dagoen informazio guztia eskuz analizatzea ezinezkoa da, beraz, azken hamarkadan iritziaren azterketa automatikoa ikertzeko premia sortu da. Ingelerarako lan eskerga egin da iritziaren azterketan. Batez ere bi metodologia ditugu: ikasketa automatiko gainbegiratua eta positibo, negatibo eta neutralki etiketaturiko lexikoia erabiltzen duten metodo ez-gainbegiratuak. Ikasketa gainbegiratuak corpus etiketatu bat behar du polaritatea adierazten duten ezaugarriak zeintzuk diren ikasi ahal izateko. Bi metodologien artean zenbait konparaketa egin dira, esate baterako, filmei buruzko iritziak jasotzeko orduan gainbegiratutako metodoekin %85-ko zehaztasuna lortu zen ez-gainbegiratutakoekin %77-ra iritxi ziren bitartean (Chaovalit eta Zhou, 2005). Sistema gainbegiratuen emaitza hobeak lortu izan ohi dira domeinu konkretuetan, baina sistema horiek beste domeinuetan pasatzen badira errendimenduak behera egiten du. Sistema ez-gainbegiratuen emaitzak sendoagoak izaten dira domeinu ezberdinen artean, gainera, sistema gainbegiratuen berezko desabantaila kalitate handiko corpusaren beharra da.

Iritziaren azterketarako teknika konputazionalez gain badira gai asko kontuan hartu beharrekoak kalitate eta berrerabilpenaren inguruan. Iritziaren azterketa testu-maila ezberdinetan aplikatu daiteke: hitzak, esaldiak, paragrafoak edo dokumentuak. Hitzek polaritate

ezberdinak dituzte esanahiaren eta domeinuaren arabera, beraz esanahiaren desanbiguazioa edo dokumentuaren gaiaren identifikazioa behar du. Gainera, dokumentuko zati bakoitzak polaritate ezberdina adierazi lezake, izan ere, iritziak gai bati buruz edo solaskide ezberdinei lotuta egon daitezke. Proiektu honetan oinarrizko polaritate-mailak aztertuko dira (polaritate ona, txarra edo neutrala) eta metodo erdi-gainbegiratu baten bidez sortuko dugu sentimendurako hizkuntza-baliabidea.

2.2.5 Proiekturako zerbitzu eta baliabide lexiko baliagarriak

Proiektu hau turismoaren domeinura orientatuta dagoenez interesgarria da sarean zehar eskura ditugun turismorako web-zerbitzuak ezagutzea. Web-zerbitzu hauen bidez erabiltzaileek nolako informazioa ematen duten ezagutu dezakegu eta aplikazioak testu hauen aurrean nola jokatzen duen jakiteko aukera ematen digute. Honako hauek dira erabili ditzakegun web-zerbitzu batzuk:

- **Foursquare**[15]. Bertan nahi diren kokapenak markatzen dira eta haiei buruzko iritziak idazten dira. Zerbitzu hau gai da erabiltzaileen kokapenaren arabera leku interesgarriak gomendatzeko.
- **Google Places**[16]. Fourquare-en antzekoa da baina bertako iritziak google-en beste zerbitzuekin integratuta daude.
- **Booking**[17]. Hotelen erreserbak egiteko zerbitzua da, bertan hotelen iritziak eta ranking-ak ditugu.

Erabiltzaileen iritziak sare sozialen bidez ere lortu daitezke, **Facebook**[18]-en badira turismo zerbitzuei buruz idazten duen jendea.

Arestian aipatu dugu proiektu honetarako hizkuntza-baliabide bat sortu beharko dugula sentimenduak etiketatu ahal izateko, ataza honetarako oso lagungarria izango da sareak eskaintzen dizkigun baliabide lexikoak ezagutzea. Proiektu honetarako hainbat baliabide lexiko ditugu kontuan hartu beharrekoak:

- **WOLF**[19] (Sagot et al., 2008). Baliabide lexikoen artean baliteke famatuena WordNet (Miller, 1995) izatea. WordNet-en bidez izen arruntak, aditzak, adjektiboak eta adberbioak multzokatzen dira esanahi lexikoaren arabera, eta aldi berean multzo hauek elkarren artean lotzen dira erlazio semantikoen bidez. Horixe dugu WOLF, frantses hizkuntzarako sortutako WordNet-a.
- **SentiWordNet**[20] (Esuli eta Sebastiani, 2006). SentiWordNet-ek WordNet-en ideiari

[15] `https://es.foursquare.com`
[16] `https://plus.google.com/local`
[17] `http://www.booking.com`
[18] `https://www.facebook.com/`
[19] `http://alpage.inria.fr/~sagot/wolf-en.html`
[20] `http://sentiwordnet.isti.cnr.it/`

helduz, sentimendu positibo, neutral eta negatiboa duten esanahiak lotzen dituen sarea da. SentiWordNet-en abantailarik garrantzitsuenetariko bat WordNet-en synsetak erabiltzen dituela da, hortaz aipatutako WOLF sarearekin lotzea posible da.

- **MCR**[21] (Gonzalez-Agirre et al., 2012) eta **EuroWordNet**[22] (Vossen et al., 1997). Bi hauek WordNet-en synsetak lotzen dituzte Europako hainbat hizkuntzetarako (EuroWordNet-en kasuan nederlandera, italiera, gaztelania, alemana, txekiera eta estoniera eta MCR-an gaztelania, euskara, katalan eta galiziera), eta hizkuntzen arteko lotura ontologien bidez aberasten da. Baliabide hauen erabilera interesgarria gerta liteke proiektuaren etorkizuneko zabalkuntzari begira.

[21] `http://adimen.si.ehu.es/web/MCR/`
[22] `http://www.illc.uva.nl/EuroWordNet/`

2.3 Erlazionatutako proiektu europarrak

1. taulan ikus ditzakegu proiektu honekin erlazionatutako proiektu Europarrak.

Proiektuaren izena	Proiektuaren azalpena	Ezberdintasunak gure proiektuarekin
ARTIFACTO - Analyzing and recognizing time, factuality, and opinion in text	Proiektu honen helburua LNP teknikak hizkuntza batetik bestera egokitzea da. Horretarako denbora eta sentimendua prozesatzen da	Gure proiektua etorkizunean hizkuntza ezberdinetara hedatzeko asmoz garatu da edozein hizkuntzarekiko independenteak diren metodoekin, izan ere, OpeNER proiektuak sei hizkuntza ezberdin integratzen ditu
MONNET - Multilingual Ontologies for Networked Knowledge	Monnet-en bidez semantikan oinarritutako tresnak eskainiko dira informazioa atzitzeko hizkuntzarekiko modu independentean. Gainera, ontologien inguruan ikerketak egin dira modelo semantikoak normalizatzeko informazioaren atzipen, integrazio eta erauzketari begira	Gure proiketuaren helburuetako bat hizkuntza baliabideak integratzeko metodoak estandarizatzea da
CALBC - Collaborative annotation of a large biomedical corpus	Proiektu honen helburua etiketak irudikatzeko eta konparatzeko formatu estandar bat definitzea da, gaur egun izen-entiteen detekzio sistema ezberdinak konparatzea ez baita lan erraza	Gure proiektuan, LNP modulu guztiek irteera dokumentuen ezaugarriak hizkuntzarekiko independea den XML formatu berezi baten bidez etiketuko ditu
ACCURAT - Analysis and evaluation of comparable corpora for under resourced areas of machine translation	ACCURAT-en helburua hizkuntza-baliabideen tamaina handitzeko metodoak ikertzea da corpus konparagarrien bidez, batez ere Itzulpen Automatikorako sistemak hobetzea lortzeko	Gure proiektuaren bidez, hizkuntza-baliabide eleanitzak sortu ahal izateko tresnak garatuko dira. Nahiz eta tresnak izen-entitate eta sentimenduaren detekziorako bereziki prestatuta egon inplementatutako metodoak beste domeinu eta erabilpen kasuetara moldatzea posible izango da

Taula 1: Erlazionatutako proiektu Europarrak

3 Diseinua

Diseinuari begira, tresna honek bete beharreko helburuak honako hauek ditugu:

- Instalatzeko erraza. Ahal den neurrian instalazioa erraztea komeni da. Edozein erabiltzailek nahiko lukeena komando bakar batekin sistema osoa instalatzea da, baina hori lortzea oso zaila izanik modulu guztien instalazioa nolabait estandarizatu behar da.
- Moldatzeko eta eraldatzeko erraza. Modulu guztiak era independentean ibiltzeko gai izan behar dira, baina ez hori bakarrik, etorkizunari begira moduluen hedapenak ahalik eta buruhauste gutxien eman behar ditu.
- Arkitektura irekia. Proiektu honetatik kanpo modulu berriak garatzea eta arkitekturan gehitzea posible izan behar da.

Aipatutako helburuak bete ahal izateko diseinu irizpide amankomun batzuk jarraitu behar dira. Diseinu orokorrerako kate itxura duen arkitektura bat aukeratu da, hau da, modulu bakoitzaren sarrera aurrekoaren irteera izango da. Diseinu erabaki honen bidez etorkizunean funtzionalitate berri bat inplementatu eta integratzea nahiko erraza da, modulu berri batek sarrera eta irteera formatuak aintzat hartzen baditu arazorik gabe integratuko genuke exekuzio-katean. 2. irudian ikus daiteke nola antolatzen diren modulu guztiak kate-arkitekturan.

Irudia 2: Kate-arkitektura orokorra

Jakina, modulu bakoitzaren integrazioa ez da modulua nolanahi sisteman jartzera murriztten, eta horretaz gainera moduluetatik zehar pasatzen diren dokumentuen formatua ere kontu handiz pentsatu behar da. Bi erronka horiek nola saihestu diren hurrengo azpiataletan azaltzen da.

3.1 Moduluen integrazioa

Aurreko ataleko erronka gainditzeko asmoz modulu guztiek arkitektura berbera izan beharko dute, eta aldi berean arkitekturak ahalik eta teknologia mota gehien integratu ahal izateko malgutasuna eskaini behar du. Horretarako, aplikazioa honako osagaiez hornituta egongo da:

HAP masterra

- **Modulua**. Aplikazioaren kate-arkitektura modulu multzo batez osatzen da. Modulu bakoitza Glue osagai bat eta Kernel osagai integratzen du bere baitan.
- **Kernela**. Moduluaren barne hizkuntza-teknologia inplementatzen duen osagaia da. Adibidez, tokenizaziorako moduluan testua tokenizatzen duen osagaia izango da kernela.
- **Glue-a**. Kernelaren inguruan kanporako konektibitatea ahalbidetzen duena. Glue-ak TCP, WS, CLI eta Router osagaiak ditu.
- **TCP**. TCP zerbitzaria duen osagaia, Glue-aren parte da. TCP zerbitzariaren bidez kernelarekin konexioa egiten da TCP portu baten bidez mezuak bidaliz.
- **WS**. WS (Web Service), Glue-aren parte da. Web zerbitzuari esker kernela atzitu daiteke web-eskaeren bidez.
- **CLI-a**. CLI (Command Line Interface), Glue-aren parte da. Kernelarekin komando lerrotik komunikatzeko CLI-a erabiliko dugu.
- **Routerra**. Routerra Glue-aren parte da. Routerrak beste Glue-ekin konexioa egiten du eskaerak luzatzeko. Adibidez, fitxategi bat tokenizatu eta etiketatu nahi bada tokenizatzailearen Routerrak testua etiketatzeko eskaera luzatuko dio etiketatzaileari.

Osagai guztiak nola antolatzen diren ulertu ahal izateko ikus 3. irudia.

Arkitektura honek aplikazioak eskatzen duen malgutasuna eskaini ahal izateko programazio lengoaia ezberdinen bidez programatutako kernelak integratzeko gauza izan behar gara. Hori lortu ahal izateko Glue-a Ruby[23] lengoaiaz programatuko da, eta Kerneletarako Java, Perl eta Python erabiliko ditugu. Ruby-ren bidez modu garden batean edozein kernel exekutatzeko gai izango gara.

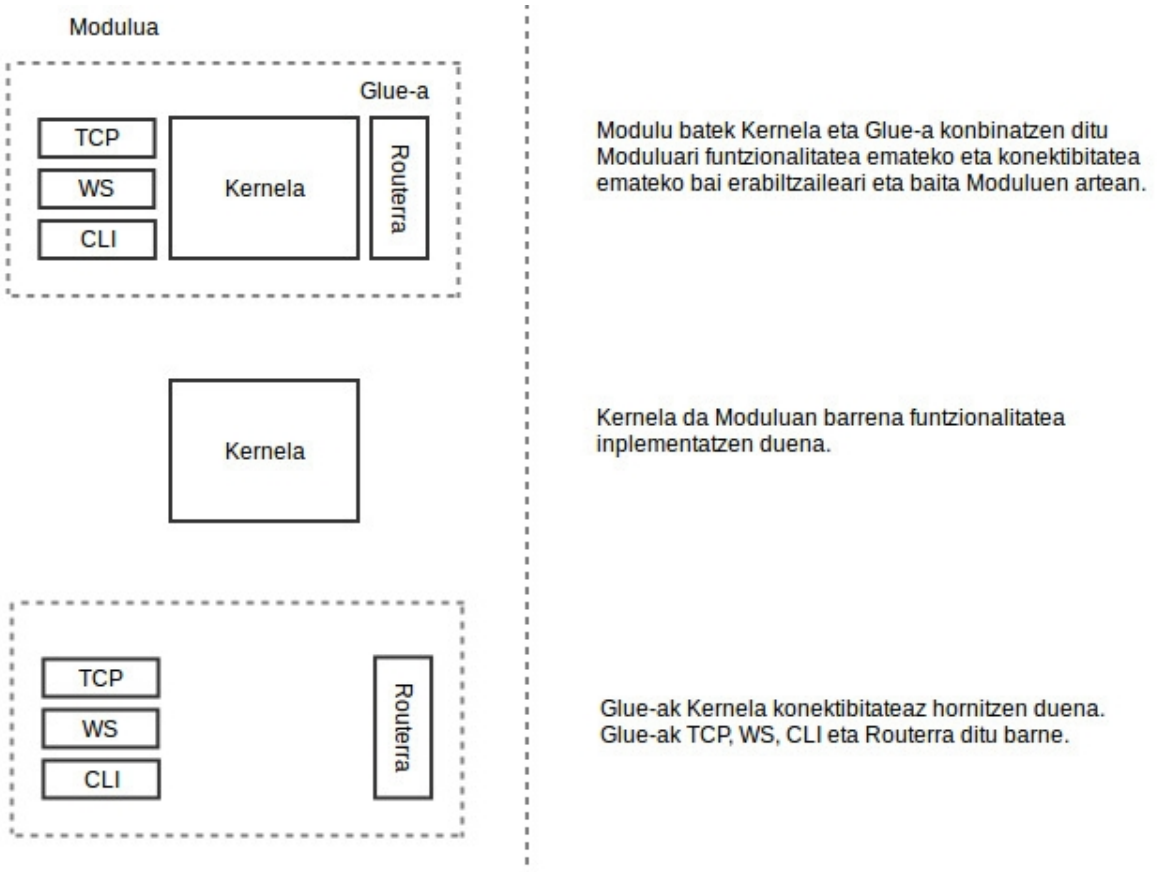
3.2 Moduluen arteko komunikazioa (KAF formatua)

Modulu bakoitza kateatu ahal izateko hauen artean informazioaren elkartrukatzea modu estandar batean egin behar da. Horrela, aplikazioaren hedapenari begira informazioren zabaltzea arautuz gero arau horiek betetzen dituen edozein modulu gehitu genezake aplikazioaren exekuzio-katean (ikus 3. atala).

Modulu guztiak zuzenean kateatu ahal izateko modulu bakoitzaren sarrera aurrekoaren irteera izango da, hau da, modulu bakoitzaren hizkuntza-prozesaketaren ondorioz sarrerako dokumentua osatuko da eta hurrengo modulura joango da zuzenean. Horretarako, dokumentu formatu estandar bat behar dugu honako baldintza hauek bete behar dituena:

- Mota askotako hizkuntza-informazioa gordetzeko gorde beharko da dokumentuan.

[23] `https://www.ruby-lang.org/es/`



Irudia 3: Aplikazioaren osagaiak

- Ahalik eta programazio lengoaia gehienak formatu hau prozesatzeko gai izan behar dira.
- Moduluek ez ezik, gizakiontzako ere irakurgarria izan behar du.
- Ez da oso formatu pisutsua izan behar.

Baldintza guzti horiek betetzen dituen formatua KAF dugu (Agirre et al., 2009). KAF formatua XML meta-lengoaian oinarritzen denez oso formatu estandarra dugu, izan ere, programazio lengoaia guztiak XML prozesatzeko gai dira. KAF formatua testu bateko hizkuntza-informazio antolatzeko bereziki prestatuta dago. Proiektu honetarako jatorrizko KAF formatuari berrikuntza txiki batzuk egin dizkiogu gure beharretara egokitzeko.

KAF formatuaren barne hizkuntza-informazio mota bakoitzeko geruza bat aurki dezakegu. Ideia nagusia modulu bakoitzak KAF dokumentu bat osatzea da bertan geruzak gehituz edo osatuz. Salbuespena lehenengo modulua dugu, honek sarreran testu fitxategi soil du eta irteeran lehengo KAF dokumentu bat utziko du oinarrizko informazioarekin. Hona hemen geruza bakoitzaren azalpen labur bat:

HAP masterra

- **kafHeader**. Dokumentuaren goiburua da. Bertan izango dugu jatorrizko testu dokumentuaren meta-informazioa (fitxategi izena, lengoaia, data, ...) eta baita erabilitako hizkuntza-prozesatzaileak. Hizkuntza-detektatzaileak beteko du.
- **raw**. Jatorrizko fitxategiko testua gordetzen duen geruza. Hau ere hizkuntza-detektatzailearen lana da.
- **text**. Bertan dugu testua tokenizatuta. Token bakoitzak bere identifikatzailea eta esaldia zenbakia ditu besteak beste. Esaldi banatzaile eta tokenizatzaileak beteko du geruza hau.
- **terms**. Token guztiak terminoetan batzen dira bakoitzaren kategoria gramatikalarekin batera. Kategoria etiketatzailea jardungo da honetan.
- **entities**. Detektatutako entitate-izen bakoitza hemen biltzen da bere motarekin batera. Entitate-izenen detektatzaile eta sailkatzailea lotzen dugu geruza honekin.
- **sentiment**. Termino bakoitzaren baitan sentimendu informazioa gordetzen etiketa honen bidez. Polaritate etiketatzaileak prozesatuko du.

