Itzulpen automatikorako tresnen egokitzapena euskararako: post-edizioa, ebaluazioa eta aurre-edizioa

Karrera-bukaerako proiektua
Uztaila, 2012

Unai Cabezón

Gainbegiralea:
Kepa Sarasola

Eskertza

IXA taldeko lankideei, batez ere proiektuaren zuzendari *Kepa Sarasola*ri, *Iñaki Alegria*ri, eta baita *Gorka Labaka*ri, proiektuan zehar eskainitako laguntza guztia dela eta.

*Euskal Wikipedia*tik lagundu gaituen jendeari, gehien bat *Unai Fernández de Betoño*ri; eta *OpenMT2* proiektuan parte hartu eta Matxin eta OmegaT erabiliz artikuluen itzulpenak egin eta Euskal Wikipediara igo dituzuen guztiei egindako saio desberdinetan.

OmegaT aplikazioaren garapen talde zabal eta aktiboari, beren mezu sarriekin aldaketei buruz informatzen eta galderak egiten eta erantzuten dabiltzanei. Baita *Asiya* eta *DiSeg* tresnen egile eta mantentzaileei, beren laguntza oso beharrezkoa gertatu ez den arren, sarean eskuragarri egoteagatik.

Fakultateko nire *klasekideei*, hasieratik egon direnei, geroago ezagutu garenoi eta bidean gelditu zinetenei.

Eta azkenik, baina garrantzitsuenak, nire *familiari*, batez ere *aita*, *ama* eta *Estitxu*ri, garai hauetan zehar euskarri izateagatik eta beti laguntzeko prest egoteagatik.

Laburpena

Proiektu honetan zehar Itzulpen Automatikoa eta horren inguruko tresnen inguruan jorratu da. Lengoaia Naturalaren Prozesamendua eta itzulpen automatikoa ikasi eta aztertu egin dira ikuspuntu zabal batetik. Itzulpen automatiko orokorraz eta horren aplikazio mota desberdinetatik gain, bestelako kontzeptuak ere tratatu dira, hala nola, itzulpenean laguntzeko tresnak, itzulpen automatikoaren ebaluazioa eta itzulpen automatikorako testuen aurre-edizioa eta post-edizioa.

Ikasketa- eta aztertze-prozesu horretaz gain, erlazionatuta dauden tresnak erabili edota moldatu egin dira euskararako itzulpen automatikoan barne. Hiru atal nagusi nabarmendu daitezke:

Lehenengo, OmegaT, itzulpenean laguntzeko softwarea, moldatu da Matxin euskararako itzultzaile automatikoa gehituz. Gainera, IXA Taldearen eta Euskal Wikipediaren arteko kolaborazio-lanean, Wikipediako artikuluak eskuratu, itzuli eta igotzeko aukera egokitu zaio OmegaT-ri eta horren erabilera sustatu da Euskal Wikipediako komunitatean eta UPV/EHUko Informatikako ikasle eta irakaslegoaren artean. Bestalde, lan honetaz baliatuz, OmegaT-k sortzen dituen itzulpen-memoriak, Matxin-en itzulpenen

gaineko post-edizioan oinarrituak, eskuratzeko modu bat egin da, horiekin Matxin-en funtzionamendua hobetu ahal izateko.

Ondoren, Asiya programan integratu egin da euskara. Asiya-k itzulpen automatikoaren ebaluazio eta meta-ebaluazioak egin ditzakeen aplikazioa da. Hainbat metrika aztertu dira euskara aztertzeko balio ote duten begiratzeko. Besteen artean, lau metrikari euskara gehitzeko saiakera egin nahi izan da IXA Taldeko euskarazko testuen analizatzaile batek eskainitako informazio sintaktikoa gehituz, baina bi metrika soilik egokitu ahal izan dira.

Azkenik, DiSeg esaldi-segmentatzailea erabili egin da gaztelerazko corpus baten gainean esaldi luzeak banatzeko. Aurre-edizio hori eta gero itzuli egin dira eta Asiya erabiliz emaitzen ebaluazioa eta konparazioa egin dira, esaldi laburragoekin itzulpen automatiko eraginkorragoa lortzen oten den aztertzeko.

Hitz gakoak:

Itzulpen Automatikoa
Lengoaia Naturalaren Prozesamendua
Testuen aurre-edizioa
Testuen post-edizioa
Itzulpen Automatikoaren ebaluazioa
Itzulpenerako laguntzazko tresnak
Matxin
OpenMT2 proiektua eta Wikipedia
OmegaT
Asiya
DiSeg

Aurkibidea

Irudien, taulen eta kodeen aurkibidea

OMEGAT-REN EGOKITZAPENA

ASIYA-REN EGOKITZAPENA

DISEG-EKIN ESPERIMENTUAK

1. Proiektuaren Helburu-Dokumentua

1.1 Proiektuaren irismena

Proiektu honen helburu nagusitzat Lengoaia Naturalaren Prozesamenduaren (LNP) arloaren barnean itzulpen automatikoaren (MT, *machine translation*) munduan murgiltzea da, erreminta desberdinen erabilerak aztertuz eta horien kodeak aldatzen euskarazko funtzioak izan ditzaten eta euskararekin erlazioa duten proiektuen parte beazala. Itzulpen automatikoaren barnean, itzulpenerako laguntzazko programak, itzulpen automatikoaren ebaluazioa eta aurre-edizioa ikusi egingo dira. Lan hau UPV/EHUko Donostiako Informatika Fakultateko IXA taldearekin[1] batera egin dut, bertako bekadun bezala hasieran, eta karrera-bukaerako proiektu formal gisa azken lauhilekoan zehar, Eusko Jaurlaritzako bekadun gisa.

Berez, proiektuan zehar aurretik IXAko bekakide bezala egindako lana ere hartu egin da kontuan, hori zabaldu baita proiektua egikaritze bitartean, eta oinarri bezala hartu baita hainbat zentzutan. Azken finean bi lanak oso erlazio estua dutenez, honela egitea erabaki egin da. Hala izanik, proiektuaren zati bat 2010 eta 2011 urteetan egin zen, bederatzi hilabetetik gorako bi zatitan, eta beste zatia 2012an egin da, sei hilabeteko tartean.

1 http://ixa.si.ehu.es/Ixa

Proiektuaren izaera berezi hau dela eta, proiektuaren helburu-dokumentu hau egiteko orduan sei hilabeteko zati hori da planifikatzerakoan kontuan hartu beharko dena.

Lehenengoz, OmegaT itzulpenerako laguntzazko aplikazioa egokitu egin da bi emaitza izanda buruan: euskarazko Matxin itzultzaile automatikoa integratzea eta euskal Wikipediarako artikuluak itzuli eta igo ahal izatea tresna erabiliz. Geroago irismen hau zabaldu egin da, edozein hizkuntzetako Wikipedietarako ere ibili ahal izateko. Lan hau OpenMT2[2] (itzulpen automatiko hibridoa eta ebaluazio aurreratua) proiektuaren parte da eta Euskal Wikipediarekin batera egin da lan aurrera jarraitzeko. Aipaturiko aldaketak egiteaz gain, horien emaitzak eta Wikipediako komunitateak nola jaso duen aztertuko dira labur.