KAF formatua nahiko konplexua da, horregatik xehetasun guztiak zatika azalduko dira 4. atalean. Nahi izanek gero honako web-helbidean aurkitu daiteke KAF formatuari buruzko informazio guztia: `https://github.com/opener-project/kaf/wiki/KAF-structure-overview`.

4 Moduluak

Atal honetan zehar proiekturako egindako moduluetan sakonduko dugu. 3.1. atalean zehazten den bezala moduluak baino kernelak azalduko ditugu, bertan prozesatzen baita testua. Kernel bakoitza zertan datzan eta nola garatu den ikusiko dugu, eta ebaluazioaren emaitzak ere aztertuko ditugu. Kernelak ebaluatzeko honako estatistikoak erabiliko ditugu:

- **Asmatzen-tasa** (ACC). Populazioaren eta egiazko emaitzen (bai positibo eta negatiboak) arteko proportzioa adierazten du:

$$ACC = \frac{TP + TN}{TP + FP + TN + FN}$$ [24]

- **Precision** (P). Sistemak positibotzat hartu dituen eta benetan asmatu dituen laginen arteko proportzioa da:

$$P = \frac{TP}{TP + FP}$$ [24]

- **Recall** (R). Populazioko lagin positiboaren eta egiazko positiboen arteko proportzioa.

$$R = \frac{TP}{TP + FN}$$ [24]

- **F1 score** ($F1$). Precision eta Recall estatistikoen arteko media harmonikoa da.

$$F1 = \frac{2TP}{2TP + FP + FN}$$ [24]

4.1 Hizkuntza-Detektatzailea

Bere izenak dioen bezala, hizkuntza-detektatzailearen eginkizuna sarrera testua zein hizkuntzan idatzita dagoen identifikatzea da. Exekuzio katearen lehenengo modulua izanik, hurrengo moduluak gai izango dira prozesamendua dagokion hizkuntzara egokitzeko. Hizkuntza-detektatzailea sarrera gisa testu hutsa duen modulu bakarra da, oinarrizko KAF dokumentua (ikus 3.2. atala) sortuko du eta prozesamendu-katean zehar dokumentu hori betetzen joango da. Oinarrizko KAF dokumentu honek goiburuan detektaturiko hizkuntza izango du raw geruzan jatorrizko testuarekin batera.

Hona hemen adibide bat:

```
$ echo "J'aime manger au restaurant Luxembourg." | language-identifier
```

Komandoaren emaitza:

[24] TP = egiazko positiboak; TN = egiazko negatiboak; FP = positibo faltsuak; FN = negatibo faltsuak.

```
<KAF xml:lang="fr" version="2.1">
  <raw>J'aime manger au restaurant Luxembourg.</raw>
</KAF>
```

Hizkuntza-detektatzaileak bete behar duen ezaugarri nagusia zehaztasuna da, bestela hurrengo modulu guztiek porrot egiteko arriskua dugu. Ordenagailuak oso onak dira hizkuntza detektatzen eta %95etik gorako zehaztasunak lortzen dira (Torres-Carrasquillo et al., 2002). Beste ezaugarri garrantzitsu bat azkartasuna da, litekeena da aplikazioa etengabe dokumentuak prozesatzen ibiltzea, eta ez bada prozesamendu ahalmen handi bat lortzen baliteke erabiltzaileek aplikazioa bertan behera uztea.

Aipaturiko ezaugarriak betetzeko asmoz, bi hizkuntza-detektatzaile probatu dira, bata perlez idatzitako *Text::Language::Guess*[25], eta bestea javaz programatutako *Cybozu Language-Detection* (Shuyo, 2010).

4.1.1 Text::Language::Guess

Hizkuntza-detektatzaile hau oso sinplea da. *Text::ExtractWords* liburutegiaren bidez hitzak erauzten ditu eta *Lingua::StopWords* liburutegiko stopword listarekin konparatzen ditu, erauzitako hitza hizkuntza bateko stopword bat bada hizkuntza horri puntu bat ematen zaio. Azkenean puntu gehien dituen hizkuntza pantailaratuko da. Hamar hizkuntza detektatzeko gauza da: ingelera, frantsesa, gaztelania, portugesa, italiera, alemana, nederlandera, suediera, norvegiera eta daniera.

Perl lengoaiaren berezko prozesamendu azkartasuna eta azaldutako algoritmoaren konbinatuz oso emaitza azkarrak lortzen dira. 4.1.3. atalean ikusiko ditugu emaitzak.

4.1.2 Cybozu Language-Detection

Wikipedia corpusaren bidez eraikitako sare bayestarren bidez identifikatzen dira hizkuntzak. Ausaz sarrerako testutik hainbat n-grama (1etik 5era) aukeratzen dira eta sare bayestarraren bidez hizkuntza bakoitzerako probabilitateak kalkulatzen dira. Prozesu hau hainbat iteraziotan errepikatzen da, eta azkenean, probabilitate handieneko hizkuntza pantailaratzen da. Esan beharra dago iterazio kopurua zazpira mugatuta dagoela jatorrizko kodean, honek zenbait errore eragiten ditu, izan ere, testuak oso laburrak badira (2-4 hitz) gerta liteke beti hizkuntza berbera ez identifikatzea. Arazo hori konpontzeko iterazio kopurua 200era igo da.

Hizkuntza-detektatzaile honek 53 hizkuntza inguru identifikatzeko gauza da, gainera oso erraza hizkuntza berriak gehitzea. Hizkuntza berriak gehitzeko n-grama lista bat duen fitxategi bat gehitzea da egin beharreko lan bakarra.

[25] `http://search.cpan.org/dist/Text-Language-Guess/lib/Text/Language/Guess.pm`

4.1.3 Ebaluzioa

Bi hizkuntza-detektatzaileak ebaluatuko ditugu proiektuko helburuak betetzeko egokiena zein den jakiteko. Lehenago aipatu dugu hizkuntza-detektatzaile batek azkarra eta zehatza izan behar duela. 2. taulan testu bat prozesatzeko denbora neurtzen da milisegundotan bataz-beste. 3. taulan berriz, sei hizkuntzatan idatzitako 10.000 esaldi analizatu dira hizkuntza bakoitzeko eta hizkuntza-detektatzaile bakoitzak izandako asmatze-tasa aurkezten da ehunekotan. Analizatutako corpusa Leipzig Corpus Bildumatik[26] lortu dugu 10-20 hitzetako esaldiak biltzen dituena.

Abiadura milisegundotan	
Text::Language::Guess	Cybozu Language-Detection
76	1035

Taula 2: Hizkuntza-detektatzaileen abiadura

Hizkuntza	Asmatze-tasa ehunekotan	
	Text::Language::Guess	Cybozu Language-Detection
Ingelera	98,44	99,51
Gaztelania	96,83	99,03
Frantsesa	83,88	99,08
Italiera	96,40	99,38
Alemana	98,97	99,73
Nederlandera	95,64	99,11
Guztira	95,03	99,31

Taula 3: Hizkuntza-detektatzaileen asmatze-tasa

Abiadurari dagokionez *Text::Language::Guess* ia 14 aldiz azkarragoa da bestea baino, ez da harritzekoa, perl lengoaia java baino azkarragoa baita, gainera askoz ere algoritmo sinpleagoa inplementatzen du. Hala ere, nahiz eta *Text::Language::Guess* detektatzailearen zehaztasuna orokorrean oso ona izan, frantsesez idatzitako testuak identifikatzeko nahiko arazoak ditu, *Cybozu Language-Detection*-ren zehaztasuna berriz paregabea da. Emaitza hauek ikusita *Cybozu Language-Detection* aukeratuko dugu, gainerako moduluen arrakastarako frantses hizkuntza ondo identifikatzea kritikoa baita.

4.2 Esaldi-banatzailea eta tokenizatzailea

Modulu honen baitan esaldi-banatzailea eta tokenizatzailea elkartuko ditugu. Tokenizatzailearen lana hurrengo prozesatzaileentzat jatorrizko testuko unitate lexiko minimoak

[26] http://corpora.uni-leipzig.de/download.html

identifikatzea izango da. Adibide gisa, `c'est` hitz-forma `c'` eta `est` tokenetan zati dezakegu. Tokenizazioa ez bada modu egokian egiten hurrengo prozesatzaileen errendimenduak behera egingo du, hauen entrenamendua tokenizaturiko testuarekin egiten baita.

Zenbait hizkuntza-prozesatzailek (OpenNLP hauen artean) sarrerako testua esalditan banatuta badute emaitza hobeak lortzen dituzte, esaldi-banatzaileari esker testua osorik edo esaldika prozesatu dezakegu.

Modulu honek sarrerako KAF dokumentuari *text* geruza erantsiko dio tokenizazio informazioarekin. Hizkuntza-detektatzailearen adibidearekin jarraituz, sarreran honako *lang-detected.kaf* dokumentua bagenu:

```
<KAF xml:lang="fr" version="2.1">
  <raw>J'aime manger au restaurant Luxembourg.</raw>
</KAF>
```

Honela tokenizatuko genuke:

```
$ cat lang-detected.kaf | tokenizer
```

Hona hemen emaitza:

```
<KAF xml:lang="fr" version="v1.opener">
  <kafHeader>
    <fileDesc />
    <linguisticProcessors layer="text">
      <lp name="opener-sentence-splitter-fr" version="0.0.1"
                      timestamp="2014-05-18T15:53:21Z"/>
      <lp name="opener-tokenizer-fr" version="1.0.1"
                      timestamp="2014-05-18T15:53:21Z"/>
    </linguisticProcessors>
  </kafHeader>
  <text>
    <wf wid="w1" sent="1" para="1" offset="0" length="2"><J'></wf>
    <wf wid="w2" sent="1" para="1" offset="2" length="4"><aime></wf>
    <wf wid="w3" sent="1" para="1" offset="7" length="6"><manger></wf>
    <wf wid="w4" sent="1" para="1" offset="14" length="2"><au></wf>
    <wf wid="w5" sent="1" para="1" offset="17" length="10"><restaurant></wf>
    <wf wid="w6" sent="1" para="1" offset="28" length="10"><Luxembourg></wf>
    <wf wid="w7" sent="1" para="1" offset="38" length="1"><.></wf>
  </text>
</KAF>
```

Diseinuaren kate-arkitekturarekin jarraituz, nahi izanez gero hizkuntza-detektatzailea eta tokenizatzailea oso erraz kateatu genitzake emaitza berbera lortzeko, begira nola:

HAP masterra

```
$ echo "J'aime manger au restaurant Luxembourg." | language-identifier | tokenizer
```

Aurreko adibidean ikus daitekeen bezala token bakoitzak zenbait atributu ditu. Hona hemen atributu bakoitzaren azalpena:

- **wf**: tokenaren identifikazioa.
- **sent**: esaldi zenbakia (1-etik hasten da).
- **para**: parrafo zenbakia (hau ere 1-etik hasita).
- **offset**: jatorrizko testuan tokenaren posizioa.
- **length**: tokenaren tamaina.

4.2.1 Moses tokenizatzailea

Modulu hau eraikitzeko orduan bi aukera hartu ditugu kontutan: gainbegiratutako esaldi-banatzaile eta tokenizatzaile bat ikastea, edo Moses SMT (Koehn et al., 2007) tresnaren baitan integraturiko erregelan oinarritutako scriptak erabiltzea. Lehenengo irtenbiderako badugu ESTER (Galliano et al., 2005) entrenamendurako corpusa, gainera OpenNLP-ri esker oso erraza da ikasketa prozesua, baina proba batzuk egin eta gero konturatu gara komatxoek arazo asko ematen dituztela. Nahiz eta hizkuntza bakoitzean erabili beharreko komatxoak araututa egon, errealitatean jendeak nahi dituen komatxoak erabiltze ditu, horregatik oso garrantzitsua da komatxo mota guztiak biltzen dituen entrenamendu corpus bat izatea, tamalez hori oso zaila da.

Moses-en baitan aurkitzen den esaldi-banatzailea eta tokenizatzailea perl-ez idatzita daude eta 20 hizkuntzetarako probatuta da (katalana, txekiera, alemana, greziera, ingelera, gaztelania, frantsesa, hungariera, islandiera, italiera, letoniera, nederlandera, poloniera, portugesa, errumaniera, errusiera, esloveniera, eslovakiera, eskandinaviera eta tamilera). Beste abantaila bat kodea moldatzeko erraztasuna da, edozein hizkuntzarako erregelak idaztea posible da eta badira *nombreakin_prefix* izeneko fitxategiak zatitzea nahi ez ditugun kasu bereziak sartzeko. Proiektu honetarako Moses esaldi-banatzailea eta tokenizatzailea script bakar batean bildu ditugu eta frantseserako erregela pare bat idatzi ditugu.

4.2.2 Ebaluzioa

Tokenizatzailea ebaluatzeko ESTER corpusa erabiliko dugu. ESTER corpusa hiru zatitan banatuta dago entrenamendu, garapen eta ebaluazio atazak burutu ahal izateko. 4.4.2. atalean sakonki deskribatzen den bezala ESTER corpusa Frantziako irrati saioetako transkripzioak bilduz sortu zen, horregatik badira zenbait espresio ahozko hizkuntzan bakarrik erabiltzen direnak, adibidez, igorlea nahasi egiten bada *eeh, eeh...* bezalako tokenak

agertzen dira komunikazioarekin jarraitzeko, eta batzuetan igorlearen eta hartzailearen komunikazioa nahasten da. Horregatik baliteke ESTER corpusa ebaluaziorako egokiena ez izatea, baina frantseserako ez daude gaztelania edo ingelerarako bezainbeste baliabide.

4. taulan ESTER ebaluazio corpusaren estatistikak ikus ditzakegu. Tokenizatzaileak asmatu beharreko hitz-formak puntuazio ikurrak dituztenak dira, ez du zentzurik puntuazio ikurrik ez duten hitz-formak ebaluatzea hauek ez baitira tokenizatzen.

ESTER ebaluazio corpusa	
Esaldi kopurua	3.966
Hitz kopurua	124.510
Asmatu beharreko hitz-forma kopurua	20.717

Taula 4: ESTER ebaluazio corpusaren estatistikak

5. taulan modu egokian tokenizatutako hitz-formen portzentaia ikus dezakegu. Oso garrantzitsua da asmatze-tasa oso altua izatea hurrengo hizkuntza-prozesatzaileek ahalik eta zarata gutxien izan dezaten. Taulako emaitzak ikusita tokenizatzailea oso egokia da proiektu honetarako, asmatze-tasa %98,44koa da eta ESTER ebaluazio corpus osoaren tokenizazioak 4 segundo besterik ez du iraun.

Tokenizatzailearen estatistikak	
Prozesatze abiadura milisegundotan	4096
Asmatze-tasa ehunekotan	98,44

Taula 5: Tokenizatzailearen estatistikak

4.3 Etiketatzaile-morfologikoa

Etiketatzaile-morfologikoaren eginkizun nagusia sarrerako testuko kategoria gramatikalak identifikatzea izango da. Kategoria gramatikalak ezagutzea oso garrantzitsua hizkuntza-prozesatze ataza askotarako, esate baterako, chunking-a, korreferentziaren erresoluzioa, polaritatearen etiketatzea etab. Horregatik, proiektu honetan tokenizatzailearen ostean kokatuko dugu etiketatzaile-morfologikoa.

Proiektu honetako moduluek bezala sarreran KAF dokumentu bat izango dugu. KAF dokumentua prozesatu ondoren morfologikoki etiketatutako KAF dokumentu bat itzuliko da. Sarrerako KAF dokumentuak nahitaez `<text>` geruza izan beharko du, bertan azaltzen baita tokenizatutako jatorrizko testua. KAF dokumentuan analisi morfologikorako `<terms>` geruzan egiten da eta bertan termino bakoitzeko `<term>` elementu bat izango dugu. Hauexek dira `<term>` elementuaren atributuak:

- *tid*: identifikatzailea, *t* letraz hasten da.
- *type*: terminoaren mota, bi balio izan ditzake:

  - *open*: klase irekiko terminoa, izen-arruntak, adjektiboak, aditzak eta adberbiak dira.
  - *close*: klase itxiko terminoa, bestelako kategoria gramatikala dutenak dira.
- *lemma*: terminoaren lema.
- *pos*: kategoria gramatikala.
  - izen-arrunta (N).
  - izen-berezia (R).
  - adjektiboa (G).
  - aditza (V).
  - preposizioa edo posposizioa (P).
  - adberbioa (A).
  - juntagailua (C).
  - determinatzailea (D).
  - bestelakoa (O).
- *morphofeat (aukerazkoa)*: ezaugarri morfosintaktikoa adierazteko, azpikategoria gramatikala adibidez.
- *head*: terminoak hitz-konposatua bada, hitz nagusia zein den adierazten du.

`<term>` elementu bakoitzak honako azpi-elementuak izan ditzake:

- span: terminoak zein token erreferentziatzen duen erakusten digu. `<target>` azpi-elementu bat du *wid* atributuarekin tokenaren identifikatzailearekin. Terminoa Hitz Anitzeko Unitate Lexikala (HAUL) bada `<target>` elementu bat egongo da token bakoitzeko.
- *component*: terminoa hitz konposatu bat bada hitz konposatua osatzen duen elementu bakoitzeko `<component>` azpi-elementu bat beharko dugu. Adibidez, *porte-parole* terminoa (*bozeramailea* frantsesez) honela deskonposatzen da:

```
<term head="t176.3" lemma="porte-parole" pos="N" tid="t176"
            type="open">
  <span>
    <target id="w178"/>
  </span>
  <component id="t176.1" lemma="porte" pos="V"/>
  <component id="t176.2" lemma="-" pos="O"/>
  <component id="t176.3" lemma="parole" pos="N"/>
</term>
```

`<component>` azpi-elementuaren atributuak hauek dira:

  - *id*: identifikatzailea *t* letraz hasita.
  - *lemma*: terminoaren lema.
  - *pos*: kategoria gramatikala.
  - *case* (hautazkoa): hitz komposatua inflexiaren bidez sortu bada, inflexio-kasua.