Jarraian itzulpen automatikoaren ebaluazioan sakondu da Asiya tresnaren bitartez. Honek erabiltzen dituen metrikei eta horien esanahiei buruz jorratu da eta horietako hainbat euskararako egokitzea da helburua, IXA taldeak eskainitako testuen analisiak gehituz horiei (*chunking, stemming, dependencies*...). METEOR, TERp, Shallow Parsing eta Dependency Parsing metrikak eta horien aldaerak izango dira aztergai. Azkenik, proiektuaren hirugarren atal nagusian itzulpen automatikoan testuen aurre-edizioa jorratu da, horretarako Diseg aplikazioa erabiliz esaldi luzeak laburragoetan banatzeko eta Matxin eta Asiya erabiliz emaitzak aztertuz.

Hortaz, proiektu honen helburuen artean hiru aplikazio desberdinekin itzulpen automatikoa lantzea da, IXA taldeko beste proiektuekin erlazionatuta dagoen lana eginez. Hala eta guztiz ere, ez da ahaztu behar azken finean hau karrera-bukaerako proiektu bat ere badela, eta beste helburu bat ere badu, azkena izan arren garrantzitsua oso: ikastea. Izan ere, proiektu honetan landutako dena erabilgarria izango da etorkizunean hainbat zentzutan. Alde batetik, Lengoaia Naturalaren Prozesamenduan oso nabaria bihurtzen ari den arlo batean, itzulpen automatikoa, sakontzeko aukera da hau, tresna, kontzeptu eta komunitate

2 http://eu.wikipedia.org/wiki/Wikiproiektu:OpenMT2_eta_Euskal_Wikipedia

berriak ezagutuz. Bestetik, programazioaren aldetik ere asko ikasteko aukera dago proiektu honetan, beren kodea moldatuko diren bi tresnek, OmegaT eta Asiya, nik aurretik landu gabeko programazio-lengoaietan idatzita baitaude: Java eta Perl, hurrenez-hurren. Amaitzeko, proiektuaren izaera eta tamaina direla eta, proiektu handi batean lan egitea jorratuko da, horrek dakarren zailtasun eta arazoei aurre egin beharrarekin.

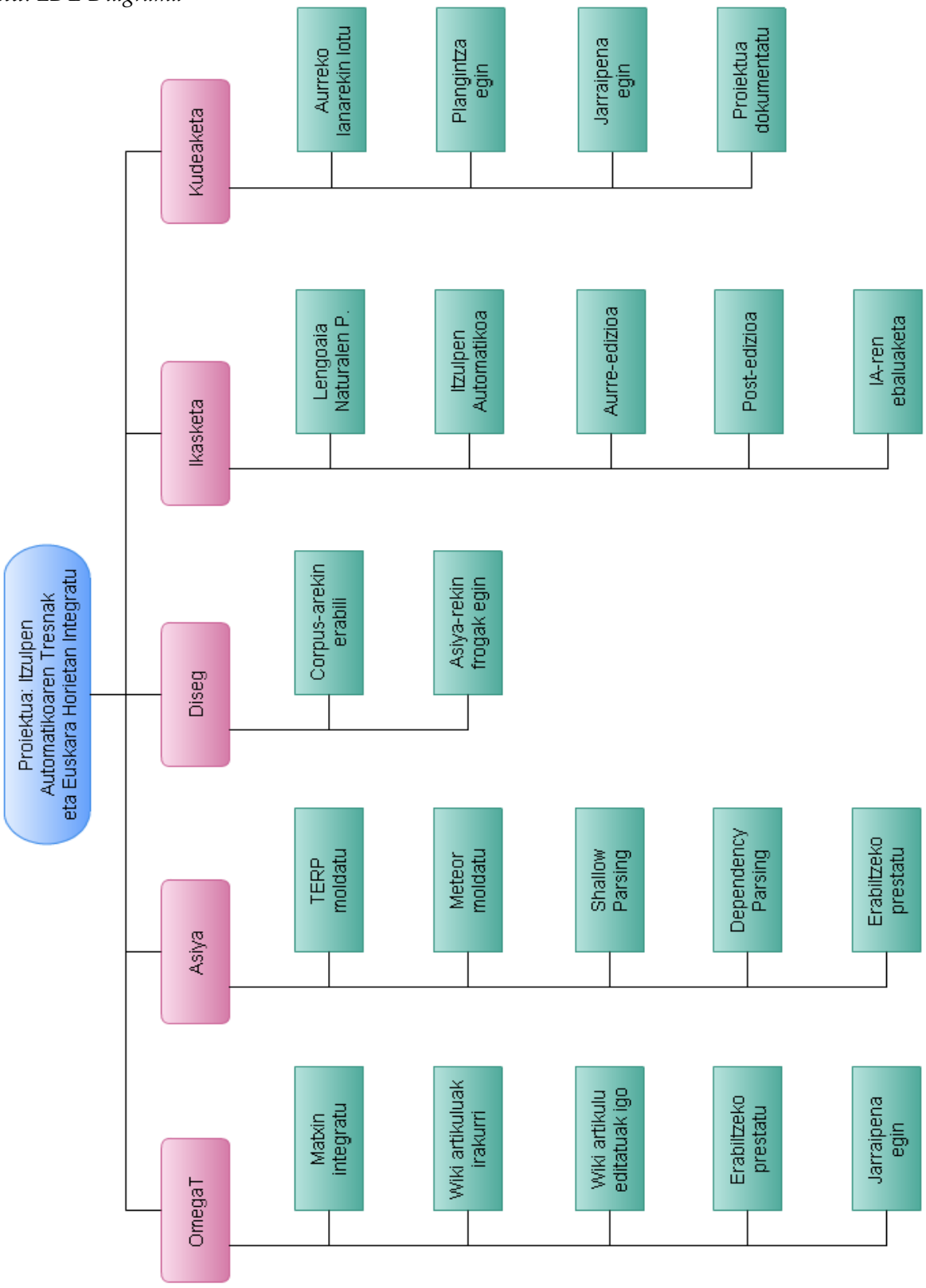
Proiektuak irismen zabala du beraz, hiru atal zehatz eta ikasketa-prozesu orokor batekin. Dokumentu honetan zehar, bai ikasitakoari buruz eta aplikazioei buruz jorratuko da, egindako lana erakusteaz gain, kontestu handiago batean jarri eta ikusteko.