- *sentiment*: sentimenduari dagokion informazioa kodetzeko. 4.5. atalean sakontzen da honetan.
- *externalReferences*: baliabide lexiko eta semantikoak erreferentziatzako erabiltzen da. Elementu hau polaritate-etiketatzaileak sortuko duenez 4.5. atalean azalduko dugu.

HAUL-ak KAF dokumentuetan irudikatzeko `<term>` elementuaren barne `<target>` azpi-elementu bat sartzen da hitz bakoitzeko, eta hitz-elkartuekin egiten den bezala, `<component>` azpi-elementuak erabiltzen dira. Adibidez, *Premier* eta *Ministre* terminoak HAUL analisia egin baino lehen honela ipiniko genituzke:

```
<term lemma="Premier" pos="N" tid="t5" type="open">
  <span>
    <target id="w5"/>
  </span>
</term>

<term lemma="Ministre" pos="N" tid="t6" type="open">
  <span>
    <target id="w6"/>
  </span>
</term>
```

HAUL-a prozesatu eta gero:

```
<term head="5.2" lemma="Premier_Ministre" pos="N" tid="t5" type="open">
  <span>
    <target id="w5"/>
    <target id="w6"/>
  </span>
  <component id="t5.1" lemma="Premier" pos="N"/>
  <component id="t5.2" lemma="Ministre" pos="N"/>
</term>
```

Azaldutako `<terms>` geruza hobeto argitzeko tokenizatutako KAF dokumentu bat nola prozesatzen den ikusiko dugu. Sarreran honako *tokenized.kaf* dokumentua bagenu:

```
<KAF xml:lang="fr" version="v1.opener">
```

HAP masterra

```
  <kafHeader>
    <fileDesc />
    <linguisticProcessors layer="text">
      <lp name="opener-sentence-splitter-fr" version="0.0.1"
                      timestamp="2014-05-18T15:53:21Z"/>
      <lp name="opener-tokenizer-fr" version="1.0.1"
                      timestamp="2014-05-18T15:53:21Z"/>
    </linguisticProcessors>
  </kafHeader>
  <text>
    <wf wid="w1" sent="1" para="1" offset="0" length="2"><J'></wf>
    <wf wid="w2" sent="1" para="1" offset="2" length="4"><aime></wf>
    <wf wid="w3" sent="1" para="1" offset="7" length="6"><manger></wf>
    <wf wid="w4" sent="1" para="1" offset="14" length="2"><au></wf>
    <wf wid="w5" sent="1" para="1" offset="17" length="10"><restaurant></wf>
    <wf wid="w6" sent="1" para="1" offset="28" length="10"><Luxembourg></wf>
    <wf wid="w7" sent="1" para="1" offset="38" length="1"><.></wf>
  </text>
</KAF>
```

Honela analizatuko genuke:

```
$ cat tokenized.kaf | postagger
```

Hona hemen emaitza:

```
<KAF xml:lang="fr" version="v1.opener">
  <kafHeader>
        ⋮
    <linguisticProcessors layer="terms">
      <lp name="opennlp-pos-treetagger-fr"
                      timestamp="2014-05-18T15:54:21Z" version="1.0" />
      <lp name="opennlp-multiword-fr"
                      timestamp="2014-05-18T15:54:21Z" version="1.0" />
    </linguisticProcessors>
  </kafHeader>
  <text>
     ⋮
  </text>
  <terms>
    <!--J'-->
    <term tid="t1" type="close" lemma="J'" pos="Q" morphofeat="CL1ms">
      <span>
        <target id="w1" />
```

HAP masterra

```
      </span>
    </term>
    <!--aime-->
    <term tid="t2" type="open" lemma="aime" pos="V" morphofeat="VP3s">
      <span>
        <target id="w2" />
      </span>
    </term>
    <!--manger-->
    <term tid="t3" type="open" lemma="manger" pos="V" morphofeat="VW">
      <span>
        <target id="w3" />
      </span>
    </term>
    <!--au-->
    <term tid="t4" type="close" lemma="au" pos="P" morphofeat="P">
      <span>
        <target id="w4" />
      </span>
    </term>
    <!--restaurant-->
    <term tid="t5" type="open" lemma="restaurant" pos="V" morphofeat="VG">
      <span>
        <target id="w5" />
      </span>
    </term>
    <!--Luxembourg-->
    <term tid="t6" type="open" lemma="luxembourg" pos="N" morphofeat="NPms">
      <span>
        <target id="w6" />
      </span>
    </term>
    <!--.-->
    <term tid="t7" type="close" lemma="." pos="O" morphofeat=".">
      <span>
        <target id="w7" />
      </span>
    </term>
  </terms>
</KAF>
```

Hizkuntza-detektatzailea, tokenizatzailea eta etiketatzaile-morfologikoa kateatzea posible da emaitza berbera lortzeko:

HAP masterra

```
$ echo "J'aime manger au restaurant Luxembourg." \
       | language-identifier | tokenizer | postagger
```

4.3.1 OpenNLP bidez entrenaturiko Etiketatzaile-Morfologikoa

Etiketatzaile-morfologikorako gainbegiraturiko MaxEnt (Maximum Entropy) modelo bat entrenatzea erabaki dugu, kalitatezko entrenamendu corpus bat izanda emaitza onak lortzen baitira (Ratnaparkhi et al., 1996). Proiektuaren hedapenari begira MaxEnt modeloak erabiltzearen beste abantaila bat hizkuntza berrietarako etiketatzaile-morfologiak egitea oso erraza dela da, funtsean entrenamendu corpus bat besterik ez da behar. Modelo berria entrenatu eta gero jatorrizko etiketatzaile-morfologikoan hiruzpalau kode lerrorekin integrazioa eginda dago.

MaxEnt eredu probabilistikoaren oinarria inposatutako muga batzuen artean ahalik eta hipotesi gutxien egitea da. Muga horiek entrenamendurako corpusetik eratortzen dira ezaugarrien eta emaitzen artean nolabaiteko erlazioak ikasiz. Beheko propietatea betetzen duen eredu probabilistikoa entropia altuena duena da. Eredu probabilistikoa bakarra da, aukera maximoaren banaketa (maximum-likelihood distribution) betetzen du eta forma esponentziala du Pietra et al. (1997):

$$p(o|h) = \frac{1}{Z(h)} \prod_{j=1}^{k} \alpha_i^{f_j(h,o)}$$

non o emaitza den, h kontestua, eta $Z(h)$ normalizazio funtzioa. $f_j(h,o)$ funtzio bakoitza bitarra da. Adibidez, hitz batek zein kategoria (etiketa morfologikoa gure kasuan) duen estimatzeko, o egiazkoa edo gezurrezkoa da, non h-k inguruko testuari egiten dio erreferentzia:

$$f_j(h,o) = \begin{cases} 1 & \text{if } o = egiazkoa, \text{ aurrekoa} = the \\ 0 & \text{bestela} \end{cases}$$

α_i parametroak Generalized Iterative Scaling (GIS) (Darroch eta Ratcliff, 1972) prozeduraren bidez estimatzen dira. Prozesu iteratibo bat da non iterazio bakoitzean parametroen estimazioa hobetzen den.

MaxEnt modeloa entrenatzeko OpenNLP-k eskaintzen dizkigun liburutegiak (*opennlp.tools.postag.PosT*
paketea) erabiliko ditugu (Baldridge et al., 2002). Oso erraza da liburutegi honen bidez modeloak entrenatzea eta liburutegien integrazioak Javaz idatzita egonik ez dago arazo handirik integrazioari begira.

Entrenamendu corpusari begira OpenNLP-k formatu berezia eskatzen du. OpenNLP formatuan lerro bakoitzeko esaldi bat ipini behar da token/kategoria morfologiko pare bakoitza "_" karakterearen bidez konbinatzen delarik:

```
C'_D est_V un_D texte_N en_P français_N ._O
Comment_A tu_Q t'_Q appelles_V ?_O
```

HAP masterra

4.3.2 OpenNLP bidez entrenaturiko Hitz-elkartu eta HAUL detektatzilea

Hitz-elkartu eta HAUL-ak detektatu ahal izateko oso irtenbide erraza aukeratu dugu, etiketatzaile-morfologikoarekin egin dugun antzera OpenNLP-ko opennlp.tools.postag.PosTaggerME paketea erabili dugu sailkatzaile bat entrenatzeko. Kasu honetan, etiketa-morfologikoak beharrean hitz bat elkartua ala HAUL baten parte bada `C` etiketa izango du eta `S` etiketa bestela. Adibide honetan *prise d'otages* (bahitua frantsesez) hitz-elkartua entrenamendu corpus batean nola etiketatuko litzatekeen ikus dezakegu:

```
Une_S prise_C d'_C otage_C vise_S à_S retenir_S des_S personnes_S
contre_S leur_S volonté_S ._S
```

4.3.3 Lematizatzailea

Lematizatzailea egiteko lema-hiztegi bat sortu dugu entrenamendu corpusean ikusitako informazioarekin. Token berberak bere kategoria morfologikoaren arabera lema bat edo beste izan dezake, honen adibide bat frantsesezko *muse* tokena dugu: *muse* izen arrunta bada bere lema *muse* da (alosia, musa), baina bere kategoria aditza bada orduan lemak *muser* behar du (paseatu aditza). Anbiguetate lexiko hori saihesteko lematizatzaileak sarreran tokena eta etiketa morfologikoa izango ditu eta lematizatzaileak token/etiketa parea hiztegian ez badu aurkitzen tokena bera izango da irteerako lema.

Lema hiztegia hash taula baten bidez gordeko dugu, hash taularen giltza tokena izango da eta balioa lema/kategoria bikote posible guztiak izango dira 6. taulan erakusten den moduan.

Giltza	Balioa
muse	muse/N muser/V
embringuées	embringuer/G embringuer/V
frisé	friser/G frisé/G frisé/N friser/V
⋮	⋮

Taula 6: Lema hiztegia

4.3.4 French Treebank corpusa

Etiketatzaile-morfologikoaren sailkatzaile eta hitz-elkartu detektatzailearen entrenamendurako eta lema-hiztegia eraikitzeko French Treebank[27] corpusa aukeratu dugu (Abeillé et al., 2003), bertan analisi morfosintaktikoa eginda dago, beraz, gure beharretarako ez ezik analizatzaile sintaktiko bat ikasteko ere balioko liguke. French Treebank corpusak 21.565 esaldi eta 587.690 token ditu.

[27] http://www.llf.cnrs.fr/Gens/Abeille/French-Treebank-en.php

Formatuari dagokionez XML dokumentutan antolatzen da corpusa. Esaldi bakoitzak `<SENT>` etiketa darama eta tokenentzako `<w>` etiketa erabiltzen da. `<w>` etiketen barne hainbat atributu daude ezaugarri morfologikoak kodetzeko, guri `pos` (kategoria), `lemma` (lema) eta `compound` (hitz-elkartu edo HAUL-a) atributuak bakarrik interesatzen zaizkigu.

Hona hemen French Treebank corpuseko esaldi bat:

```
<SENT nb="8000">
  <w compound="yes" cat="ADV" ee="ADV" ei="ADV" lemma="tout au plus">
    <w catint="ADV">Tout</w>
    <w catint="P">au</w>
    <w catint="D"/>
    <w catint="ADV">plus</w>
  </w>
  <w lemma="un" cat="D" subcat="ind" ee="D-ind-fp" ei="Dfp" mph="fp">
     des
  </w>
  <w cat="A" ee="A-qual-fp" ei="Afp" lemma="petit" mph="fp" subcat="qual">
     petites
  </w>
  <w cat="N" ee="N-C-fp" ei="NCfp" lemma="chose" mph="fp" subcat="C">
     choses
  </w>
  <w cat="P" ee="P" ei="P" lemma="à">à</w>
  <w cat="V" ee="V--W" ei="VW" lemma="changer" mph="W" subcat="">
     changer
  </w>
  <w cat="P" ee="P" ei="P" lemma="sur">sur</w>
  <w cat="D" ee="D-def-fs" ei="Dfs" lemma="le" mph="fs" subcat="def">l'</w>
  <w cat="N" ee="N-C-fs" ei="NCfs" lemma="intégration" mph="fs" subcat="C">
     intégration
  </w>
  <w cat="PONCT" ee="PONCT-S" ei="PONCTS" lemma="." subcat="S">.</w>
</SENT>
```

Jakina, corpus honekin OpenNLP bidez sailkatzailea ikasteko formatu aldaketa bat egin behar dugu.

Corpusa egokitu ondoren hiru zatitan banatu dugu entrenamendu, garapen eta ebaluazio atazak egin ahal izateko, 7. taulan erakusten da zati bakoitzaren tamaina.

HAP masterra

	Train	Develop	Test	Guztira
Esaldi kopurua	18.235	1.642	1.688	21.565
Token kopurua	503.133	42.622	41.935	587.690
Hitz-elkartu kopurua	14.154	1.853	1.287	17.294

Taula 7: French Treebank corpusaren zatien tamainak

4.3.5 French Multi-word Nouns Laporte corpusa

HAUL-ak identifikatu ahal izateko *French corpus annotated for Multiword Nouns* corpusa erabili dugu (Laporte et al., 2008). Corpus honek Frantziako Legebiltzarreko zenbait transkripzio eta Jules Verne-ren *Le Tour du monde en quatre-vingts jours* nobelatik erauzitako testua biltzen ditu. Formatuari dagokionez, corpus hau testu soila da HAUL-ak identifikatzeko XML etiketa batzuekin. XML etiketek honako patroia jarraitzen dute:

- *N* elementua.
- *fs* atributua.

fs atributuaren balioa azpi-kategoria identifikatzailearekin hasten da, ":" karakterearen jarraitzen du, eta azkenik numeroa eta generoa darama. Adibidez: `<N fs='AN:ms'>premier ministre</N>`.

Honako hauek dira corpusaren estatistikak:

HAUL corpusa	
Esaldi kopurua	3.560
Token kopurua	172.202
HAUL kopurua	1.863

Taula 8: French corpus annotated for Multiword Nouns corpusaren estatistikak

4.3.6 Ebaluazioa

Entrenaturiko kategoria-morfologiko sailkatzailearen ebaluazioa French Treebank corpusaren garapen eta ebaluaziorako corpusen gainean egin da. 9 eta 10. tauletan guztira eta kategoria bakoitzeko lortutako estatistikak aurkezten dira, 11. taulan berriz, etiketatzaile morfologikoak KAF dokumentu bat prozesatzen ematen duen denbora ikus dezakegu.

Oro har, etiketatzaile-morfologikoaren estatistikak oso onak dira, ebaluaziorako corpuseko etiketen %95-a asmatzen baita, hala ere, interjekzio eta atzerriko hitzak identifikatzeko zailtasunak ditu. Ez da harritzekoa, izan ere, corpus osoan oso token gutxi daude kategoria horiekin. Bestalde, prozesaketa abiadurari erreparatuz ia bi segundo irauten ditu, seguru aski denbora gehiena OpenNLP sailkatzailea memorian kargatzen irauten du.

HAP masterra

	Develop	Test
Asmatze-tasa	95,42	94,9

Taula 9: Kategoria-morfologiko sailkatzailearen asmatze-tasa ehunekotan

Ebaluaziorako Corpusaren Estatistikak					
Kategoria morf.	Erroreak	Zenbat	Precision	Recall	F1 score
Adberbioak	494	2.204	92,4	77,6	84,3
Adjektiboak	362	2.634	88,3	86,3	87,3
Aditzak	222	4.901	93,5	95,5	94,5
Izenak	258	10.190	94,6	97,5	96,0
Preposizioak	237	6.344	96,3	96,3	96,3
Perts. izenordainak	102	1.271	99,0	92,0	95,4
Beste izenordainak	182	792	96,4	77,0	85,6
Juntagailuak	114	1.466	93,4	92,2	92,8
Determinatzaileak	134	6.091	94,3	97,8	96,0
Puntuazio ikurrak	11	5.976	99,0	99,8	99,4
Interjekzioak	11	12	1,0	8,3	15,4
Atzerriko hitzak	8	8	0,0	0,0	0,0
Aurrizkiak	1	20	1,0	95,0	97,4

Taula 10: Kategoria-morfologiko sailkatzailearen estatistikak kategoriaka ebaluaziorako corpusean

KAF dokumentu bat prozesatzeko denbora
2,442 segundo

Taula 11: KAF dokumentu bat prozesatzeko denbora

4.4 Entitate-izenen detektatzaile eta sailkatzailea

Entitate-izenak detektatzea eta sailkatzea entitate zehatzak erreferentziatzen dituzten unitate-lexikoak detektatzea eta zein motatako entitateak diren (pertsonak, lekuak, erakundeak eta abar) determinatzea da. Informazioaren erauzketarako oso baliagarria da entitateen analisia egitea, honi esker testuak filtratu ditzakegu eta nolabaiteko ezagutza erauzi dezakegu. Ikus honako adibidea:

> *Ancien chef charismatique d'* <START:organization> *Al-Qaida* <END> *au* <START:location> *Maghreb* <END> *islamique –* <START:organization> *AQMI* <END> *, l' Algérien* <START:person> *Mokhtar Bel-Mokhtar* <END> *a une trajectoire remarquable dans la nébuleuse islamiste .*

Goiko adibidean entitate-izenek ematen diguten informazioarekin bakarrik badakigu zein entitatek hartzen duten parte eta erakunde eta lekuekin lotu ditzakegu.

HAP masterra

Gure entitate-izenen modulua etiketatzaile-morfologikoaren ostean kokatzen da, sarrerako KAF dokumentuaren terminoak prozesatzen ditu (`<terms>` geruza) eta irteerako KAF dokumentuan irakurritako terminoak izen-entitateekin lotzen dira. Izen-entitateei buruzko informazio guztia `<entities>` geruzan antolatzen da izen-entitate bakoitzak `<entity>` elementu bat duelarik.

Hona hemen `<entity>` etiketaren atributu eta azpi-elementuak:

- *eid*: izen-entitatearen identifikatzailea.
- *type*: izen-entitate mota. Zortzi balio izan ditzake:
  - *person*: izen-entitatea pertsona bat da.
  - *organization*: erakundeak dira hauek.
  - *location*: lekuzko izen-entitatea.
  - *date*: datek mota hau daramate.
  - *time*: denbora adierazten duten izen-entitateak.
  - *money*: monetentzako.
  - *misc*: bestelako izen-entitateak.

12. taulan izen-entitate mota bakoitzaren hautazko atributuak erakusten dira.

`<entity>` elementu bakoitzak honako azpi-elementuak izan ditzake:

- *references*: elementu honek `<span>` elementu bat edo gehiago darama.
- *externalReferences* (hautazkoa): elementu honek `<externalRefs>` elementu bat edo gehiago darama.

`<span>` azpi-elementua izen-entitate berberaren agerpenak erreferentziatzeko erabiltzen da. Zein termino erreferentziatzen diren adierazteko `<target>` azpi-elementua erabiliko genuke, izen-entitatea termino batek baino gehiagok osatzen badute hainbat `<target>` azpi-elementu erabiliko beharko ditugu. Hauexek dira `<target>` azpi-elementuaren atributuak:

- *id*: terminoaren identifikatzailea, hots terminoaren *tid* atributua.
- *head* (hautazkoa): "yes"balioa terminoa izen-entitatearen termino nagusia dela adierazteko.

Hautazko `<externalRef>` azpi-elementua terminoak baliabide lexiko edo semantikoekin lotzeko erabiltzen da. Hauek dira bere atributuak:

- *resource*: baliabidearen identifikatzailea.
- *reference*: erreferentziatzen den elementuaren kodea.
- *confidence* (hautazkoa): konfidantza balioa.

HAP masterra

Mota	**Hautazko atributuak**
organization	subtype=”company”
location	subtype=leku mota (adib. ”kalea”, ”estatua”, ”herria”)
date	dateISO[28]=”2012/12/31”
time	timeISO[29]=”15:38:00”
money	moneyISO[30]=”15:38:00”
misc	subtype=”car” *country*[31]: automobilaren erregistrazio kodea
misc	subtype=”phone” *phonetype*: telefono zenbaki mota (adib. ”imei”, ”mobile”, ”landline”) *country*: herrialdearen kodea
misc	subtype=”personal” *cardtype*: txarpel pertsonal mota (adib. ”passport”, ”idcard”, ”driver_license”) *country*: herrialdearen kodea
misc	subtype=”banking” *banktype*: banku entitate mota (adib. ”iban”, ”ccard”, ”account”) *country*: herrialdearen kodea
misc	subtype=”internet” *nettype*: sare entitate mota (adib. ”ipaddress”, ”macaddress”, ”email”, ”url”)

Taula 12: Izen-entitateen azpi-motak eta hautazko atributuak

`<entities>` geruzaren erabilera argitzeko beheko adibidea ikus dezakegu:

```
<entities>
  <entity type="person" eid="e1">
    <references>
      <!-- Jules Ferry -->
      <span>
        <target id="t1"/>
        <target id="t2"/>
      </span>
      <!-- lui -->
      <span>
        <target id="t1"/>
      </span>5
    </references>
    <externalReferences>
      <externalReference confidence="0.7"
                         reference="13982343" resource="JRCNames"/>
```

```
      <externalReference confidence="0.3"
                         reference="834354" resource="JRCNames"/>
    </externalReferences>
  </entity>
</entities>
```

Beraz, funtsean proiektu honetarako garatzen den edozein izen-entitateen detektatzaile eta sailkatzailek honako sarrera/irteera diseinua bete behar du:

1. Sarreran morfologikoki etiketaturiko KAF dokumentu izan behar du. Demagun *pos-tagged.kaf* izeneko dokumentua dugula:

```
<KAF xml:lang="fr" version="v1.opener">
  <kafHeader>
        ⋮
  </kafHeader>
  <text>
    <wf wid="w1" sent="1" para="1" offset="0" length="2">J'</wf>
    <wf wid="w2" sent="1" para="1" offset="2" length="4">aime</wf>
    <wf wid="w3" sent="1" para="1" offset="7" length="6">manger</wf>
    <wf wid="w4" sent="1" para="1" offset="14" length="2">au</wf>
    <wf wid="w5" sent="1" para="1" offset="17" length="10">restaurant</wf>
    <wf wid="w6" sent="1" para="1" offset="28" length="10">Luxembourg</wf>
    <wf wid="w7" sent="1" para="1" offset="38" length="1">.</wf>
  </text>
  <terms>
    <!--J'-->
    <term tid="t1" type="close" lemma="J'" pos="Q" morphofeat="CL1ms">
      <span>
        <target id="w1" />
      </span>
    </term>
    <!--aime-->
    <term tid="t2" type="open" lemma="aime" pos="V" morphofeat="VP3s">
      <span>
        <target id="w2" />
      </span>
    </term>
    <!--manger-->
    <term tid="t3" type="open" lemma="manger" pos="V" morphofeat="VW">
      <span>
        <target id="w3" />
      </span>
    </term>
```

```
    <!--au-->
    <term tid="t4" type="close" lemma="au" pos="P" morphofeat="P">
      <span>
        <target id="w4" />
      </span>
    </term>
    <!--restaurant-->
    <term tid="t5" type="open" lemma="restaurant" pos="V" morphofeat="VG">
      <span>
        <target id="w5" />
      </span>
    </term>
    <!--Luxembourg-->
    <term tid="t6" type="open" lemma="luxembourg" pos="N" morphofeat="NPms">
      <span>
        <target id="w6" />
      </span>
    </term>
    <!--.-->
    <term tid="t7" type="close" lemma="." pos="O" morphofeat=".">
      <span>
        <target id="w7" />
      </span>
    </term>
  </terms>
</KAF>
```

2. Agindu honekin exekutatuko genuke izen-entitateen modulua:

```
$ cat postagged.kaf | nerc
```

3. Irteeran KAF dokumentu bat izango dugu detektaturiko izen-entitiateekin:

```
<KAF version="v1.opener" xml:lang="fr">
  <kafHeader>
      ⋮
    <linguisticProcessors layer="entities">
      <lp name="opennlp-fr-ner-all"
                  timestamp="2014-05-18T15:51:27Z" version="1.0"/>
    </linguisticProcessors>
  </kafHeader>
  <text>
    ⋮
  </text>
```

```
  <terms>
    ⋮
  </terms>
  <entities>
    <entity eid="e1" type="location">
      <references>
        <span>
          <!-- Luxembourg -->
          <target id="t6"/>
        </span>
      </references>
    </entity>
  </entities>
</KAF>
```

4. Hizkuntza-detektatzailea, tokenizatzailea, etiketatzaile-morfologikoa eta izen-entitateen detektatzaile eta sailkatzailea kateatzea posible da emaitza berbera lortzeko:

```
$ echo "J'aime manger au restaurant Luxembourg." \
        | language-identifier | tokenizer | postagger | nerc
```

4.4.1 OpenNLP bidez entrenaturiko Izen-Entitateen Detektatzaile eta Sailkatzailea

Izen-Entitateen moduluak OpenNLP bidez entrenaturiko gainbegiratutako MaxEnt (Maximum Entropy) modelo bat erabiltzen du (4.3.1. atalean azaltzen da MaxEnt algoritmoa) 17. Conference on Text, Speech and Dialogue (TSD2014) konferentzian aurkeztuko dena (Azpeitia et al., 2014). Ekbal eta Bandyopadhyay-ren ikerlanen arabera (Ekbal eta Bandyopadhyay, 2008) Support Vector Machines-etan (SVM) oinarritutako modeloak entrenatzerako orduan kategoria morfologia oso informazio baliagarria da izen-entitateak detektatzeko, baina zoritxarrez OpenNLP-ren izen-entitateen *opennlp.tools.namefind.NameFinderME* liburutegia ez dago prest kategoria morfologikoan oinarritutako ezaugarriak erauzteko. Proiektu honetarako OpenNLP-ko izen-entitateen liburutegia hedatu dugu etiketatzaile-morfologiko baten laguntzaz kategoria morfologikoak kalkulatu eta prozesatzeko.

OpenNLP prestatuta dago testu batetik ezaugarriak bere kabuz ikasteko, *FeatureGenerator* izeneko prozesatzaileek egiten dute lan hori. Sailkatzaileak ikasteko guk nahi ditugun *FeatureGenerator*-ak erabili ditzakegu eta gainera parametro batzuen laguntzaz konfiguratu ditzakegu. Hauek dira erabilitako *FeatureGenerator*-ak:

- **Sentence Boundaries**. Izen-entitatea esaldiko lehen edo azken elementua izateak garrantzia duen ala ez. Bi balio bitarren bidez konfiguratzen da, lehenengoak esaldien hasierarako eta bestea bukaerarako.

HAP masterra

- **Neighbouring tokens**. Izen-entitatearen alboko tokenak, alboko token-patroiak (adibidez, $tok_{-2} = le, tok_{-1} = hotel, tok = lekuzko\_izen\_entitatea, tok_{+1} = zenbakia, tok_{+2} =$ \$) edota alboko token-klaseak (letrak, zenbakiak, sinboloak edo puntuazio ikurrak) erabiltzen dira ezaugarriak erauzteko. Konfigurazioan tokenak, token-patroiak edota token-klaseak adierazi dezakegu eta bakoitzerako bi zenbaki adierazi ditzakegu, lehenengoak uneko tokenaren ezkerrera zenbat token analizatzen diren adierazten du, eta bigarrenak gauza berbera adierazten du tokenaren eskuinera. Adibidez, konfigurazio posible bat hau izango litzateke:
  - token: true, 3, 3.
  - token-pattern: false.
  - token-class: true, 2, 1.
- **Bigrama leihoa**. Bigramak aztertzen dira izen-entitate bakoitzaren ezker eta eskubira. *Neighbouring token* parametroaren antzera, ezker eta eskuinera zenbat bigrama aztertzen diren adierazi dezakegu.
- **Aurrizki eta atzizki leihoa**. Aurrizki eta atzizkiak gehienez 4 karakterekoak dira. Token batek 3 karaktere baditu, aurrizkia eta atzizkia tokena bera izango lirateke. Ezker eta eskuin zenbat aurrizki eta atzizki hartzen diren zehazten da.
- **Charngram luzera**. Izen-entitate bakoitzerako karaktere n-gramak analizatzen dira. Parametroaren balioak n-grama luzera minimoa eta maximoa dira. Adibidez, balio minimo eta maximoa 2 eta 3 badira hurrenez hurren, izen-entitateen bigrama eta trigrama guztiak prozesatzen dira:
  - charngram=2, 3.
  - izen-entitatea=Hotel_Paris.
  - bigramak=Ho, ot, te, el, l_, _P, Pa, ar, ri, is.
  - trigramak=Hot, ote, tel, el_, l_P, _Pa, Par, ari, ris.
- **Kategoria-morfologiko leihoa**. Hauxe da OpenNLP liburutegien hedapenari esker sortu dugun parametro berria. Izen-entitateen ezker eta eskuin zenbat tokenen kategoria-morfologiko begiratzen diren zehazten da.
- **Cutoff**. Ezaugarri berbera zenbat aldiz pasa behar den kontutan hartzeko zehazten da parametro honen bidez.
- **Iterazio kopurua**. 4.3.1. atalean azaltzen den Generalised Iterative Scaling (GIS) prozeduran zenbat iterazio egiten diren finkatzen da.

Entrenamendu formatuari dagokionez lerro bakoitzean tokenizatutako esaldi bat behar dugu. Izen-entitateak identifikatzeko etiketa berezi batzuk erabili behar dira izen-entitatearen hasieran eta bukaeran, `<START:kategoria>` eta `<END>` ditugu etiketa hauek. Nahiz eta oso

entrenamendu formatu erraza izan badu desabantaila handi bat, ezin ditugu izen-entitateak bata bestearen barnean jarri. Hona hemen adibide bat:

```
Je m' appelle <START:person> Andoni Azpeitia <END> .
J' habite au <START:location> Donostia <END> .
```

Nahiz eta proiektu honetako KAF formatuak zazpi motatako izen-entitate bereizi, entrenamendurako corpusaren mugak direla medio sailkatzaileak sei kategoria ezberdin bereizten ditu: pertsonak, lekuak, erakundeak, datak, denbora eta monetak.

4.4.2 ESTER corpusa

Izen-entitate detektatzaile eta sailkatzailea entrenatzeko ESTER corpusa (Galliano et al., 2005) erabili dugu. ESTER corpusa Frantziako sei irrati saioetako transkripzioek osatzen dute. Transkripzioek 1700 ordu baino gehiago biltzen dituzte hauetatik 100 ordu eskuz egin direlarik. Corpusaren tamaina 13. taulan ikus daiteke.

ESTER corpusa	
Hitz-kopurua	1.200.000
Hitz-hiztegia	37.000
Izen-entitate kopurua	74.082
Izen-entitate hiztegia	15.152

Taula 13: ESTER corpusaren tamaina

Corpusean aurki ditzakegun izen-entitateek honako kategoriatan sailkatzen dira:

- Pertsonak (pers): gizakiak, pertsonaiak eta animaliak.
- Lekuak (loc): entitate geografikoak, helbideak, errepideak, etab.
- Erakundeak (org): edozein motatako erakundeak.
- Talde geo-sozio politikoak (gsp): familiak, nazioak eta lurralde administratiboak.
- Monetak (amount): diru-kantitateak.
- Denbora (time): denborazko izen-entitateak.
- Productuak (prod): artelanak, inprimakiak, sariak eta automobilak.
- Instalazioak (fac): eraikinak eta monumentuak.

Corpusa hiru zatitan banatzen da: entrenamendurako zatia (%77,8), garapenerako zatia (%9,7) eta ebaluaziorako zatia (%12,5). Sei hilabeteko aldea dago entrenamendu/garapen eta ebaluazio zatien artean: entrenamendu zatian 2002tik 2003rako irrati saioak ditugu, eta ebaluazio zatian 2004ko transkripzioak dauzkagu. Gainera, ebaluazio zatian bi irratsaio berri daude. ESTER corpusaren berezitasun garrantzitsuenetako bat izen-entitate

etiketen arteko anbiguetatea da (adibidez, lurralde administratibo eta leku geografiko ugari anbiguoak dira): entrenamendu zatian %40-ko anbiguotasuna dugu eta ebaluazio zatian %32-koa.

Horretaz gainera, azpimarratu beharra dago agian ESTER corpusaren domeinua ez dela guztiz egokia, transkripzioetan ahozko hizkuntzan bakarrik gertatzen diren espresioak aurkitzen baitira, ohikoa da *ehh, ehh...* bezalako esamoldeak aurkitzea eta igorlea nahastea.

ESTER corpusa XML dokumentuez osatzen da. Tamalez testua ez dago tokenizatuta eta izen-entitateen parte ez den testu-zatia letra xehez idatzita dago, beraz aurreprozesaketa bat egitea beharrezkoa da. Hemen duzue XML dokumentu baten zati bat:

```
<Event desc="org" type="entity" extent="begin"/>
le gouvernement
<Event desc="org" type="entity" extent="end"/>
et
<Event desc="org" type="entity" extent="begin"/>
l'opposition
<Event desc="org" type="entity" extent="end"/>
enterrent la polmique sur la mission de
...
```

4.4.3 Ebaluzaioa

Ebaluazioa ESTER corpusaren garapen eta ebaluazio zatien gainean egin da. 15 eta 14.tauletan ikus daiteke izen-entitate mota bakoitzarekin izandako arrakasta garapen eta ebaluazio zatian.

Kategoria	Precision	Recall	F-Measure
Location	88,56	85,84	87,18
Person	88,59	80,84	84,54
Time	89,61	81,66	85,45
Date	84,03	74,59	79,03
Organization	76,79	55,42	64,38
Money	74,19	25,14	37,55
Total	86,20	75,85	80,69

Taula 14: Izen-entitateen ebaluazioa ESTER corpusaren ebaluazio zatian

Tauletan erakusten denez orokorrean emaitza onak dira baina badirudi erakunde eta monetekin arazoak daudela. Erakunde eta moneta gutxi detektatzearen arrazoi posible bat ESTER corpuseko kategorien anbiguetatea da (gogoratu entrenamendu zatian %40-koa dela eta ebaluazio zatian %32-koa). Erakundeen kasuan beste arrazoi bat erakunde kategoriaren berezko heterogenotasuna da, izan ere, mota askotako erakundeak aurki ditzakegu

Corpusa	Precision	Recall	F-Measure
Garapenerako zatia	91.5	85.39	88.34
Ebaluaziorako zatia	86.2	75.85	80.69

Taula 15: ESTER corpuseko garapen eta ebaluzio zatietan izandako emaitzen arteko konparaketa

(erakunde politikoak, pertsona-izenak dituztenak, leku-izena dituztenak, kirolarekin zerikusi dutenak, eta abar). Deigarria den beste fenomeno bat garapen eta ebaluazio zatien arteko aldea da, ikasketa gainbegiratuan beti egiten da *overfitting* apur bat garapen zatiaren gainean, baina horretaz gainera gure kasuan ebaluazio zatian bi irratsaio berri ditugu gai eta izen-entitate berriekin.

16.taulan KAF dokumentu bat prozesatzeko denbora ikus dezakegu. Aurreko moduluekin konparatuta denborak gora egin du, baina ez da harritzekoa, hizkuntza-teknologiaren aldetik orain arteko modulurik konplexuenaren aurrean gaude eta, gainera denbora gehiena entrenatutako sailkatzailea memorian kargatzen ematen du eta hori lehenengo prozesuak bakarrik egiten du.

Dokumentu kopurua	Denbora segundotan
1	2,75
2	3,42
5	7,55
10	15,12
20	30,55

Taula 16: KAF dokumentuak prozesatzeko denbora

4.5 Polaritate-etiketatzailea

Polaritate-etiketatzailearen xedea frantsesez idatzitako testu batean nolabaiteko sentimena adierazten duten hitzak etiketatzea izango da. Adibidez, *"Le restaurant Zuberoa est très bon"* esaldian *"très"* eta *"bon"* hitzak positiboki etiketatu behar lirateke. Terminoen etiketak bi multzotan bana ditzakegu 17. taulan adierazten den bezala.