1.2 Atazak

Proiektuan zehar agertzen diren atazen deskribapenak:

- **OmegaT**
  - *Matxin integratu*: OmegaT aplikazioa moldatu euskarazko Matxin itzultzaile automatikoa sartzeko beste itzultzaile automatikoen artean, edozein erabiltzailek atzitzeko moduan.
  - *Wiki artikuluak irakurri*: OmegaT-k MediaWiki artikuluak jaso eta editatzeko ahalmena ematea zihurtatu, Euskal Wikipediari begira batez ere.
  - *Wiki artikulu editatuak igo*: OmegaT-k itzulitako MediaWiki artikulu bat euskarazko edo beste hizkuntza bateko Wikipediara igotzeko ahalmena izatea.
  - *Erabiltzeko prestatu*: Behar adina moldaketa egin aurreko funtzioak ongi integratzeaz gain erabiltzaileak modu erosoan erabiltzeko, berau Wikipediako eta OmegaT-ren formatuetaz arduratu gabe.
  - *Jarraipena egin*: Behin OmegaT jendeari eskainita, OpenMT2 proiektuaren parte bezala lortzen diren emaitzen azterketa.

***1. Irudia**: LDE Diagrama*



- **Asiya**
  - *TERP metrika moldatu*: Asiya-ren ebaluaketarako metriken artean, TERP metrikak moldatu euskarazko analizatzailea erabiliz euskara ere aztertu ahal izateko.
  - *METEOR metrika moldatu*: TERP-en berdina, baina METEOR metrikari aplikatua.
  - *Shallow Parsing metrika moldatu* : Aurreko bera, baina Shallow Parsing (SP) metrikarekin.
  - *Dependency Parsing metrika moldatu*. Dependency Parsing (DP) metrikak ere euskara ebaluatu ahal izatea.
  - *Erabiltzeko prestatu*: Moldatutako metriketaz gain beste hainbat daude euskarara itzulitako testuak onartuko dituztenak gure analisia gabe – horiek denak ere prestatu analisiak egin ahal izateko.

- **Diseg**
  - *Corpus-arekin erabili*: Proiektuan erabilitako corpus-aren gainean erabili Diseg aplikazioa esaldiak mozteko.
  - *Asiya-rekin frogak egin*: Asiya erabiliz, ebaluatu ea aurreko atazean egindako aldaketek zer eragina duten itzulpen automatikoaren emaitzetan.

- **Ikasketa**
  - *Lengoaia Naturalaren Prozesamendua*: Proiektua egikaritzen den bitartean Lengoaia Naturalaren Prozesamendua hautazko ikasgaia ikasiko da, bertan gaiari buruz gehiago ikasiz eta ikasitakoak proiektuan aplikatu ahal izanik.
  - *Itzulpen Automatikoa*: Itzulpen automatikoa gai zabala da, baina horri buruz ahal bezain sakon ezagutzen saiatuko da. Erabiliko diren lengoaiak (Java eta Perl) ikastea ataza honetan sartzen da, gehienbat mundu honetan daukaten

erabilerak ikasiko direlako, eta hori atal nahiko orokorra delako.

- *Aurre-edizioa*: Itzulpen automatikoaren aurre-edizioari buruz ikasi, Diseg esaldi segmentatzailearekin erabiltzeko gero.
- *Post-edizioa*: Itzulpen automatikoaren post-edizioari buruz ikasi, OmegaT itzulpenerako laguntzazko tresnan aplikatuko baita.
- *IA-ren ebaluaketa*: Itzulpen automatikoaren ebaluaketarako dauden metrikak ikasi eta ulertu, Asiya-k erabiltzen baititu eta batzuk moldatuko baitira.

- **Kudeaketa**
  - *Aurreko lanarekin lotu*: Proiektua bezala, berez, egingo dena aurretik gai berean egindako lanarekin lotu, proiektu osoa eta horren ikuspuntua eratuz.
  - *Plangintza egin*: Proiektuaren plangintza prestatu, hau modu egituratuan garatu ahal izateko.
  - *Jarraipena egin*: Proiektuan zehar plangintzan ezarritakoa nola aldatzen den aztertu, arazoei modu egokian aurre egin, etab.
  - *Proiektua dokumentatu*: Bukatzerakoan, dokumentazioa egin beharko da. Hainbat zati aurrikusten dira: proiektuaren helburu dokumentua, proiektuan egindakoaren atzean dagoen gaiari buruz hitz egin, egindako lanaren azalpena, emaitzen azterketa eta beharrezkoak diren eranskinak.

1.3 Baliabideen esleipena

Proiektu honetan erabiliko diren baliabide materialak asko ez dira, lan indibiduala izanik eta ordenagailuen alde fisikoan asko jorratzen ez duelako. Software aldetik, aldiz, aplikazio anitz erabili beharko dira finkatu diren helburuek tresna desberdinekin lan egitea exigitzen baitute.

Hardware aldetik:

- Hiru ordenagailu desberdin erabili dira proiektuan zehar, une bakoitzeko beharrak asetzeko. Kontuan hartu behar da gainera pare bat urteetan zehar zabaltzen dela proiektu hau eta litekeena giro desberdinetan lan egin behar izatea dela. Bakoitzak betekizun batzuk baditu, bakoitzarekin behar izan diren Software baliabideak erabili ahal izatea, esaterako.
    - Erabilitako lehen makina IXA Taldearen mahaigaineko ordenagailu bat da, bekarientzat prestatua eta Red Hat-en Fedora Sistema Eragilea erabiltzen duena. Hau proiektuaren lehenengo zatian erabili da batez ere, OmegaT-rekin erlazionatua.
    - Bigarrena ere IXA taldearen mahaigaineko ordenagailua da, arazo teknikoak direla eta egin denik aldaketa, eta Ubuntu 11.04 erabiltzen duena; proiektuaren azken urtean zehar erabili egin da.
    - Hirugarrena mahaigaineko ordenagailu pertsonal bat da, etxetikan ere lan egiteko baliagarria. Windows XP erabiltzen du eta batez ere dokumentazio lanak egiteko erabiliko da, nahiz eta bestelako frogak egiteko ere balio duen, OmegaT probatzeko, esaterako.
- Datuak batetik bestera pasatu eta garraiatzeko, 16 GB-eko USB memoria bat.
- Azkenik, datuen segurtasun kopiak modu seguru eta erosoan gorde ahal izateko, etxeko ordenagailuaren parean dagoen 200 GB-eko kanpoko disko gogor bat erabiliko da.

Softwarearen aldetik, berriz, honakoak dira erabiliko diren baliabideak:

- Fedora Sistema Eragilea proiektuaren lehen zatian, OmegaT-rekin zerikusia duena. Hau erabiltzen saiatuko da OmegaT-n aldaketak egiten: Matxin itzultzeailearekin komunikatu SOAP eta web aplikazioen bitartez eta Wikipediaren aplikazioak erabiliz ere itzulitako testuak bertara igotzea.
- Ubuntu 11.04 Sistema Eragilea proiektuaren bigarren zatian, Asiya-rekin zerikusia

duena. Erabiltzeko erosoagoa deritzot Fedora baino.

- Windows XP Sistema Eragilea etxeko ordenagailuan, batez ere dokumentazioarekin zerikusia duen lana egiteko.
- OpenOffice eta LibreOffice dokumentuak editatu, ikustatu eta formatua emateko.
- Eclipse Galileo, NetBeans, eta hainbat *add-on*, OmegaT-ren Java kodea editatzeko.
- Apache Axis, SOAP web zerbitzua sortzeko OmegaT eta Matxinentzat.
- Gedit, hainbat testu fitxategi editatzeko, eta baita Asiyaren kodea Perl-ez aldatzen.
- Google Chrome, Chromium eta Mozilla Firefox, Internet nabigatzaile bezala.
- Java eta Perl, programazio-lengoaiak bezala.
- Proiektuarekin zerikusi jakina duten baliabide asko ere erabiliko dira. **Baliabideak** atalean aztertzen dira mantsoago.
    - *NewsTEST* corpusa
    - Wikipedia[3]
    - Matxin[4]
    - OmegaT[5]
    - Asiya[6]
    - Diseg[7]

1.4 Denboraren planifikazioa

(hurrengo orrialdean) ***2. Irudia****: Kronograma.*

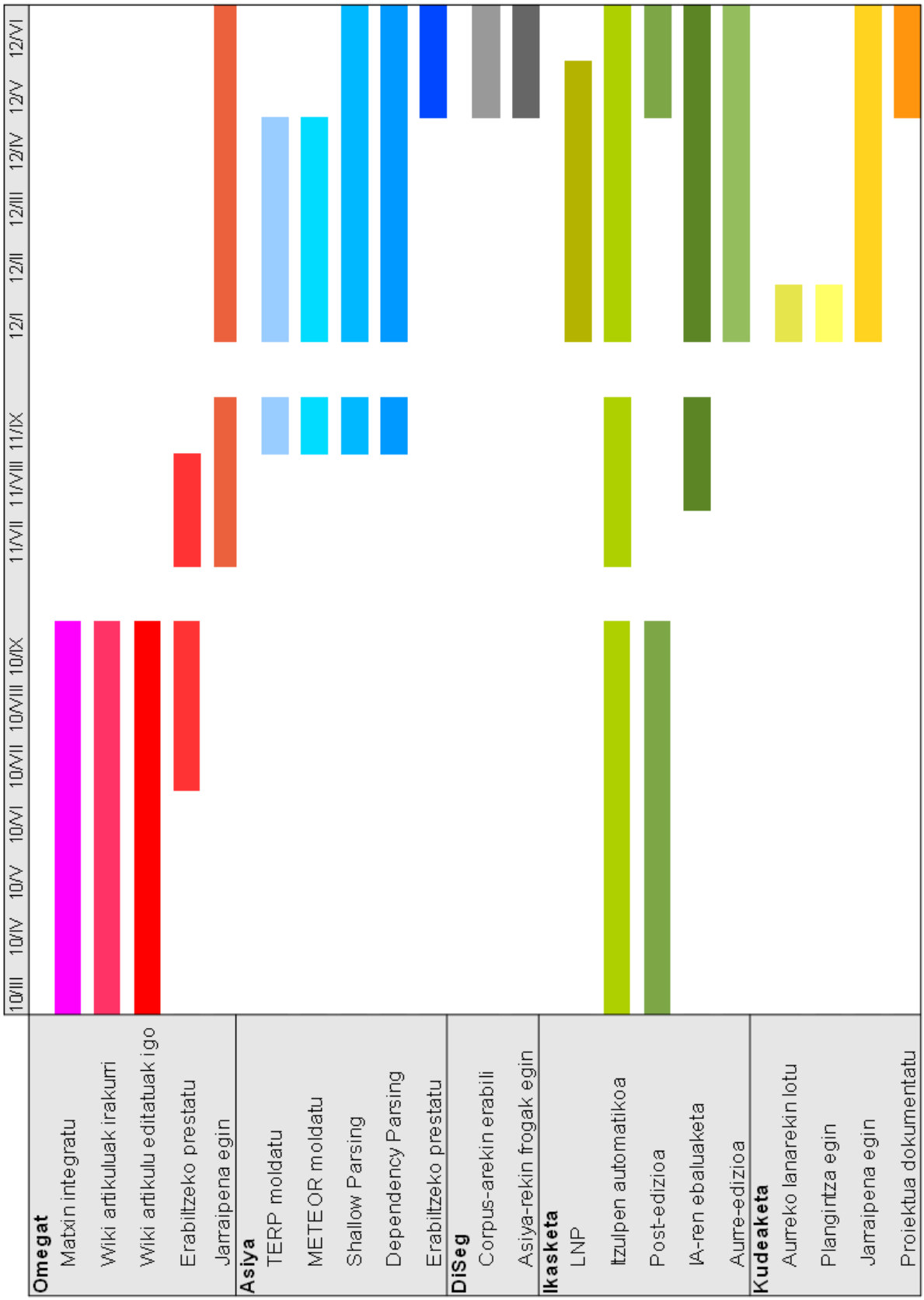
3 http://www.wikipedia.org/

4 http://matxin.sourceforge.net/

5 http://www.omegat.org/en/omegat.html

6 http://nlp.lsi.upc.edu/asiya/

7 http://daniel.iut.univ-metz.fr/~iula/WebDiSeg/index.php


10/III
10/IV
10/V
10/VI
10/VII
10/VIII
10/IX
11/VII
11/VIII
11/IX
12/I
12/II
12/III
12/IV
12/V
12/VI
Omegat
Matxin integratu
Wiki artikuluak irakurri
Wiki artikulu editatuak igo
Erabiltzeko prestatu
Jarraipena egin
Asiya
TERP moldatu
METEOR moldatu
Shallow Parsing
Dependency Parsing
Erabiltzeko prestatu
DiSeg
Corpus-arekin erabili
Asiya-rekin frogak egin
Ikasketa
LNP
Itzulpen automatikoa
Post-edizioa
IA-ren ebaluaketa
Aurre-edizioa
Kudeaketa
Aurreko lanarekin lotu
Plangintza egin
Jarraipena egin
Proiektua dokumentatu

2010ean, guztira, 440 ordu eman ziren. 80 inguru (%18a) ikasketa atalean sartzen dira (itzulpen automatikoa eta post-edizioa, OmegaT-rekin zerikusia dutenak), eta 360 ordu (%82a) OmegaT bera lantzen, funtzio desberdinak gehitzen eta frogak eginez.

2011an, guztira, 198 ordu eman ziren. Estimatzen da denboraren bi heren, 132 ordu, gutxi gora-behera, OmegaT-ri azken ukituak ematen, programaren erabilera azaltzen eta zabaltzen, eta jarraipena egiten hasten eman zirela. Beste 66 orduak, herena, Asiya-ren proiektuan murgiltzen izan ziren alde batetik, eta itzulpen automatikoaren erabilerari buruz ikasten eta softwarearekin trebatzen, bestetik.

2012an, 300 ordu estimatzen dira. 35 ordu, %12a, ikasketa-prozesu jarraiari eskaini zitzaion, bai Asiya eta DiSeg-ekin jarraituz eta baita dokumentazioa idazteko informazioa lortuz (ez da Lengoaia Naturalaren Prozesamendua ikasgaiko denbora kontuan hartzen honetarako); 150 ordu, %50a, Asiya moldatzen eta erabiltzen jorratuko zen. 15 ordu, %5a, DiSeg-ekin esperimentua egiteko. Beste 100 orduak, %33a, dokumentazioa egiteko eskainiko dira.