Proiektuko moduluetan ohizkoa den bezala polaritate-etiketatzaileak sarrerako KAF dokumentua prozesatuko du eta irteeran dokumentu berbera utziko du polaritate informazioarekin. Modulu honek izen-entitateen moduluak utzitako KAF dokumentua izango du sarreran eta sentimendua `<terms>` geruzan idatziko du. Hona hemen KAF formatuan sentimendua nola etiketatzen den:

`<terms>` geruzako `<term>` elementu bakoitzaren barne `<sentiment>` elementu bat egongo da atributu hauekin:

HAP masterra

Nolabaiteko polaritatea adierazten dutenak	Polaritate positiboa
	Polaritate neutrala
	Polaritate negatiboa
Polaritatea aldatzen dituztenak	Polaritate indartzaileak
	Polaritate ahultzaileak
	Polaritate trukatzaileak

Taula 17: Terminoen sentimendu etiketa-motak

- *resource*: kanpoko sentimendu-baliabidearen erreferentzia edo identifikatzailea.
- *polarity* (hautazkoa): polaritatearen balioa. Hauek dira balio posibleak:
  - *positive*: polaritate positiboa.
  - *negative*: polaritate negatiboa.
  - *neutral*: polaritate neutrala.
- *strength* (hautazkoa): polaritatearen indarra adierazten du. Hauek dira izan ditzaken balioak:
  - *weak*: indar ahula.
  - *average*: nola-halako indarra.
  - *strong*: indar handia.
  - Zenbakizko balioa.
- *subjetivity* (hautazkoa): polaritatearen subjektibotasuna adierazten du. Hauek dira bere balioak:
  - *subjective* edo *objective*: polaritatea objektibo edo subjektiboa den.
  - *factual* edo *opinionated*: polaritatea gertaera edo datu objektiboetan oinarrituta dagoen ala ez.
- *sentiment_semantic_type* (hautazkoa): polaritatearekin erlazionatutako mota-semantikoak:
  - *aesthetics_evaluation*.
  - *moral_judgment*: epaiketa moral bat egiten bada.
  - *emotion*: emozioa adierazten bada.
- *sentiment_modifier* (hautazkoa): polaritatea aldatzen duten terminoentzat erabiltzen da honako balio hauekin:
  - *intensifier*: polaritate indartzaileak.
  - *weakener*: polaritate ahultzaileak.

  - *polarity_shifter*: polaritate trukatzaileak.

- *sentiment_marker* (hautazkoa): termino hauek soilik ez dute polaritaterik baina bai zenbait egoeratan (pentsatu, aurkitu, nire ustez, etab.). Atributuaren balioa terminoa bera da.
- *sentiment_product_feature* (hautazkoa): zein domeinutan aplikatzen den terminoa.

Hona hemen adibide bat:

```
<term lemma="super" pos="G" tid="t93" type="open">
  <span>
    <!--super-->
    <target id="w93"/>
  </span>
  <sentiment polarity="positive"
             resource="General domain lexicon for French"/>
</term>

<term lemma="tout" pos="P" tid="t122" type="close">
  <span>
    <!--tout-->
    <target id="w122"/>
  </span>
  <sentiment sentiment_modifier="intensifier"
             resource="General domain lexicon for French"/>
</term>
```

Modulu hau proiektuan nola integratzen den ikusteko aurreko moduluetan ikusi dugun adibidearekin jarraituko dugu.

- Sarreran *entities.kaf* bezalako dokumentu bat behar dugu izen-entitate analisia egin da:

```
<KAF version="v1.opener" xml:lang="fr">
  <kafHeader>
      ⋮
  </kafHeader>
  <text>
    <wf length="2" offset="0" para="1" sent="1" wid="w1">J'</wf>
    <wf length="4" offset="2" para="1" sent="1" wid="w2">aime</wf>
    <wf length="6" offset="7" para="1" sent="1" wid="w3">manger</wf>
    <wf length="2" offset="14" para="1" sent="1" wid="w4">au</wf>
    <wf length="10" offset="17" para="1" sent="1" wid="w5">restaurant</wf>
    <wf length="10" offset="28" para="1" sent="1" wid="w6">Luxembourg</wf>
    <wf length="1" offset="38" para="1" sent="1" wid="w7">.</wf>
```

```
</text>
<terms>
  <term lemma="J'" morphofeat="CL1ms" pos="Q" tid="t1" type="close">
    <span>
      <!-- J' -->
      <target id="w1"/>
    </span>
  </term>
  <term lemma="aime" morphofeat="VP3s" pos="V" tid="t2" type="open">
    <span>
      <!-- aime -->
      <target id="w2"/>
    </span>
  </term>
  <term lemma="manger" morphofeat="VW" pos="V" tid="t3" type="open">
    <span>
      <!-- manger -->
      <target id="w3"/>
    </span>
  </term>
  <term lemma="au" morphofeat="P" pos="P" tid="t4" type="close">
    <span>
      <!-- au -->
      <target id="w4"/>
    </span>
  </term>
  <term lemma="restaurant" morphofeat="VG" pos="V" tid="t5" type="open">
    <span>
      <!-- restaurant -->
      <target id="w5"/>
    </span>
  </term>
  <term lemma="luxembourg" morphofeat="NPms" pos="N" tid="t6" type="open">
    <span>
      <!-- Luxembourg -->
      <target id="w6"/>
    </span>
  </term>
  <term lemma="." morphofeat="." pos="O" tid="t7" type="close">
    <span>
      <!-- . -->
      <target id="w7"/>
    </span>
```

```
    </term>
  </terms>
  <entities>
     ⋮
  </entities>
</KAF>
```

- Komando honekin egiten da sentimendu analisia:

```
$ cat entities.kaf | polarity-tagger
```

- Honako hau da komandoaren irteera:

```
<KAF version="v1.opener" xml:lang="fr">
  <kafHeader>
     ⋮
  </kafHeader>
  <text>
     ⋮
  </text>
  <terms>
    <term lemma="J'" morphofeat="CL1ms" pos="Q" tid="t1" type="close">
      <span>
        <!-- J' -->
        <target id="w1"/>
      </span>
    </term>
    <term lemma="aime" morphofeat="VP3s" pos="V" tid="t2" type="open">
      <span>
        <!-- aime -->
        <target id="w2"/>
      </span>
      <sentiment polarity="positive"
                 resource="General domain lexicon for French .
                 Vicomtech_general_lexicon_french"/>
    </term>
    <term lemma="manger" morphofeat="VW" pos="V" tid="t3" type="open">
      <span>
        <!-- manger -->
        <target id="w3"/>
      </span>
    </term>
    <term lemma="au" morphofeat="P" pos="P" tid="t4" type="close">
      <span>
```

```
        <!-- au -->
        <target id="w4"/>
      </span>
    </term>
    <term lemma="restaurant" morphofeat="VG" pos="V" tid="t5" type="open">
      <span>
        <!-- restaurant -->
        <target id="w5"/>
      </span>
    </term>
    <term lemma="luxembourg" morphofeat="NPms" pos="N" tid="t6" type="open">
      <span>
        <!-- Luxembourg -->
        <target id="w6"/>
      </span>
    </term>
    <term lemma="." morphofeat="." pos="O" tid="t7" type="close">
      <span>
        <!-- . -->
        <target id="w7"/>
      </span>
    </term>
  </terms>
  <entities>
     ⋮
  </entities>
</KAF>
```

- Hizkuntza-detektatzailea, tokenizatzailea, etiketatzaile-morfologikoa, izen-entitateen detektatzaile eta sailkatzailea eta polaritate-etiketatzailea kateatzea posible da emaitza berbera lortzeko:

```
$ echo "J'aime manger au restaurant Luxembourg." \
        | language-identifier | tokenizer | postagger | nerc \
        | polarity-tagger
```

Aurreko moduluetan ez bezala polaritate-etiketatzailearen muina baliabide-lexikoa da, bertan azaltzen baita nolabaiteko polaritatea duen lexikoa. Polaritate-etiketatzaile moduluak benetan egiten duena termino bakoitzaren hitz-forma eta kategoria-morfologikoarekin baliabide-lexikoa kontsultatzea. Baliabide-lexikoan termino bat aurkitzean denean bertako informazioarekin terminoa osatzen da KAF dokumentuan. Baliabide lexikoaren xehetasunak 4.5.1. atalean azaltzen dira.

HAP masterra

4.5.1 Sentimendu baliabide-lexikoa

Atal honetan azaltzen den sentimendu baliabide-lexikoa 9. Language Resources and Evaluation Conference (LREC2014) konferentzian aurkeztu den ikerlan baten parte da (Maks et al., 2014). Baliabide-lexikoak polaritatearen informazioaz gain kategoria gramatikala beharrezkoa du, izan ere, lexikoa anbiguoa da eta kategoriari esker anbiguotasun kasu asko ebatzi daitezke. Terminoak adierekin lotzea ere lagungarria da, horrela beste hizkuntzetako adiera berberekin lotura egiteko gai izango gara. Adierak identifikatzeko synset-ak (sinonimo-talde identifikatzailea) erabiliko ditugu. Baliabide-lexikoa sortzeko 4.5.1.1. azpiatalean azalduko den LMF (*Lexical Markup Framework*) formatua erabiltzea erabaki da. LMF formatua gaur eguneko estandarrak jarraituz informazio lexikoa biltzeko helburuarekin asmatu zen. Gainera, XML meta-lengoaian oinarritzen da LMF formatua.

4.5.1.2. atalean ikusiko dugu sortze prozesua baina ideia global bat izatearren, 1000 hitz inguruko bi lista izango dira abiapuntuak, bata polaritate positiboko hitzekin eta bestea polaritate negatiboarekin. Lista horietako hitz bakoitzeko dagokion synset-a bilatuko da eta dagokion synset-eko hitz (*variant*) bakoitzarekin bi listak osatu eta gero, erlazio semantiko batzuen bidez (sinonimia eta antonimia esate baterako) WordNET-an (aurrerago azalduko da WordNET-ei buruzko informazioa) zehar zerikusia duten synset-ak bilatuko dira. Hurrengo pausua synset bakoitza osatzen duten hitz guztiak LMF fitxategi batean biltzea izango da, eta azkenik, polaritatea nolabait aldatzen duten lexikoiekin osatuko da LMF fitxategia, izan ere *"polita"* hitza positiboa da, baina *"batere"* hitza aurretik kateatzen badiogu (*"batere polita"*) esanahia guztiz aldatzen da.

4.5.1.1 LMF formatua

Baliabide-lexikoen produkzio, mantenu eta hedapenak izugarrizko garrantzia dauka LNP (Lengoaia Naturalen Prozesamendua) munduan. Kontutan eduki beharreko beste kontu bat aplikazio ezberdinen arteko integrazioa da. Aipatutako bi arazoak kontuan hartuta LMF (*Lexical Markup Framework*) formatuaren helburua LNP tresnentzako baliabide lexiko estandarizatu bat sortzea da.

Bi mailan antolatutako formatu bat diseinatu da estandar-multzo koherente bat osatzeko:

- Goi-mailako zehaztapenek estandarizatutako atributu-izen eta konstanteen bidez osatutako klaseak definitzen dituzte.
- Behe-mailako zehaztapenak estandarizatutako atributu-izen eta konstanteak finkatzen dituzte.

LMF formatuaren diseinuak jarraitzen dituen bi estandar nagusiak UML (Unified Modeling Language) eta XML dira. UML dokumentuak eta datu-egiturak modelatu ahal izateko balio duen diagramak eraikitzeko lengoaia da. UML lengoaia aukeratzeko arrazoiak honako hauek dira:

HAP masterra

- UML industria munduan erabiltzen den *de facto* estandarra da, beraz, aditu askok erraz ulertuko dute.
- UML ondo definitutako eta dokumentatutako lengoaia da.
- Egitura bat erakusteko orduan diagramen erabilpena oso lagungarria da.
- UML-k diseinua zatitzeko aukera ematen du modulu bakoitza modu independentean lantzeko aukera emanez.
- Tresna ahaltsu asko daude UML lengoaia inplementatzen dituztenak.

Egitura

LMF formatua bi osagai hauek osatzen dute:

- **Elementu nagusia**. Sarrera lexikoen oinarrizko informazioa duen egitura da.
- **Elementu nagusiaren hedadurak**. Baliabide lexiko konkretu baterako sarrera lexiko bakoitzaren informazioa hedatzen duen elementuak.

Gure beharren arabera elementu nagusiaren hedadura batzuk edo beste erabiliko ditugu. Adibidez, analizatzaile sintaktiko bat garatu nahi bada lexikoiaren informazio sintaktikoa hedadura sintaktikoan bertan gordeko da. Hainbat hedadura erabiltzea posible da, baina jakina, hedadura guztiek elementu nagusiaren menpe egon behar dute. Elementu nagusiaren hedadurak honako hauek dira:

- **Morfologia**. Lexikoiaren informazio morfologikoa bildu ahal izateko morfologia hedadura erabiliko dugu.
- **Sintaxia**. Hedadura honetan egitura sintaktikoa kudeatzea da.
- **Semantika**. Helburua esanahi bat eta berarekin erlazionatutako besteekin dauden erlazioak deskribatzea da. Hauxe da guk erabiliko dugun hedadura.

Adibidea

Arestian azaldu dugu LMF formatua UML diagramen bidez adierazitako egitura bat jarraitzen duela eta LMF formatua XML fitxategien bidez inplementatzen dela. Hemen duzue XML fitxategi bat LMF formatua jarraitzen duena:

```
<LexicalEntry>
  <DC att="partOfSpeech" val="noun"/>
  <Lemma>
    <DC att="writtenForm" val="oak tree"/>
  </Lemma>
  <Sense id="oak_tree0" synset="12100067"/>
```

HAP masterra

```
</LexicalEntry>
<LexicalEntry>
  <DC att="partOfSpeech" val="noun"/>
  <Lemma>
    <DC att=writtenForm" val="oak"/>
  </Lemma>
  <Sense id="oak0" synset="12100067"/>
  <Sense id="oak2" synset="12100739"/>
</LexicalEntry>
<Synset id="12100067">
  <SemanticDefinition>
    <DC att="text" val="a deciduous tree of the genus Quercus"/>
    <Statement>
      <DC att="text" val="has acorns and lobed leaves"/>
    </Statement>
    <Statement>
      <DC att="text" val="great oaks grow from little acorns"/>
    </Statement>
  </SemanticDefinition>
  <SynsetRelation targets="12100739"
    <DC att="label" val="substanceHolonym"/>
  </SynsetRelation>
</Synset>
<Synset id="12100739">
  <SemanticDefinition>
    <DC att="text" val="the hard durable wood of any oak"/>
    <Statement>
      <DC att="text" val="used especially for furniture and flooring"/>
    </Statement>
  </SemanticDefinition>
</Synset>
```

4.5.1.2 LMF fitxategiaren sortze-prozesua

Atal honetan barrena baliabide lexikoaren sortze-prozesua nolakoa den argitzen saiatuko gara. Horretan hasi aurretik lehendabizi beharrezkoak diren dokumentuak zeintzuk diren eta zertarako balio duten ulertu beharra dago.

4.5.1.2.1 Beharrezko dokumentuak

Baliabide lexikoaren sortze-prozesurako beharrezko informazioa biltzen duten dokumen-

tuak zeintzuk diren eta zertarako erabiltzen diren jakitea ezinbestekoa da, dokumentu horiek WordNET-a, polaritate-listak, erlazio-dokumentua, frekuentzia-lista eta polaritatea aldatzen duten hitzak ditugu.

WordNET-a LMF formatuan

Dokumentu honetan azaltzen den baliabide lexikoan bilduko den lexikoi guztia WordNET batetik aterako dugu. WordNET bat terminoen esanahien inguruan ezagutza-lexikoa biltzen duen informazio-iturria da, eta bertan esanahiak sinonimo-multzoak diren *synset* (*synonym-set*) izeneko taldetan biltzen dira, hortaz, *synset* bakoitzaren barne esanahi berbera duten hitzak aurkituko ditugu. Hitz horiek *variant* izena hartzen dute.

WordNET-en abantaila nagusia bertako synset-en barne antolaketa da, hauek elkarren artean lotzen dira beraien arteko erlazio semantikoaren arabera. Erlazio semantikoak mota askotakoan izan daitezke baina arruntenak sinonimia, antonimia, hiponimia eta hiperonimia bezalako erlazioak dira. Gure lanerako frantses hizkuntzarako lizentzia librepean eskura dagoen WOLF WordNET-a (Sagot et al., 2008) aukeratu dugu.

Prozesatu ahal izateko WordNET-ak LMF formatuan egon behar du. Lehenago ikusi dugu LMF formatuak LNP-rako elementu ugari eskaintzen dituela, gure lanerako hedadura semantikoa bakarrik beharko dugu, bereziki *Sense* eta *Synset* azpiklaseak. Hona hemen fitxategiaren adibide bat:

```
<LexicalResource>
  <GlobalInformation label="Proposal for Kyoto-internal
                            WordNet representation"/>
  <Lexicon language="fre" languageCoding="ISO 639-3">
    <LexicalEntry id="LE-115.97.117.109.117.114.101_v">
      <Lemma partOfSpeech="v" writtenForm="saumure"/>
      <Sense id="S-115.97.117.109.117.114.101_v_1"
             synset="fre-30-00216561-v"/>
    </LexicalEntry>
    <LexicalEntry id="LE-115.97.117.114.111.112.111.100.97_n">
      <Lemma partOfSpeech="n" writtenForm="sauropoda"/>
      <Sense id="S-115.97.117.114.111.112.111.100.97_n_1"
             synset="fre-30-01708778-n"/>
      <Sense id="S-115.97.117.114.111.112.111.100.97_n_2"
             synset="fre-30-01708998-n"/>
    </LexicalEntry>
        ⋮
    <Synset id="fre-30-07365024-n" baseConcept="0">
      <SynsetRelations>
```

```
        <SynsetRelation relType="has_hyponym" target="fre-30-07365193-n"/>
        <SynsetRelation relType="related_to" target="fre-30-01369758-v"/>
        <SynsetRelation relType="related_to" target="fre-30-02530003-v"/>
        <SynsetRelation relType="related_to" target="fre-30-02530003-v"/>
        <SynsetRelation relType="related_to" target="fre-30-01369758-v"/>
      </SynsetRelation>
    </SynsetRelations>
  </Synset>
  <Synset id="fre-30-07365193-n" baseConcept="0">
    <SynsetRelations>
      <SynsetRelation relType="related_to" target="fre-30-02523521-v"/>
      <SynsetRelation relType="related_to" target="fre-30-02523521-v"/>
      <SynsetRelation relType="has_hyperonym" target="fre-30-07365024-n"/>
      <SynsetRelation relType="gloss" target="fre-30-02528380-v"/>
      <SynsetRelation relType="gloss" target="fre-30-00029378-n"/>
      <SynsetRelation relType="gloss" target="fre-30-00008007-r"/>
    </SynsetRelations>
  </Synset>
       ⋮
</LexicalResource>
```

Adibidean ikusten denez lehendabizi sarrera lexiko guztiak azaltzen dira bakoitzak izan ditzakeen esanahiei buruzko informazio guztiarekin, eta gero, synset elementuaren bidez esanahien arteko lotura egiten da.