Beraz, guztira, ordu kopuru hauek estimatzen dira:

OmegaT:	360 + 132 =	492 ordu
Asiya:	30 + 150 =	180 ordu
DiSeg:	15 =	15 ordu
Ikasketa:	80 + 36 + 35 =	151 ordu
Kudeaketa:	100 =	100 ordu
GUZTIRA:		**1038 ordu**

1.5 Kalitatearen kudeaketa plana

Proiektuan zehar kalitatezko gutxieneko batzuk ezartzea saiatuko da, bai honen kudeaketan zehar eta baita proiektuan bertan ere.

Dokumentazioan:

- Azala
- Zein atal egongo diren
- Atal bakoitza orrialde bakoiti batean hasiko da
- Proiektuaren laburpena azalduko da lehenik
- Aurkibide argi eta zehatz bat hasieran
- Bibliografia eta iturriak adieraziko dira modu argian
- Beharrezko eranskinak gehituko zaizkio dokumentazioari
- Testuak formatu argia eta irakurgarria, nahikoa marginekin
- Letra mota: *Book Antiqua*, 12, testuan, *Courier New*, 8, kodean eta *Arial* tauletan.
- Orrialdeak zenbakituak egongo dira
- Orrialde orotan egongo da ikaslearen identifikazioa
- Irudi, taula eta kode zatiak zenbakitu eta aurkibide propioa izango dituzte
- Kode-zatietan iruzkinak kenduko dira labur eta zehatzago izateko; beharrezko azalpenak inguruko testuan agertuko dira.
- Garaiz entregatuko da

OmegaT-n:

- Beste itzultzaile automatikoak bezain erraz atzitu eta erabiliko da Matxin.
- Matxin atzitzea ahal den modu azkar eta eraginkorrenean egingo da.
- Modu errazean deskargatu eta editatu ahal izango dira Wikipediako artikuluak.
- Posible izango da itzulitako artikulu bat bere hizkuntzako Wikipediara igotzea.
- Itzulpen-memoriaren bidalketa erabltzaileak denbora galdu gabe egingo da.

- Programa berez konplexua denez, erabiltzaile-gida argi bat egingo da.

Asiya-n:

- Erabakitako metrikak euskaraz itzulitako testuekin ibiliko dira.
- Ezer gehiago adierazi barik erabiliko da behin 'eu' hizkuntza gehituta.
- Programa konplexua izanik, eranskin bat sortuko da erabiltze basikoa egiteko aholku batzuekin.

DiSeg-en esperimentuekin:

- Esperimentu oro modu zehatz eta dokumentatuan egingo da.
- Esperimentuen emaitzak modu argian eta antolatuan erakutsiko dira.

1.6 Komunikazioen plana

Proiektu indibiduala izanik, barne komunikaziorik ez dago.

Kanpo komunikazioei dagokienez, kontaktu nagusiak hiru izango dira: Kepa Sarasola irakaslea eta proiektu honen zuzendaria; Iñaki Alegria irakaslea; eta Gorka Labaka lankidea, bulego berean lan egiten duena eta erlazionatutako proiektuetan lan egiten duena. Komunikazioa ahozkoa izango da posible denean, eta posta elektronikoz bestela. Ez dago protokolorik definituta komunikazio hauentzat.

Noizean behinkako komunikazio honetaz gain, bilera erregularrak ere egingo dira irakasleekin proiektuaren jarraipenari eta eginbehar desberdinei buruz hitz egiteko.

OmegaT-ren gainean lan egiten duen garapen talde handi bat dago. Horiek laguntza handia eskaini dezaketenez OmegaT-rekin sortzen diren zalantzei buruz, oso erabilgarria

suertatuko da beren mezu-listan sartzea. Alde batetik berez eskaintzen duten informazioa eta elkarrizketa irakurri ahal izango dira, eta bestetik, behar izatean beraiekin kontaktuan jarri ahal izango naiz.

Azkenik, bestelako garatzaileekin kontaktuan jartzeko, hau da, Asiya edo Diseg-ekin zerikusia duten horiekin, proiektuaren zuzendaria izango dira bitartekari hasiera batean.

1.7 Arriskuen kudeaketa

- Denbora-estimazio okerrak egitea.
  - Baliteke egindako denboren estimazioak txarto egotea. Hau ez da arazo larria amaierako denbora estimatutakoa baina zertxobait baxuagoa izanez gero, beti egin baitaiteke lan gehiago proiektuaren gainean. Baina denbora asko luzatzen bada, larria bihurtu daiteke, ezarritako epeak ezingo bailira bete. Gainera, proiektuak hilabete asko hartuko dituenez, horietan zehar zabalduz, estimazioa are zailagoa izango da.
  - *Eragina*: Aldakorra. Azaldu denez, denbora kantitatearen eta noranzkoaren araberakoa desberdina gerta daiteke.
  - *Probabilitatea*: Altua. Tamaina honetako lan gutxi eginda aurretik, eta azken finean proiektu hau nire ikaste-prozesuaren parte dela kontuan hartuz, zaila egingo da hasierako estimazio zehatz bat egitea.
  - *Erantzuna*: Arindu. Estimazioaz kontu handiz egin beharko dira eta denboren jarraipena tentu handiz egingo da atzerapen posibleak lehenbailehen antzemateko eta horiei aurre egiteko, kasuaren arabera. Plangintzaren errebisio eta jarraipenean hau kontuan hartu beharko da eta estimazio berriak horretara moldatu.

- Ataza berriak sortzea.
  - Lan handi bat izanik eta azken finean lehenago horrelakorik egin ez denez, posiblea da baita ere plangintzaren beste zatiak, denboraz gain, gaizki egotea. Eta atazen banaketa horietako bat izan liteke, espero ez den zeregin bat sortu baitaiteke lana egiten den bitartean.
  - *Eragina*: Ertaina. Berez ez du eragin handia, lan gehiago egin beharko da eta again ikasketa prozesu bat izango du. Baina larriena beste arriskuak sor ditzazkela da (adibidez, denbora estimazioa okertzea).
  - *Probabilitatea*: Baxua.
  - *Erantzuna*: Onartu. Atazak definitu diren moduan, bakoitzak atal zehatz batzuk hartzen baditu ere, nahikoa lasaiera dute lan berriak horietan sartzeko, eta plangintzaren zati hori ez da asko aldatuko. Denbora gehiago eskatuko badu, denbora hori eskaini beharko da edo atazak nola murriztu aztertu.

- Informazioaren galera (digitala).
  - Informatikaren munduan arrisku ohikoa da informazio digitalaren galera zalantzarik gabe. Zenbaitetan kanpoko arrazoiengatik izan daiteke: makinek arazo teknikoak izan ditzazkete, hondamen natural bat gertatu daiteke, malwareak ezabatu dezake... Beste hainbatetan, erabiltzaileen erroreengatik izango da: nahigabe borratzea, nahita borratzea geroago erabiliko dela ez ikusiz, teknikari batek borratzea...
  - *Eragina*: Oso altua. Proiektuaren informazioa galtzeak denbora eta lan kopuru oso handiak galtzea ekarriko luke.
  - *Probabilitatea*: Baxua. Gertatzeko modu asko dauden arren, ohikoa ere ez da.
  - *Erantzuna*: Ekidin. Segurtasun kopiak mantenduko dira uneoro eta modu erregularrean eguneratuko dira. Erabilitako informazio gehiena gordeko da bertan, nahiz eta posible izatea programa handi batzuk erraz mugitu ahal ez izatea. Honek arrisku gehienen aurrean ere datu digitalak salbu

mantenduko direla egiaztatzen du.

- Informazioaren galera (paperean).
  - Informatika mundutik kanpo ere paperean, modu fisikoan, dagoen informazioa ere galtzeko aukera hor dago. Proiektuan zehar informazio gehiena digitala izango den arren, baliteke paperean ere gauza anitz egotea: programatzen den bitartean hartzen diren apunteak, frogak egiterakoan hartuko direnak, bileretan esandakoa, etab. Eta paperean dagoena beti desager daiteke...
  - *Eragina*: Ertaina-txikia. Informazio galera izanik ere, garrantzitsuena ez da emen egongo eta ez da oso zaila izango, printzipioz, berreskuratzea.
  - *Probabilitatea*: Oso baxua. Paperezko dokumentazio gehiena IXAko bulegoetan edo neure etxean egongo direnez, kontrolpean egongo da beti.
  - *Erantzuna*: Ekidin. Paperean dagoen informazio erabilgarri gehiena modu digitalera pasako da lehenbailehen, hori bezala tratatuz hortik aurrera.

- Baliabide materialen erabilgarritasuna.
  - Ia edozerk sor dezake arazorik. Ordenagailuak funtzionamendu txarra edukitzen hastea, softwareren bat eskuragarri egoteari uztea, lortutako softwareren batek emaitza desiragarriak ez ematea...
  - *Eragina*: Aldakorra. Berez baizik, beste arriskuak sor ditzazkelako: denbora luzatu, atazak gehitu, informazio galera...
  - *Probabilitatea*: Ertaina. Izatez ez da oso altua, baina denbora luzeko proiektua izanik, litekeena gutxienez behin gertatzea da, nahiz eta eragin txikikoa bada.
  - *Erantzuna*: Arindu. Guztiz ezin bada ekidin ere, prest egon beharko da gauzak beste modu batean egiteko baliabide material bat ez badabil. Hala ere, baliabideak zaintzen eta modu errazean konpondu edo aldatu daitezkenak aukeratzen, eragin handienak saihesten saiatu daiteke.

- Ikaslearen egoera.
  - Pertsona baliabide bakarra du proiektu honek, baina horrek ere huts egin ahal izatea kontsideratu behar da. Baliteke osasun kontuengatik edo lan atzerapenak sortzea.
  - *Eragina*: Handia. Proiektu indibiduala da – pertsona bakarrak ez badu lan egiten, proiektua gelditu egingo da.
  - *Probabilitatea*: Baxua. Hala espero da behintzat.
  - *Erantzuna*: Onartu. Kanpo arazoak direla eta ikasleak ezingo badu lan egin, onartu beharko da. Hala ere, giza baliabideak ongi zaintzeko ahalmena egingo da.

- Proiektuan geldialdiak.
  - Proiektu honek hartzen duen denbora handia dela eta, geldiune ziurrak egongo dira. Alde batetik, opor garaietan. Eta bestetik, kurtsoetan zehar arrazoi akademikoak dituzten beste geldialdiak egin beharko direlako, ikasketei denbora gehiago eskaintzeko, beste ikasgaietako lanak egiteko eta azterketak eta ongi prestatzeko. Hori saihestezina da, baina arriskua informazioa ahaztean datza: bizpahiru hilabeteko geldialdi batek egitenari zen lana ahazteko aukera handia ematen du.
  - *Eragina*: Handia. Denbora galera asko ekar ditzazke "berrikasketak".
  - *Probabilitatea*: Oso altua.
  - *Erantzuna*: Arindu. Ahal bezain beste apunte, ohar eta bestelako anotazioutziko dira kodeetan zehar. Unerarteko dokumentazioa eta kudeaketa ere ahalik eta garbien egingo da etorkizunean argi ulertzeko.

- Ezjakintasuna.
  - Proiektuan zehar jorratuko diren gai gehienak modu batean berriak direnez niretzat, ikasketa prozesu handi bat egon beharko du, baina baliteke honek

ondorio ezikusiak izatea beste arloetan eta bestelako arriskuak ekartzea. Proiektua hasterakoan Lengoaia Naturalaren Prozesamenduari buruzko ikasgairik ez dut izan, horren tresnekin kontaktu gutxi izan dut eta erabiliko diren Java eta Perl lengoaiak ez ditut inoiz ukitu.

- *Eragina*: Txikia. Eragina badu, baina ez da larria, aurreikusia baita.
- *Probabilitatea*: Ziurra.
- *Erantzuna*: Arindu. Proiektuaren plangintzaren parte bezala gehitu egin da ikasketa prozesu bat egon beharko delaren suposizioa ataza moduan.

- Atazen emaitza ez desiragarriak.
  - Baliteke atazetan espero ez diren emaitzak lortzea edota ezinezkoak suertatzea ere. Adibidez, kodeak prest ez egotea aldaketa jakin batzuk jasateko, esperimentuen emaitzak uste zenaren desberdinak izatea edo ataza batzuen emaitzak guztiz ezinezkoak suertatzea.
  - *Eragina*: Oso handia. Proiektu osoaren helburu eta ondorioak alda daitezke.
  - *Probabilitatea*: Oso txikia. Ez da litekeena tamaina handiko aldaketak suertatzea. Kasu txikietan probabilitatea handiagoa da.
  - *Erantzuna*: Onartu. Emaitzak, emaitzak dira.

2. Sarrera

Itzulpenari buruzkoa

Antzina izan zen gizadia banandu, munduan zehar sakabanatu lurralde berrien bila eta hizkuntza desberdinak garatu zituztela. Idatzizko Historia sortu bezain pronto badakigu itzulpenak agertu zirela. Izan ere, duela lau mila urtetik gora idatzi egin zen *Gilgameshen epopeia* sumerieraz, gaur egungo gizakiak ezagutzen duen literatura antzinenetarikoa, eta handik gutxira bazegoen itzulpenik akadieraz.

Harrezkero, garrantzi kulturala eta ezagutza desira izan duten potentziek beren garaian itzulpenaren erabilera eta garapena beharrezkoak izan dituzte. Egiptoko faraoiek itzulitako testuak behar izan zituzten hainbat helbururekin, adibidez, hititekin bakea ezartzeko (Kadesheko ituna) edo faraoi berriaren boterea indartzeko (Rosettako harria). Greziar eta erromatar klasikoen filosofoek itzulpengintzari buruzko teoria zorrotzak garatzen hasi ziren. Zhou dinastiako txinatarrek itzulpen lausoak erabili zituzten budismoa zabaltzerakoan. Erdi Aroko musulmanek arrazoi politikoak zirela eta, kontaktuan zeuden beste herrialde askorekin komunikatzeko hainbat teknika garatu egin zituzten. Eta jakina da

Itun Zaharra eta Biblia bezalako testu erligiosoen itzulpenetan akats ugari daudela, batzuk aspaldiko partez hebreeratik itzulpenez-itzulpen zabaldutakoak, eta beste batzuk Errenazimendu garaian, ideologia desberdinek itzulpen desberdinak sortu zituztenean.