Polaritate-listak

Polaritate-lista polaritate jakin bateko hitzak biltzen dituen lista bat baino ez da. Lista hauek oinarritzat erabiliko ditugu LMF formatuan dagoen WordNET-ean zehar listako elementuekin erlazionaturiko lexikoia lortzeko. Polaritateari buruzko informazio semantikoa duten hitzak biltzen dira, horregatik gehienbat adjektiboak aurkituko ditugu polaritate-listetan. Lan honetarako 1000 hitz inguruko bi lista erabiliko ditugu: bata polaritate positiboko hitzekin eta bestea polaritate negatiboarekin.

Erlazio-dokumentua

Lehenago aipatu dugunez polaritate-listak erabiliko ditugu WordNET-ean zehar erlazionatutako lexikoia lortzeko, baina ez ditugu edozein erlazio kontuan hartu behar, polaritatearekin zerikusia dutenak bakarrik dira interesgarriak. Horretarako erabiliko dugu erlazio-dokumentua, erabili nahi ditugun erlazioak espezifikatzeko.

HAP masterra

Argi dago sinonimo eta antonimo-erlazioak funtsezkoak direla, hitz batek polaritate positiboa badu esanahi bereko hitzek ere polaritate berbera izango dute, eta antonimoen kasuan hitz batek esanahi positiboa badu bere antonimoek aurkako polaritatea adieraziko dute. Beste erlazio bat ere erabiliko ditugu: hiponimia. Hitz baten polaritatea positiboa bada bere hiponimoek ere lehenbizikoaren ezaugarri semantiko guztiak izateagatik polaritate berbera dute.

Erlazio-dokumentuak erabiliko ditugun erlazioen alboan pisu bat izango du. Esate baterako, sinonimia erlazioak 2 pisua badu sinonimia erlazioaren bidez kalkulatzen diren polaritateak zenbaki horrekin biderkatuko dira, eta pisua negatiboa bada polaritatea aurkakoa izango da. Baliteke hitz batek aldi berean beste hainbat hitzekin erlazionatuta egotea, kasu horietan pisu hori bozketa ponderatzeko ere erabiliko da.

Hona hemen erabilitako erlazio-dokumentua:

```
near_synonym 2
near_antonym -2
has_hyponym 1
```

Frekuentzia-lista

Lista honetan hizkuntza jakin batean (gure kasuan frantsesa) gehien erabiltzen diren hitzak ordenaturik azaltzen dira gehienetik gutxienera. Lan honetarako milioi bat hitzetako lista bat erabili dugu. Lista hau bukaeran sortzen den lexikoia frekuentziaren arabera ordenatzea da beren polaritatea eskuz zuzentzeko, horrela lexikoiaren kalitatea hobetzea lortuko dugu.

Polaritatea aldatzen duten hitzak

WordNET-ean azaltzen den lexikoiaz gain polaritatea nolabait aldatzen duten hitzak bildu behar dira bukaerako baliabide lexikoan, ez baita gauza bera hotel baten zerbitzua ona edo oso ona dela esatea. Horretarako hiru lista erabiliko ditugu:

- **Intensifiers**. Lista honetan polaritatea indartzen duten hitzak izango ditugu.
- **Weakeners**. Bertan polaritatea lehuntzen duten hitzak azaltzen dira.
- **PolarityShifters**. Hemen berriz polaritatea bihurtzen duten ditugu, hau da, lista honetako hitzekin polaritatea positiboa izatetik negatiboa izatera pasatzen da eta alderantziz negatiboekin.

HAP masterra

4.5.1.2.2 LMF fitxategia sortzeko pausuak

LMF fitxategia sortu ahal izateko dokumentu guztiak ditugula, baliabide lexikoaren sorkuntzarekin hasi gaitezke . Prozesua hainbat zatitan banatzen da: polaritate-listen zabalpena, zabaldutako listen hedapena WordNET-an zehar, lexikoaren zuzenketa, lexikoia LMF formatura pasatzea eta lexikoia osatzea. Hona hemen jarraitu beharreko pausu guztiak:

1. **Polaritate-listen zabalpena**. Egin beharreko lehenengo lana bi polaritate-listen bitartez WordNET-an zehar hedatuko den oinarrizko synset-lista lortzea izango da.
   (a) Lehendabizi polaritate-lista positibo eta negatiboak fitxategi bakarrean bilduko ditugu.
   (b) Fitxategia hitz bakoitzaren *synset* posibleekin osatuko dugu.
   (c) Bikoiztutako *synset*-ak ezabatuko ditugu listatik.
   (d) WordNET-aren laguntzaz lista ordenatuko dugu *synset* bakoitzak dituen erlazio-kopuruaren arabera.
   (e) Lehenengo 150 aditz, 200 adjektibo, 150 izen eta 50 adberbioen polaritateak eskuz zuzendu behar dira, eta beharreko kasuan polaritate neutralarekin etiketatuko ditugu.
   (f) Azkenik fitxategiaren formatua egokituko dugu:

   ```
   synset/polaritatea/categoria_gramatikala
   fre-30-001478-a/positive/a
   fre-30-001498-n/negative/n
   ```

2. **Listaren zabalpena WordNET-an zehar**. Hedapen algoritmo bati eske, aurreko pausuan lortutako lista WordNET-an zehar hedatuko dugu. Horretarako erlazio-dokumentua erabiliko dugu, bertan zehazten baita zein erlazio erabiliko ditugun hedapena egiteko. Algoritmoak informazio guztia csv fitxategi batean bilduko du. csv fitxategiak honako itxura dauka:

   ```
   synset;categoria;polaritatea;mesfidantza_zenb;lexikoa;-1
   fre-30-13962166-n;n;positive;0.5;endurance,survie;-1
   fre-30-07447261-n;n;positive;0.5;aventure,fois,fonction,fonctionner,
   occasion;-1
   fre-30-08516584-n;n;neutral;0.166666666667;boyaux,entrailles;-1
   fre-30-00730538-n;n;negative;0.166666666667;garde,garde-côte;-1
   ```

   Konfiantza zenbakia erlazio-dokumentuan erlazio bakoitzaren alboan dagoen zenbakiarekin kalkulatzen da. Polaritateak duen mesfidantza-maila adierazten du.

3. **Lexikoaren zuzenketa**. Orain *csv* fitxategiaren informazioa eskuz zuzenduko dugu.

(a) Zuzenketa egiteko frekuentzia-lista erabiliko dugu, lista honi esker *csv* fitxategia ordenatuko dugu frantsesez gehien erabiltzen direnetik gutxien erabiltzen direnera.

(b) Lehenengo 1000 *synset*-ak hartuko ditugu (hiztunek frekuentzia handiagoarekin erabiltzen dituztenak) eta polaritatea zuzenduko da.

(c) Zuzendutako *synset*-etan, *csv* fitxategian bakoitzari dagokion azken eremua, -1 zenbakia alegia, 1ean jarriko dugu. Honela synset-a eskuz zuzenduta dagoela adierazten da.

4. **Lexikoia LMF formatura pasa**. Script baten laguntzaz *csv* fitxategiaren informazioa LMF formatura pasako dugu. LMF fitxategiak honako itxura izango du:

```
<LexicalResource>
  <GlobalInformation label="Created with the standard
                            propagation algorithm"/>
  <Lexicon languageCoding="UTF-8" label="sentiment" language="-">
    <LexicalEntry id="id_107" partOfSpeech="adj">
      <Lemma writtenForm="cannibale"/>
      <Sense>
        <Confidence score="0.3125" method="automatic"/>
        <MonolingualExternalRefs>
          <MonolingualExternalRef externalReference="fre-30-03052770-a"/>
        </MonolingualExternalRefs>
        <Sentiment polarity="negative"/>
        <Domain/>
      </Sense>
    </LexicalEntry>
    <LexicalEntry id="id_153" partOfSpeech="adj">
      <Lemma writtenForm="huitième"/>
      <Sense>
        <Confidence score="0.416666666667" method="automatic"/>
        <MonolingualExternalRefs>
          <MonolingualExternalRef externalReference="fre-30-02203123-a"/>
        </MonolingualExternalRefs>
        <Sentiment polarity="positive"/>
        <Domain/>
      </Sense>
    </LexicalEntry>
    <LexicalEntry id="id_159" partOfSpeech="adj">
      <Lemma writtenForm="cool"/>
      <Sense>
        <Confidence score="0.416666666667" method="automatic"/>
        <MonolingualExternalRefs>
```

```
          <MonolingualExternalRef externalReference="fre-30-00971660-a"/>
        </MonolingualExternalRefs>
        <Sentiment polarity="positive"/>
        <Domain/>
      </Sense>
    </LexicalEntry>
    <LexicalEntry id="id_160" partOfSpeech="adj">
      <Lemma writtenForm="frais"/>
      <Sense>
        <Confidence score="0.416666666667" method="automatic"/>
        <MonolingualExternalRefs>
          <MonolingualExternalRef externalReference="fre-30-00971660-a"/>
        </MonolingualExternalRefs>
        <Sentiment polarity="positive"/>
        <Domain/>
      </Sense>
    </LexicalEntry>
        ⋮
  </Lexicon>
</LexicalResource>
```

5. **Lexikoia osatzea**. Azken lana LMF lexikoia polaritatea aldatzen duten hitz-listen informazioarekin osatzea da. Gogoratu hiru fitxategi ditugula: *intensifiers*, *weakeners* eta *polarityshifters*. Script sinple baten bidez honela geratuko litzateke aipatutako fitxategietako hitz bakoitzaren informazioa LMF formatuan:

```
<LexicalEntry id="id_25589" partOfSpeech="adv" type="weakener">
  <Lemma writtenForm="faiblement"/>
  <Sense>
    <Confidence method="manual" score="1"/>
    <MonolingualExternalRefs/>
    <Sentiment/>
    <Domain/>
  </Sense>
</LexicalEntry>
```

4.5.2 Ebaluazioa

18 eta 19. tauletan eraikitako baliabide-lexikoaren tamaina ikus dezakegu. Guztira 8.551 hitzetako lexikoia daukagu eta lexikoi positibo, negatibo eta neutralen arteko banaketa nahiko orekatua da. Kategoria gramatikalari erreparatuz, polaritate positibo, negatibo eta neutrala duten hitzak izenak, aditzak eta adjektiboak dira, hau da, esanahi lexikoa duten kategoria gramatikalak. Polaritatea nolabait aldatzen duten hitzak berriz bestelako

kategoria gramatikala daukate, batez ere adberbioak ditugu hauen artean. Bestalde, eskuz zuzenduta 1.210 hitzeko lexikoia dugu, guztira lortutako lexikoiarekin konparatzen badugu baliabide-lexikoaren %9,07-a eskuz zuzendu dugula ikusiko dugu, hau da dokumentuetan gehien azaltzea espero dugun lexikoia.

	positiboak	**negatiboak**	**neutral**	**intens./weaken.**	**shifters**	**GUZTIRA**
izenak	1.391	2.571	10	0	0	3.972
aditzak	1.321	480	19	0	0	1.820
adjektiboak	1.049	1.672	4.761	0	0	7.482
bestelakoak	0	0	3	52	15	70
GUZTIRA	3.761	4.723	4.793	52	15	13.344

Taula 18: Sentimendu baliabide-lexikoaren estatistikak

	hitzak
izenak	350
aditzak	350
adjektiboak	350
bestelakoak	60
GUZTIRA	1.210

Taula 19: Sentimendu baliabide-lexikoaren estatistikak: eskuz zuzendutako lexikoia

5 Emaitzak

Atal honetan zehar proiektuan inplementatutako web-zerbitzuen emaitzak ikusiko ditugu. Aurreko atalean azaldu dugu web-zerbitzuen barne dagoen modulu bakoitzaren ebaluazioa, horregatik oraingoan sistema bere osotasunean azaldu eta ebaluatuko dugu (5.1. eta 5.2. azpiatalak hurrenez-hurren). Proiektuaren deskribapenean esan dugu helburuetako bat proiektuko teknologia komunitatearen esku uztea dela, honi esker aplikazio ahaltsuak ahalik eta modu errazenean sortzea posible izango da. Azken helburu hau frogatzeko proiektu honetan zehar asmatutako aplikazio batzuk ikusiko ditugu (5.3. azpiatala).

5.1 Web-zerbitzuen azalpena

Proiektu honetako web-zerbitzuek arrakasta izan dezaten Cloud Computing teknologia erabiltzea erabaki da, honi esker konputazio-baliabideak (softwarea eta hardwarea) sare batean erabili ditzakegu zerbitzu baten bidez. Cloud Computing-aren abantailak honako hauek dira:

- Bizkortasuna.
- Kostuen murrizketa.
- Dispositibo eta kokapen independentzia.
- Birtualizazio teknologia.
- Eskalagarritasuna.
- Monitorizazioa.
- Segurtasuna.
- Cloud computing aplikazioen mantenu erraza.

Proiektu honek azpiegitura malgu bat eskatzen du eta cloud computing teknologiari esker ez dago hardware berririk erosi beharrik. Proiektu honetako web-zerbitzuek Amazon AWS[32] teknologia erabiltzen dute, gaur egun Amazon baitugu cloud computing teknologian nagusi.

Web-zerbitzuak erabiltzeko bi modu daude: moduluak elkarlanean, edo modulu independenteak.

5.1.1 Moduluak elkarlanean

Modulu guztiak exekuzio bakar baten bidez erabili nahi baditugu honako web-helbidera jo beharko dugu: `http://www.opener-project.eu/webservices/entrance.html`. Proiek-

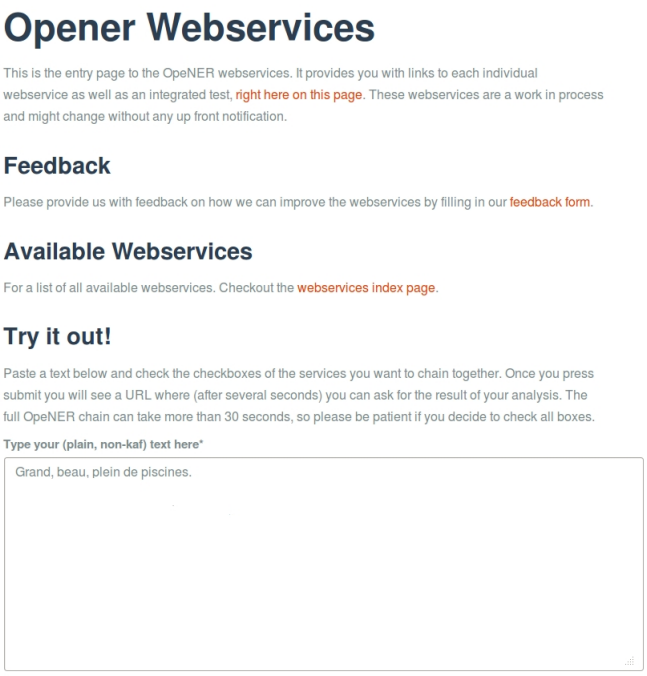
[32] `http://aws.amazon.com/es/`

tuaren deskribapenean (ikus 1.2. atala) azaldu dugu master bukaerako proiektu hau OpeNER proiektuaren barne kokatzen dela, horregatik, web-helbide horretan OpeNER-eko zerbitzu guztiak ikus daitezke. Proiektu honetan zehar egindako web-zerbitzuak honako hauek dira:

- language-identifier.
- tokenizer.
- pos-tagger (frantserako).
- ner (frantseserako).
- polarity-tagger (frantseserako).

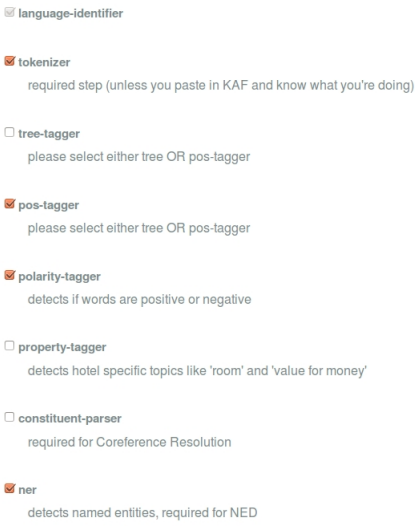
Web-zerbitzu honen erabilera oso erraza da, hona hemen jarraitu beharreko pausuak:

1. 4. irudian azaltzen den testu-kutxan analizatu beharreko testua idatzi.



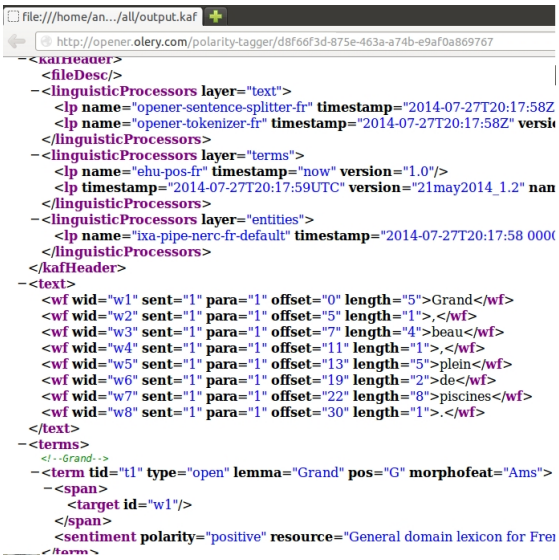
Irudia 4: Web-zerbitzua erabiltzeko idatzi goiko testu-kutxan testua.

2. Erabili nahi ditugun web-zerbitzuak markatu. 5. irudian ikus daiteke adibide bat.
3. *Submit* botoia sakatu.
4. *Submit* botoia sakatu eta gero JSON erantzun bat jasoko da (adibide honetan, {"request_id":"ca6e0e62-6f47-4399-98cd-51e6529a3c46","output_url":"http



Irudia 5: Aukeratu erabili nahi diren zerbitzuak.

://opener.olery.com/polarity-tagger/ca6e0e62-6f47-4399-98cd-51e6529a3c46}), bertan web-helbide bat azaltzen da non prozesamendu katea amaitu eta emaitza bertan jasoko den (ikus 6. irudia).