XVIII. mendean, Industri Iraultzarekin batera, itzulpengintza zabaldu egin zen itzultzaile profesionalak agertu eta itzulpengintza bera unibertsitateetan ikasten, lantzen eta ikertzen hasi zirenean. XX. mendean informatikaren etorrerarekin batera ordenagailuaren agerpenak gure sozietatea, ohitura, ikuspuntu eta ezagutza aldatu egin ditu, eta itzulpengintza ez da alde batera utzi. Honela sortu dira itzulpen automatikoa eta ordenagailuak lagundutako itzulpengintza, proiektu honetan landu egingo direnak. Baina ikusi ahal izango denez, hauek ez dira gauza erreza ordenagailuarentzat.

Izan ere, itzulpengintza ez da erraza gizakiontzat ere. Itzultzaile on batek bi hizkuntzak, jatorrizkoa eta helburu-hizkuntza, menperatu behar ditu. Eta ez hizkuntzak soilik, baita ere espresioen eta esaeren arteko erlazioak, nola edo hala horien kulturen parte direnak. Gainera, itzultzen ari den testuaren kontestuari buruzko gutxieneko ezagutza batzuk behar-beharrezkoak dira. Azkenik, antzinako greziar eta txinatarrek aurkitu zuten bezala, oreka bat bilatu behar da *metafraseatu* (modu literalean itzuli, hitzez-hitzezko egitura mantenduz) eta *parafraseatzearen* (helburu-hizkuntzan zentzua izan dezan moldatu) artean, esanahia mantentzeko asmoz.

Pertsona eleaniztun batentzat zaila izan daiteke itzulpen bat modu egokian egitea eta itzultzaile profesional batek ere hankasartzeak izan ditzake. Hortaz, zenbateko zailtasuna izango du ordenagailu batek, makina batek, itzulpen on bat egiteko? Noraino iristeko ahalmena du? Itzulpen automatikoa tresna erabilgarria da oso, nahiko garatua baina oraindik lan asko aurretik duena eta etorkizunean garrantzi handia izango duena zalantzarik gabe. Hortaz, itzulpen automatikoari buruzko lan hau egitea, arlo desberdinak landuz, oso interesgarria iruditu zait.

3. Testuingurua

3.1 Itzulpen automatikoa

Definizioz, itzulpen automatikoa, *machine translation*[8] (MT) ingeleraz, ordenagailu batek testu bat hizkuntza batetik beste batera itzultzean datza. Ezagutza-arlo bat osatzen du, itzulpen horiek egiten dituzten softwareak aztertu eta hobetzeko. Sarreran azaldu den bezala, honek garrantzi handia izan dezake etorkizunean, itzulpenaren historia eta eboluzioaren hurrengo pausoa (eta pauso ikaragarri bat) baita. Baina hor ere ikusi denez, itzulpengintzaren zailtasunak asko dira eta garatzaileentzat ez da erraza itzulpen automatikoa modu egokian suertatzea. Eta tamalez, ordenagailuek ez dute lengoaia naturala gizakiok bezala *ulertzeko* ahalmena; beraz, informatikariaren eskuetan gelditzen da makinak testu bat nola interpretatu eta itzuliko duen zehaztea. Horregatik, hainbat teknika sortu dira tarteko emaitzak lortzeko... Ez perfektuak, baina bai ulergarriak lengoaia naturala irakurtzen duen batentzat.

Zer beharko du software batek itzulpen automatikoa egikaritu ahal izateko? Hasteko hizkuntzei buruzko informazio ugari. Baita horien erlazioei buruzkoa eta kontestuari

8 http://en.wikipedia.org/wiki/Machine_translation

buruzkoa. Baina kontuan hartu behar da ordenagailuek memoria espazio handia duten arren, limitazio asko ere badituztela eta egun ez dugula modurik ezagutzen pertsona batek lengoaia bati buruzko informazio *guztia* makina batean datu moduan errepresentatzen; hizkuntzaz gain pertsonaren esperentziak eta oroitzapenak ere zerikusia baitute informazioa lengoaia baten bitartez errepresentatzeko moduan. Softwareari informazioa emateko, beraz, modu desberdinak ditugu: lexikoa eta gramatika azaltzen dituzten arau multzoak, testu-corpus handiak, gai jakin batean zentratzea... baita Internet teknologiei esker sarearen bitartez lortu daitekeen informazioa ere baliagarri izan daiteke.

Hiru itzulpen automatiko mota nagusi daudela esan daiteke. Batetik, adibideetan oinarrituakoa, corpus elebidun bat erabiliz analogiak sortzen dituena esaldi desberdinen antzekotasunak begiratuz. Bestetik, estatistikoa, baita corpus elebidun bat duena, eta hortik metodo estatistikoak erabiltzen dituena itzulpen egokiena bilatzeko modua bezala. Azkenik, arauetan oinarritutakoa, arauak erabiltzen dituena lengoaiei buruzko informazio lexiko eta gramatikala jakiteko. Horren barnean bestelakoak ere aurki daitezke: hiztegizkoa edo zuzena, hitzez hitz egiten duena; lengoaiartekoa, tarteko lengoaia independiente bat erabiltzen duena; eta transferentzian oinarritua, tarteko lengoaia ere erabiltzen duena, baina bi hizkuntzen arteko egiturak eta ezaugarriak bakarrik kontuan hartuz. Azkenik, metodo desberdinak lotzen dituzten sistema hibridoak ere badira.

Guzti horretatik lortu dezakegun ondorio nabarmen bat: sistema perfekturik ez dagoela. Teknika desberdinak edo horien arteko konbinazioak erabiltzen dituzten aplikazioak badira. Batzuk dohakoak dira, beste batzuk ordaindupekoak; batzuk kode irekikoak, eta beste batzuk ez. Horretaz gain, garrantzi handia du baita ere zein domeinutan erabiliko den softwarea. Esaterako, hizkuntza pare desberdinetarako, itzultzaile automatiko bat edo beste izan daiteke eraginkorrago. Erabilitako lexikoak ere garrantzia du, itzultzaileak dituen arauak edota corpusa-k arlo jakin batean lan handiagoa egina badute, emaitza hobeak aterako dituzte; baina zerikusi handia ez duten testu anitz itzuliko badira, baliteke itzultzaile "orokor" batekin hobeto lan egitea. Eta bestelako kontuak ere garrantzia badute: adibidez,

adibideetan oinarritutako itzulpen automatikoak emaitza hobeak lortzeko aukera du hizkuntza baten berebiziko ezaugarrietan, ingelerazko aditz partikuladunak bezala (*phrasal verbs*), itzultzerakoan.