Irudia 6: Web-zerbitzuaren emaitza.

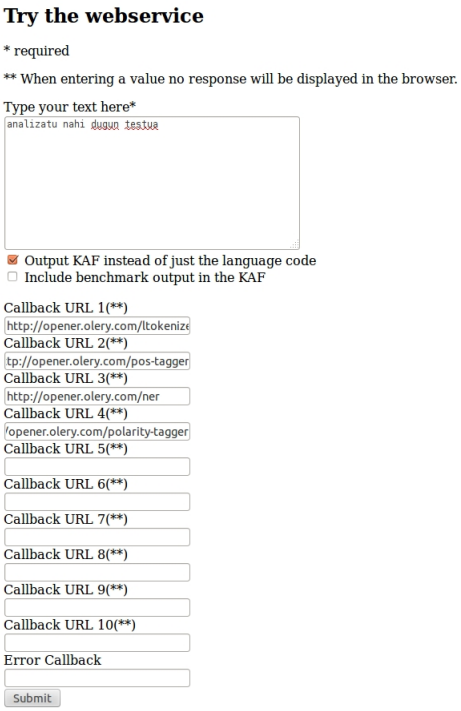
5.1.2 Modulu independenteak

Web-zerbitzu guztiak modu independentean exekutatzea posible da eta nahi izanez gero posible da komando bakar batekin modulu bakoitza web-zerbitzu bezala abiaraztea. Hala

HAP masterra

ere, aurrerago azalduko diren web-helbideen bidez atzitu ditzakegu web-zerbitzuak. Web-zerbitzu hauen abantaila handi bat `curl` komandoaren bidez atzigarriak direla da, esate baterako komando honi esker oso erraza izango litzateke script bat idaztea corpus bat prozesatzeko.

Web-zerbitzu guztien itxura oso antzekoa da, 7. irudian ikus dezakezue adibide bat. Web-zerbitzu bakoitzari dagokion atalean zehaztuko dira bakoitzaren berezitasunak. Besterik gabe ikus ditzagun web-zerbitzu guztiak.



Irudia 7: Web-zerbitzu baten adibidea.

5.1.2.1 Language Identifier

Deskribapena

Web-zerbitzu honek sarrera testuko hizkuntza detektatzeko balioko digu. Web-zerbitzuaren irteera hizkuntza kodea edo KAF dokumentua izan liteke.

Web-zerbitzua abiarazteko komandoa

```
$ language-identifier-server
```

Web-zerbitzuaren web-helbidea Amazon AWS zerbitzarian

```
http://opener.olery.com/language-identifier
```

HAP masterra

Argumentuak

Web-zerbitzuak honako argumentuak ditu:

*nahitaezko argumentuak

text*

Sarrera testua.

kaf [true|false]

Irteeran hizkuntza kodea edo KAF formatua nahi dugun adierazteko.

callbacks

Callback listan web-helbideak adieraziz gero Language Identifier zerbitzua atzeko planoan exekutatuko da eta emaitza callback listako lehen web-helbidera pasako da, gero callbacks listako web-helbide bakoitzak hurrengoari luzatuko dio emaitza. Hau oso erabilgarria da web-zerbitzuak elkarren artean kateatzeko, tokenizatzailearen web-helbidea callbacks web-helbide listan sartuz gero emaitzan hizkuntzaren identifikazioa eta tokenizazioa ikusiko genuke.

Web-zerbitzuaren atzipena komando-lerrotik

```
$ curl -d "input=analizatu nahi dugun testua&kaf=true" \
       http://opener.olery.com/language-identifier -XPOST
```

5.1.2.2 Tokenizer

Deskribapena

Sarrerako testua esalditan banatzeko eta tokenizatzeko web-zerbitzu hau erabiliko dugu. Sarrera testu hutsa ala KAF formatua izan liteke.

Web-zerbitzua abiarazteko komandoa

```
$ tokenizer-server
```

Web-zerbitzuaren web-helbidea Amazon AWS zerbitzarian

`http://opener.olery.com/tokenizer`

Argumentuak

Web-zerbitzuak honako argumentuak ditu:

*nahitaezko argumentuak

text*

Sarrera testua. Testu hutsa edo KAF formatua izan liteke `<raw>` geruzarekin.

HAP masterra

kaf [true|false]

Sarrera testu hutsa den ala KAF formatuan dagoen adierazteko argumentua.

callbacks

Callback listan web-helbideak adieraziz gero Tokenizer zerbitzua atzeko planoan exekutatuko da eta emaitza callback listako lehen web-helbidera pasako da, gero callbacks listako web-helbide bakoitzak hurrengoari luzatuko dio emaitza.

Web-zerbitzuaren atzipena komando-lerrotik

```
$ curl -d "input=<KAF xml:lang="fr"><raw>analizatu nahi dugun test
  ua</raw></KAF>&kaf=true" \
       http://opener.olery.com/tokenizer -XPOST
```

5.1.2.3 POS Tagger

Deskribapena

Zerbitzu honi esker sarrerako KAF dokumentuko token bakoitzari dagokion kategoria gramatikala ezarriko zaio.

Web-zerbitzua abiarazteko komandoa

```
$ pos-tagger-server
```

Web-zerbitzuaren web-helbidea Amazon AWS zerbitzarian

```
http://opener.olery.com/pos-tagger
```

Argumentuak

Web-zerbitzuak honako argumentuak ditu:

*nahitaezko argumentuak

text*

Sarrerako testua KAF formatuan `<text>` geruzarekin.

callbacks

Callback listan web-helbideak adieraziz gero POS Tagger zerbitzua atzeko planoan exekutatuko da eta emaitza callback listako lehen web-helbidera pasako da, gero callbacks listako web-helbide bakoitzak hurrengoari luzatuko dio emaitza.

Web-zerbitzuaren atzipena komando-lerrotik

```
$ curl -d "input=<KAF xml:lang="fr" version="v1.opener"><text><wf w
  id="w1" sent="1" para="1" offset="0" length="9">analizatu</wf><wf
  wid="w2" sent="1" para="1" offset="10" length="4">nahi</wf><wf wi
```

```
d="w3" sent="1" para="1" offset="15" length="5">dugun</wf><wf wid
="w4" sent="1" para="1" offset="21" length="6">testua</wf></text>
</KAF>" http://opener.olery.com/pos-tagger -XPOST
```

5.1.2.4 NER

Deskribapena

Izen-entitateak identifikatu eta sailkatzeko web-zerbitzua dugu, horretarako sarrerako testua KAF formatuan ipini beharko dugu token bakoitzak bere kategoria gramatikala duela.

Web-zerbitzua abiarazteko komandoa

```
$ ner-server
```

Web-zerbitzuaren web-helbidea Amazon AWS zerbitzarian

```
http://opener.olery.com/ner
```

Argumentuak

Web-zerbitzuak honako argumentuak ditu:

*nahitaezko argumentuak

text*

Sarrerako testua KAF formatuan `<text>` eta `<terms>` geruzarekin. Ez da guztiz beharrezko KAF dokumentuak `<terms>` geruza izatea, hala ere gomendagarria da horrela egitea, izan ere, baliteke etorkizunean kategoria gramatikalak edo lemak behar dituzten NERC prozesatzaile berriak inplementatzea.

callbacks

Callback listan web-helbideak adieraziz gero NER zerbitzua atzeko planoan exekutatuko da eta emaitza callback listako lehen web-helbidera pasako da, gero callbacks listako web-helbide bakoitzak hurrengoari luzatuko dio emaitza.

Web-zerbitzuaren atzipena komando-lerrotik

```
$ curl -d "input=<KAF xml:lang="fr" version="v1.opener"><text><wf w
  id="w1" sent="1" para="1" offset="0" length="9">analizatu</wf><wf
  wid="w2" sent="1" para="1" offset="10" length="4">nahi</wf><wf wi
  d="w3" sent="1" para="1" offset="15" length="5">dugun</wf><wf wid
  ="w4" sent="1" para="1" offset="21" length="6">testua</wf></text>
  <terms><term tid="t1" type="close" lemma="analizatu" pos="V"><spa
  n><target id="w1"/></span></term><term tid="t2" type="close" lemm
  a="nahi" pos="V"><span><target id="w2"/></span></term><term tid="
```

```
t3" type="close" lemma="dugun" pos="V"><span><target id="w3"/></s
pan></term><term tid="t4" type="close" lemma="testua" pos="N"><sp
an><target id="w4"/></span></term></terms></KAF>" \
     http://opener.olery.com/ner -XPOST
```

5.1.2.5 Polarity Tagger

Deskribapena

KAF dokumentu batean termino bakoitzari dagokion polaritatea (positiboa, negatiboa ala neutrala) ezartzeko balioko digu.

Web-zerbitzua abiarazteko komandoa

```
$ polarity-tagger-server
```

Web-zerbitzuaren web-helbidea Amazon AWS zerbitzarian

```
http://opener.olery.com/polarity-tagger
```

Argumentuak

Web-zerbitzuak honako argumentuak ditu:

*nahitaezko argumentuak

text*

Sarrerako testua KAF formatuan `<text>` eta `<terms>` geruzarekin.

callbacks

Callback listan web-helbideak adieraziz gero Polarity Tagger zerbitzua atzeko planoan exekutatuko da eta emaitza callback listako lehen web-helbidera pasako da, gero callbacks listako web-helbide bakoitzak hurrengoari luzatuko dio emaitza.

Web-zerbitzuaren atzipena komando-lerrotik

```
$ curl -d "input=<KAF xml:lang="fr" version="v1.opener"><text><wf w
  id="w1" sent="1" para="1" offset="0" length="9">analizatu</wf><wf
  wid="w2" sent="1" para="1" offset="10" length="4">nahi</wf><wf wi
  d="w3" sent="1" para="1" offset="15" length="5">dugun</wf><wf wid
  ="w4" sent="1" para="1" offset="21" length="6">testua</wf></text>
  <terms><term tid="t1" type="close" lemma="analizatu" pos="V"><spa
  n><target id="w1"/></span></term><term tid="t2" type="close" lemm
  a="nahi" pos="V"><span><target id="w2"/></span></term><term tid="
  t3" type="close" lemma="dugun" pos="V"><span><target id="w3"/></s
  pan></term><term tid="t4" type="close" lemma="testua" pos="N"><sp
```

```
an><target id="w4"/></span></term></terms></KAF>" \
     http://opener.olery.com/polarity-tagger -XPOST
```

5.2 Web-zerbitzu osoaren ebaluazioa

Web-zerbitzu bakoitzeko moduluak ebaluatu ditugu 4. atalean, hemen berriz web-zerbitzu guztiak bere osotasunean ebaluatuko ditugu. Lehendabizi ebaluazio corpusari buruz mintzatuko gara eta gero corpusa analizatzeko abiadura eta sortutako etiketei buruz arituko gara.

5.2.1 Ebaluaziorako corpusa

Proiektu honen aplikazio-domeinua turismoa izanik, ebaluaziorako corpus egoki batek erabiltzaileen turismo-zerbitzuei buruzko iritziak bildu beharko lituzke. OpeNER proiektuaren baitan hainbat sare-sozial eta turismorako web-zerbitzutik erabiltzaileen iritziak jaso dira Europako hainbat hizkuntzetarako, hauen artean frantsesa noski. Hauek dira erabili diren sare-sozial eta web-zerbitzuak:

- **Foursquare**[33]. Foursquare kokapenean oinarritutako web-zerbitzu bat da batez ere gailu mugikorretarako pentsatuta dagoena. Erabiltzaileei modu intuitibo batean gertu dituen hotelak edo jatetxeak bezalako zerbitzuak aurkezten zaizkio. Erabiltzaileek zerbitzuei buruzko iruzkinak idatzi ditzakete eta iruzkin horiek baliagarriak izaten dira beste erabiltzaileetarako.

- **Google Places**. Google Places-ek negozioak erakusten ditu Google Maps-en bitartez. Oso aplikazio baliagarria da bezeroak erakartzeko eta negozioak jakinarazteko. Negozio bakoitzak bere *Google Place*-a dauka, bertan produktu edota zerbitzuei buruzko informazioa idatzi daiteke eta bezeroek beren iritziak ematen dituzte.

- **Facebook**[34]. Facebook sare-sozialari esker erabiltzaileek beren perfilak eta hainbat motatako entitateak kudeatu ditzakete: 1) negozioentzako orriak; 2) zerbaiti buruz eztabaidatzeko taldeak; 3) edozein motatako gertaerak (kontzertuak, lasterketak, etab.); eta 4) leku geografikoak. Argi dago beraz Facebook oso informazio-iturri baliagarria dela.

Gorde beharreko informazioa handia, egituratua eta heterogeneoa da. Datuak nolabait bildu eta sailkatzeko kokapen-geografikoak erabili dira. OpeNER-erako aukeratutakoak honako hauek dira: Amsterdam (Holanda), Tuscani (Italia) eta Espainia.

20. taulan bildutako corpus tamaina ikus daiteke.

[33] `https://es.foursquare.com/`
[34] `https://es-es.facebook.com/`

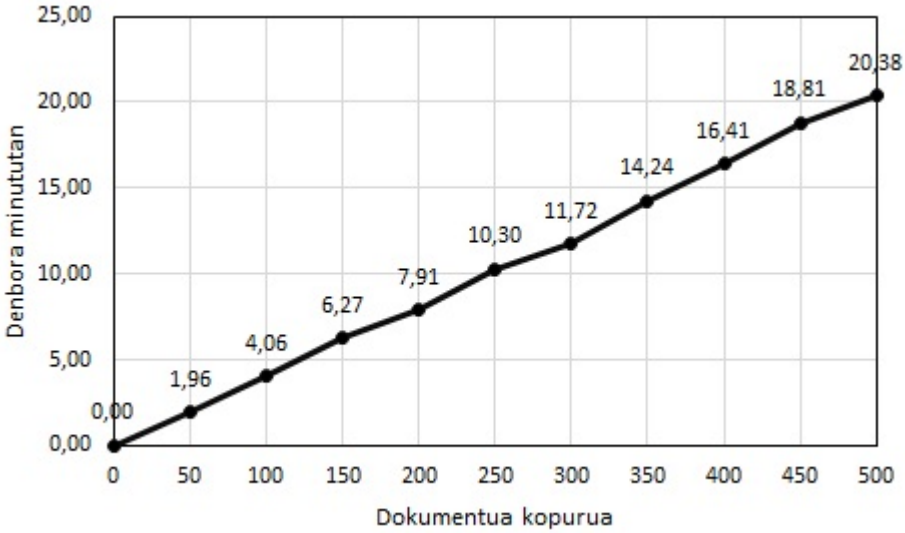
Foursquare, Google Places eta Facebook corpusa	
Dokumentu kop.	521
Esaldi kop. guztira	1.757
Esaldi kop. dokumentuko	3
Hitz kop. guztira	41.272
Hitz kop. dokumentuko	79

Taula 20: Ebaluaziorako lortutako frantseserako corpusa.

5.2.2 Ebaluazioa

Ebaluaziorako perl-ez script bat idatzi da dokumentu bat irakurri eta testua web-zerbitzuetara bidaltzeko, hori egin ahal izateko `curl` komandoa erabiltzen da. Scriptari buruzko xehetasunak nahi izanez gero 7. atalera jo daiteke.

8. irudian dokumentuak prozesatzeko abiadura ikus daiteke. Taulatik ondorioztatu dezakegu prozesamenduak nolabaiteko proportzionaltasun bat mantentzen dela, hau da, dokumentuak analizatu ahala beti oso prozesamendu denbora antzekoak lortzen dira. Web-zerbitzuaren beste abantaila bat bere azkartasuna da, dokumentu bakoitzeko prozesamenduak ia 2 segundo eta erdi irauten baitu. Jakinda LNP prozesuez gain TCP/IP komunikazioak daudela azpian nahiko prozesamendu denbora onak lortzen dira.



Irudia 8: Ebaluazioa: dokumentuak prozesatzeko abiadura.

21. taulan berriz etiketatuko izen-entitateak aurkezten dira. Orokorrean izen-entitate gutxi aurkitzen dira, jendeak iruzkin labur bat idatzi behar duenean ez da ohikoa zerbitzuen izenak idaztea, honen adibide beheko iruzkina dugu:

Grand, beau, plein de piscines. Cet hotel est à conseiller pour ses piscines et les jeux

aquatiques. Très bel hotel mais la nourriture est le point faible même si on peut manger correctement mais simplement tous les jours. ok

Izen-entitate motak aztertzen baditugu ikus dezakegu orokorrean 3 mota gailentzen direla: pertsonak, lekuak eta erakundeak. Guztiz ulergarria da, turismo-zerbitzu bati idatzi behar badugu izen-entitate mota hauei buruz hitz egingo dugu eta.

Izen-Entitate Motak	Kopurua
date	54
location	125
misc	7
money	1
organization	80
person	84
time	2

Taula 21: Ebaluazioa: aurkitutako izen-entitateak.

Etiketatutako-polaritateak 22. taulan aztertzen dira. Taulari erreparatuz ikus dezakegu polaritatea duten terminoak askoz ere gehiago erabiltzen direla izen-entitateak baino. Nahiko deigarria da polaritate positibo eta negatiboen aldea, badirudi erabiltzaileak normalean pozik daudela turismo-zerbitzuekin. Taulatik ondorioztatu dezakegu iruzkinetan polaritate indartzaileak asko erabiltzen direla, honek beraien iritziei indar gehiago ematen die.

Polaritate-mota	Kopurua
positiboa	2.735
neutrala	2.546
negatiboa	1.229
polaritate indartzaileak	1.219
polaritate trukatzaileak	657

Taula 22: Ebaluazioa: aurkitutako polaritate etiketak.

5.3 Sortutako bestelako aplikazioak

Proiektu honek etorkizuna baduela frogatzeko inplementatutako web-zerbitzuak erabiltzen dituzten aplikazio batzuk ikusiko ditugu. Aplikazio hauek OpeNER proiektuaren baitan sortu dira demo antzera web-zerbitzuak probatzeko, erabilerrazak diren ala ez ikusteko eta benetan aplikazio politak sortu daitezkeela erakusteko.

Garatutako aplikazioak honako hauek dira:

HAP masterra

- **TourPedia**[35]. Web-aplikazio honetan Europako hiriburu garrantzitsuenetan erabiltzaileek turismo-zerbitzuei buruzko sentimena agertzen da mundu-mapa batean. Hiriburu bakoitzean klikatuz mapa handitu dezakegu eta sentimenari buruzko xehetasunak ikus ditzakegu. Sentimenak polaritatearen arabera koloreztatutako borobiletan biltzen dira, eta hauetan klikatuz gero mapa gehiago handituko da eta turismo-zerbitzu bakoitzaren kokalekuan ikusiko ditugu polaritate-borobil gehiago. Azken polaritate-borobiletan klikatuz irakurri ditzakegu iruzkin guztiak.
- **EntitUp**[36]. Web-aplikazio honek analizatutako dokumentuetan agertzen diren izen-entitateen inguruko analisia egiten du. Denboran zehar izen-entitate bakoitzari buruz gehiago edo gutxiago hitz egin den aztertu dezakegu grafiko batzuei esker.
- **Moodmap**[37]. LREC 2014 konferentzian twitter-en idatzitako mezuak aztertzen dira. Grafiko batean ikus dezakegu zeintzuk diren jendeari gehien gustatu zaizkion gaiak.
- **KAF Browser**[38]. OpeNER-eko web-zerbitzuekin prozesatutako KAF dokumentuetan dagoen informazioa grafikoki ikusteko balio digun web-aplikazioa da. Nahi dugun testua prozesatu dezakegu eta bertan koloreztatuta ikus ditzakegu izen-entitateak eta sentimena. Detektaturiko izen-entitateak DBpedia-rekin[39] lotzen dira esteken bidez, eta ez hori bakarrik, lekuzko izen-entitateak agertzen badira mundu-mapa batean etiketatzen dira.

[35] http://tour-pedia.org/gui/demo/
[36] http://128.65.123.12/entitup/social.php
[37] http://tour-pedia.org/moodmap/
[38] http://demo2-opener.rhcloud.com/welcome.action
[39] http://es.dbpedia.org/

6 Ondorioak eta etorkizuneko lanak

Proiektu honetan zehar atera daitekeen ondorio garbi bat hizkuntza-teknologia oso baliagarria dela da. Internet eta sare-sozialen hazkundearekin ia konturatu gabe corpus erraldoi bat sortu da sarean etengabe hazten dena. Corpus horren lagin txiki bat hartuz, hau da turismoaren domeinuarekin zerikusia duena, eta proiektuko tresnak aplikatuz gizartearen behar konkretu bati erantzuna eman diogu (turismo-zerbitzuak automatikoki ebaluatzea).

Proiektu honetan ikusi dugu formatu estandarrak erabiltzea abantaila ona dela, horrela egin izan ez balitz askoz zailagoa izango litzateke proiektuko prozesatzaileak inplementatzea, eta ez hori bakarrik, etorkizunari begira zailagoa izango litzateke moduluen hobekuntza eta modulu berrien integrazioa. Formatu estandarren artean KAF formatua dugu, formatu honek erakutsi digu edozein motako LNP informazioa gordetzeko oso baliagarria dela, baina formatua bera ez da batere atsegina erabiltzaileentzat. KAF formatuan gordetzen den informazio erakusteko modu ezberdinekin esperimentuak egin daitezke erabiltzaileentzat aplikazio erakargarriagoak egiteko.

Proiektu honen ideia nagusia luzaroan erabiltzea espero den hizkuntza-teknologia garatzea da. Helburu hori lortu ahal izateko oso lagungarria da garatutako tresnak komunitateari erakustea eta ahalik eta kode gehien haien esku uztea. Txosten honetan zehar aipatu dugun bezala proiektu hau OpeNER-en parte da, eta bertan, komunitatearen sorrera bermatzeko hainbat lehiaketa antolatu dira Amsterdamen[40] eta Islandian[41] garatutako tresna guztiak ezagutzera emateko. Horretaz gainera proiektu honen modulu guztien kodea github-en dago edozeinen eskura[42].

Proiektuaren etorkizuna bermatzeko ezinbestekoa da denboran zehar hobekuntzak egitea, bestela tresna hobeak azalduko dira eta proiektuko teknologia ez da erabiliko. Ildo horretatik jarraituz proiektuko moduluetan hobekuntzak egin beharko dira eta teknologia berriak integratzen dituzten moduluak garatu beharko dira teknologia konpetitiboa izan dadin.

Turismoa da proiekturako aukeratu den aplikazio-domeinua eta nahiko aplikazio interesgarriak sortu dira proiektuko web-zerbitzuak erabiliz (ikus 5.3. atala). Era berean oso interesgarria izango litzateke egindako lana beste aplikazio-domeinuetara egokitzea. Beste domeinuetan emaitza onenak lortu ahal izateko komenigarria izango litzateke moduluetan zenbait egokitzapen egitea (corpus bereziak erabiliz modelo berriak entrenatu, modulu bereziak sortzea, lexikoia egokitzeko mekanismoak garatzea, etab.).

Proiektu honetan frantseserako moduluak landu dira eta OpeNER-en barne 5 hizkuntza gehiagorekin lan egin da (gaztelania, italiera, nederlandera, alemana eta ingelesa). Etorkizunean hizkuntza gehiagorekin aproba egitea komenigarria da hauen integrazioa benetan

[40] `http://www.opener-project.eu/2013/07/18/Opener-hackathon-in-amsterdam-1-2-july-2013.html`

[41] `http://www.opener-project.eu/2014/05/28/the-opener-hackathon-has-been-a-success.html`

[42] `https://github.com/opener-project`

nolakoa izango litzatekeen jakiteko. Gainera, integratutako hizkuntza guztiekin esperimentuak eginez hizkuntzen arteko ezberdintasunak analizatu genitzake, baliteke hizkuntzen arteko konparaketaren bitartez ondorio interesgarriak ateratzea (gerta liteke hizkuntza bateko hizlariek espresio positibo gehiago erabiltzea, turismoari buruz hitz egitean garrantzia gehiago ematea zerbitzu batzuei, bidaia gehiago egitea leku batzuetara, etab.).

7 Eranskinak

Behean ikusten den bezala oso erraza da dokumentu bat irakurri eta testua proiektu honetako web-zerbitzuetara bidaltzen duen script bat idaztea. Beheko scripta perl-ez idatzita dago:

```
#########################################################
#  $ perl opener-pipeline.pl input_file output_file  #
#  processes input_file through the opener pipeline  #
#########################################################
use strict;
use encoding 'utf8';

my $LANGUAGE_IDENTIFIER_SERVICE = "http://opener.olery.com/language-identifier";
my $TOKENIZER_SERVICE           = "http://opener.olery.com/tokenizer";
my $POSTAGGER_SERVICE           = "http://opener.olery.com/pos-tagger";
my $NERC_SERVICE                = "http://opener.olery.com/ner";
my $POLARITY_TAGGER_SERVICE     = "http://opener.olery.com/polarity-tagger";

if (scalar(@ARGV) < 2) {
  print "Usage: perl opener-pipeline.pl input_file output_file\n";
  print "processes input_file through the opener pipeline.\n";
  exit;
}

my $text = &readfile($ARGV[0]);
$text    = &language_identifier($text);
$text    = &tokenizer($text);
$text    = &postagger($text);
$text    = &nerc($text);
$text    = &polarity_tagger($text);
&writefile($text, $ARGV[1]);

######### METHODS FOR WEB-SERVICES HANDLING ##########
sub language_identifier {
  my $text = shift(@_);
  my $result = &curl($text, $LANGUAGE_IDENTIFIER_SERVICE, "&kaf=true");
  return $result;
}

sub tokenizer {
  my $text = shift(@_);
  my $result = &curl($text, $TOKENIZER_SERVICE, "&kaf=true");
```

```
  return $result;
}

sub postagger {
  my $text = shift(@_);
  my $result = &curl($text, $POSTAGGER_SERVICE, "");
  return $result;
}

sub nerc {
  my $text = shift(@_);
  my $result = &curl($text, $NERC_SERVICE, "");
  return $result;
}

sub polarity_tagger {
  my $text = shift(@_);
  my $result = &curl($text, $POLARITY_TAGGER_SERVICE, "");
  return $result;
}

#################### CURL COMMAND ####################
sub curl {
  my $text = shift(@_);
  my $web_service = shift(@_);
  my $options = shift(@_);

  $text =~ s/\"/\\"/g;
  $text =~ s/\\+/\\/g;
  my $result = `curl -d "input=$text$options" $web_service -XPOST 2>/dev/null`;
  $result =~ s/\"/\\"/g;
  $result =~ s/\\+/\\/g;

  return $result;
}

############## FILE READ / WRITE METHODS #############
sub readfile {
  my $file = shift(@_);
  open (IN, $file) || die $!;
  binmode(IN, ":utf8");

  my $text = "";
```

```
  while (my $line = <IN>) {
    $text .= $line;
  }
  close(IN);

  return $text;
}

sub writefile {
  my $text = shift(@_);
  my $outfile = shift(@_);

  $text =~ s/\\\"/\"/g;
  open (OUT, ">".$outfile) || die $!;
  binmode(OUT, ":utf8");
  print OUT "$text\n";
  close(OUT);
}
```

Erreferentziak

Anne Abeillé, Lionel Clément, eta François Toussenel. Building a treebank for french. In *Treebanks*, pages 165–187. Springer, 2003.

Eneko Agirre, Xabier Artola, Arantza Diaz de Ilarraza, German Rigau, Aitor Soroa, eta Wauter Bosma. Kaf: Kyoto annotation framework. Technical report, Technical Report TR 1-2009, Dept. Computer Science and Artificial Intelligence, University of the Basque Country, 2009.

C. Aone, L. Halverson, T. Hampton, eta M. Ramos-Santacruz. SRA: Description of the IE2 system used for MUC-7. In *Proceedings of MUC-7*, 1998.

Douglas E. Appelt, Jerry R. Hobbs, John Bear, David Israel, Megumi Kameyama, Andy Kehler, David Martín, Karen Myers, eta Mabry Tyson. International fastus system MUC-6 test results and analysis. Proceedings of the 6th Messsage Understanding Conference, pages 237–248. Morgan Kaufmann Publishers, Inc., Columbia, Maryland, 1995.

Jordi Atserias, Marieke van Erp, Isa Maks, German Rigau, eta J Fernando Sánchez-Rada. Eurolovemap: Confronting feelings from news. In *Come Hack with OpeNER! Workshop Programme*, page 5.

Andoni Azpeitia, Alexandra Balahur, Montse Cuadros, Antske Fokkens, eta Ruben Izquierdo. The snowball effect: following opinions on controversial topics. In *Come Hack with OpeNER! Workshop Programme*, page 15.

Andoni Azpeitia, Montse Cuadros, German Rigau, eta Sean Gaines. Nerc-fr: Supervised named entity recognition for french. In *Proceedings of 17th International Conference on Text, Speech and Dialogue (TSD2014)*, page to appear, September 2014.

Jason Baldridge, Thomas Morton, eta Gann Bierner. The opennlp maximum entropy package. Technical report, Technical report, Technical report, SourceForge, 2002.

Daniel M. Bikel, Scott Miller, Richard Schwartz, eta Ralph Weischedel. Nymble: A high performance learning name-finder. In *Proceedings of the 5th Conference on Applied Natural Language Processing, ANLP*, Washington DC, 1997.

A. Borthwick. A maximum entropy approach to named entity recognition. In *Ph.D. thesis, New York University*, 1999.

A. Borthwick, J. Sterling, E. Agichtein, eta R. Grishman. Description of the named entity system as used in MUC-7. In *Proceedings of the 7th Message Understanding Conference*, 1998.

Indra Budi eta Stéphane Bressan. Association rules mining for name entity recognition. 2003.

HAP masterra

Pimwadee Chaovalit eta Lina Zhou. Movie review mining: A comparison between supervised and unsupervised classification approaches. In *Proceedings of the Hawaii International Conference on System Sciences (HICSS)*, 2005.

M. Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the EMNLP'02*, 2002.

Michael Collins eta Yoram Singer. Unsupervised models for named entity classification. In *Proceedings of the Joint SIGDAT Conference on EMNLP*, 1999. URL `citeseer.nj.nec.com/collins99unsupervised.html`.

Stefano Cresci, Andrea D'Errico, Davide Gazzé, Angelica Lo Duca, Andrea Marchetti, eta Maurizio Tesconi. Tour-pedia: a web application for sentiment visualization in tourism domain. In *Come Hack with OpeNER! Workshop Programme*, page 18.

Alessandro Cucchiarelli eta P. Velardi. Unsupervised named entity recognition using syntactic and semantic contextual evidence. Computational Linguistics 27:1.123-131., Cambridge: MIT Press, 2001.

S. Cucerzanand eta D. Yarowsky. Language independent named entity recognition combining morphological and contextual evidence. In *Proceedings of the Joint SIGDAT Conference on EMNLP and VLC*, 1999.

Aniket Dalal, Kumar Nagaraj, Uma Sawant, eta Sandeep Shelke. Hindi part-of-speech tagging and chunking: A maximum entropy approach. *Proceeding of the NLPAI Machine Learning Competition*, 2006.

John Darroch eta Douglas Ratcliff. Generalized iterative scaling for log-linear models. *The annals of mathematical statistics*, pages 1470–1480, 1972.

Asif Ekbal eta Sivaji Bandyopadhyay. Named entity recognition using support vector machine: A language independent approach. *International Journal of Computer Systems Science & Engineering*, 4(2), 2008.

Andrea Esuli eta Fabrizio Sebastiani. Sentiwordnet: A publicly available lexical resource for opinion mining. In *Proceedings of LREC*, volume 6, pages 417–422, 2006.

Sylvain Galliano, Edouard Geoffrois, Djamel Mostefa, Khalid Choukri, Jean-François Bonastre, eta Guillaume Gravier. The ester phase ii evaluation campaign for the rich transcription of french broadcast news. In *Interspeech*, pages 1149–1152, 2005.

Aitor García-Pablos, Montse Cuadros, Seán Gaines, eta German Rigau. Opener demo: Open polarity enhanced named entity recognition. In *Come Hack with OpeNER! Workshop Programme*, volume 501, page 12.

Aitor Gonzalez-Agirre, Egoitz Laparra, eta German Rigau. Multilingual central repository version 3.0. In *LREC*, pages 2525–2529, 2012.

HAP masterra

Gregory Grefenstette eta Pasi Tapanainen. What is a word, what is a sentence? problems of tokenization. In *Proceedings of the 3rd International Conference on Computational Lexicography*, page 79–87, 1994.

Claire Grover, Michael Matthews, eta Richard Tobin. Tools to address the interdependence between tokenisation and standoff annotation. In *Proceedings of the 5th EACL-2006 Workshop on NLP and XML (NLPXML-2006): Multi-Dimensional Markup in Natural Language Processing*, page 19–26. Association for Computational Linguistics (ACL), 2006.

Jan Hajič. Morphological tagging: Data vs. dictionaries. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 94–101. Association for Computational Linguistics, 2000.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 177–180. Association for Computational Linguistics, 2007.

George R. Krupka eta Kevin Hausman. IsoQuest, inc.: Description of the NetOwl$^{\text{TM\$}}$ extractor system as used for MUC-7. In *Proceedings of the 7th Message Understanding Conference*, 1998.

Eric Laporte, Takuya Nakamura, Stavroula Voyatzi, et al. A french corpus annotated for multiword nouns. In *Proceedings of the Language Resources and Evaluation Conference. Workshop Towards a Shared Task on Multiword Expressions*, pages 27–30, 2008.

Isa Maks, Ruben Izquierdo, Francesca Frontini, Rodrigo Agerri, Piek Vossen, eta Andoni Azpeitia. Generating polarity lexicons with wordnet propagation in 5 languages. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, eta Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland, may 2014. European Language Resources Association (ELRA). ISBN 978-2-9517408-8-4.

Christopher D Manning eta Hinrich Schütze. *Foundations of statistical natural language processing*. MIT press, 1999.

A. Mikheev, M. Moens, eta C. Grover. Named entity recognition without gazetteers. In *Proceedings of the 9th EACL*, pages 1–8, 1999.

Andrei Mikheev, Claire Grover, eta Marc Moens. Description of the system used for MUC-7. In *Proceedings of the 7th Message Understanding Conference*, 1998.

HAP masterra

George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

David D. Palmer eta Marti A. Hearst. Adaptive multilingual sentence boundary disambiguation. In *Computational Linguistics*, page 23(2):241–267, 1997.

S. Della Pietra, V. Della Pietra, eta J. Lafferty. Inducing features of random felds. IEEE Trans. Pattern Analysis and Machine Intell., 19(4), April 1997.

Sara Pupi, Giulia Di Pietro, eta Carlo Aliprandi. Ent-it-up. In *HCI International 2014-Posters' Extended Abstracts*, pages 3–8. Springer, 2014.

Adwait Ratnaparkhi et al. A maximum entropy model for part-of-speech tagging. In *Proceedings of the conference on empirical methods in natural language processing*, volume 1, pages 133–142. Philadelphia, PA, 1996.

E. Riloff eta R. Jones. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, 1999.

Benoît Sagot, Darja Fišer, et al. Building a free french wordnet from multilingual resources. In *OntoLex*, 2008.

Satoshi Sekine. Nyu: Description of the japanese NE system used for met-2. In *Proc. Message Understanding Conference*, 1998.

Nakatani Shuyo. Language detection library for java, 2010. URL `http://code.google.com/p/language-detection/`.

Anders Søgaard. Simple semi-supervised training of part-of-speech taggers. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 205–208. Association for Computational Linguistics, 2010.

Pedro A Torres-Carrasquillo, Elliot Singer, Mary A Kohler, Richard J Greene, Douglas A Reynolds, eta John R Deller Jr. Approaches to language identification using gaussian mixture models and shifted delta cepstral features. In *INTERSPEECH*, 2002.

Yoshimasa Tsuruoka, Yuka Tateishi, Jin-Dong Kim, Tomoko Ohta, John McNaught, Sophia Ananiadou, eta Jun'ichi Tsujii. Developing a robust part-of-speech tagger for biomedical text. In *Advances in informatics*, pages 382–392. Springer, 2005.

Piek Vossen et al. Eurowordnet: a multilingual database for information retrieval. In *Proceedings of the DELOS workshop on Cross-language Information Retrieval*, pages 5–7, 1997.

Ralph Weischedel. Description of the plum system as used for MUC-6. Proceedings of the 6th Messsage Understanding Conference, pages 55–69. Morgan Kaufmann Publishers, Inc., Columbia, Maryland, 1995.

HAP masterra