Hala eta guztiz ere, nonahi aurki ditzakegu itzulpen automatikoaren erabilerak. Gaur egungo mundu globalizatuan, garapen teknologikoari esker pertsona oro gaude elkarrekin konektatuta eta telekomunikazio, Internet eta garraiobide berriei esker, distanziak ia existituko ez balira bezala izanda, egunero dago komunikazio handia kultura eta hizkuntza desberdinak dituzten pertsonen artean. Hori horrela izanik, teknologien onurak aprobetxatu daitezke itzulpenak berehalakoak, ohartezinak, lekukoak izatea lortu daiteke. Mahigaineko ordenagailu batean edo gailu mugikor batean, sare sozialetan edo enpresen arteko komunikazioetan, edonon agertzen dira itzulpen automatikoko aplikazioak.

Ondorengo ataletan lan honetan landuko diren atalekin zerikusia duten itzulpen automatikoaren gai zehatzak azalduko dira orain: itzulpenean laguntzeko programak, itzulpen automatikoaren ebaluazioa eta aurre-edizioa.

3.2 Itzulpen automatikoa euskaraz: Matxin

Mundu globalizatu honetan, euskarak ere bere lekua baduela esan daiteke. Sistema eragile eta aplikazio anitz eskuratu daitezke hizkuntza honetan, Windows edo Ubuntu, Microsoft Office edo LibreOffice. Sarean ere webgune oso erabili ugari euskaraz daude: Facebook eta Tuenti bezalako sare sozialak, Google eta Bing bezalako bilatzaileak eta Wikipedia bezalako orriak. Izan ere, Wikipediak argitaratutako azken estatistiketan, 2008ko urrian, euskarazkoa 47.a azaltzen da. Baina hiztunen kopurua aztertuz, badira ehundaka hizkuntza jende gehiagok hitz egiten dituenak (Ethnologue[9]-ren arabera, ia 400 hizkuntzek

9 http://www.ethnologue.org/

dute milioi bat hiztun baino gehiago, euskara zazpirehun milara iristen ez delarik). Artikulu gehien dituzten 50 Wikipediak aztertuz, bik soilik dute euskarak baino artikulu / hiztun erlazio altuagoak (hizkuntza artifizialak baztertuz): luxenburgera eta estoniera.

Hizkuntza	Zenbatgarren Wikia	Artikulu kop	Hizlariak (k)	Erlazioa
Luxenburgera	49	24331	320	**76**
Estoniera	34	54213	1050	**51,6**
Euskara	47	30454	665	**45,8**
Norvegiera	13	182666	4600	**39,7**
Finlandiera	14	179520	5000	**35,9**

***3. irudia**: Wikipedian biztanleko artikulu gehien dituzten hizkuntzak*

Ondorio nabarmen bat lortu dezakegu hortatik: euskara, hiztunen kopuruarekin alderatuz, eta nahiz eta beste hainbat hizkuntza nabarmenago izan Interneten, garrantzia badu eta bere lekua hartu egin du sarean eta XXI mendeko hizkuntzen artean.

Baina sareko presentzia horrek itzulpen automatikoan ere euskara bere lekuan egotea eskatu egiten du. Adibidez, OpenTrad[10], Apertium[11] eta Eusko Jaulraitzak[12] badute euskararekin itzultzeko aukerarik, eta baita GoogleTranslatek ere, baina ageri da ez direla gauza handirik eta asko gelditzen dela lantzeke horietan. Izan ere, euskara arlo askotan da desberdina beste hizkuntzekiko eta itzultzaile automatiko horiek eskaintzen dituzten beste hizkuntza gehienekin alderatuz hiztun kopurua baxua da eta ez dute hainbeste landu.

Proiektu honen interesei begira garrantzitsuena, zalantzarik gabe, Matxin itzultzailea da. IXA taldeak garatutako itzultzaile hau transferentzian oinarritutakoa da, arauetan oinarritutakoen artean, beraz. Aipaturiko teknologiekin bezala, lan asko gelditzen da aurretik... Kontuan hartu daiteke ingelera eta gaztelania bezalako hizkuntza handientzat ere

10 http://www.opentrad.com/

11 http://www.apertium.org/

12 http://www.itzultzailea.euskadi.net

itzulpen automatikoak zailtasunak dituela oraindik. Hala ere, aukera ona da, dohakoa eta publikoa, eta garrantzitsuena, IXA taldeko kideek etengabe ari direla hobekuntzak egiten eta itzultzailea hobetzen.

Horrez gain, Matxinek badu Internet bitartez atzitu daiteken API bat, aurrerago erabiliko dena OmegaT aplikazioan integratzeko. Gazteleraz edo ingeleraz dagoen testu bat pasata eta horren hizkuntza adieraziz, besterik gabe, itzulpena eskainiko da. Google Translate bezalako beste aplikazio batzuk bezain azkarra ez den arren, gure asmotarako oso erabilgarria suertatu da.

3.3 Itzulpenean laguntzeko programak eta post-edizioa: OmegaT

Itzulpen automatikoak erabilpen anitz ditu bere kabuz. Normalean itzulpen automatikoaren emaitzak oso zehatzak ez badira ere, hizkuntza bat ongi ezagutzen ez duen norbaitentzat, honelako itzulpen bat nahikoa izan daiteke irakurtzen ari den testua ulertzeko. Itzulpen automatiko azkarrak eduki ahal izateak ere hainbat abantaila eta erabilera ditu aplikazio askotarako: berehalako mezugintza sarean edota gailu mugikorretan, hizkuntza arrotzetako webguneak itzuli eta irakurtzerakoan... estatu-inteligentzia zerbitzuek ere atzerriko herrialdeen komunikazioak hobeto ulertzeko erabiltzen ohi dituzte, espioitza, kontrainteligentzia eta terrorismoari aurre egiteko, adibidez.

Baina badago beste arlo bat zeinetan itzulpen automatikoak laguntza handi bat eskaini dezakeen: itzulpengintza bera. Sarreran aipatu den bezala, itzulpen on bat egitea ez da erraza pertsonentzat ere, eta edozein laguntza izan daiteke ona. Baina bestalde, itzulpen automatikoa hain ona ez dela ere esan da. Zertan datza, orduan, itzulpenerako ordenagailu bidezko laguntza?

Itzulpen automatikoa alde batera utzita ere, informatikak tresna erabilgarri asko eskaintzen dizkio itzultzaile bati, jarraian, OmegaTren adibidearekin, ikusi ahal izango den bezala. Bilaketak, hiztegiak, aurretik itzulitako testuak atzitzea... Guzti hori modu oso azkarrean lortu dezake itzultzaileak ordenagailu bati esker. Horri itzulpen automatikoa gehitzen bazaio, itzulpenean laguntzeko tresna eraginkorra lortu daiteke.

OmegaT aplikazioak honelako asko eskaintzen ditu, itzulpen automatikoa barne. OmegaT-k testua itzultzeko aukera egoki batzuk ematen ditu, segmentuetan banatuz; itzulpen automatikoa edota aurrerago azalduko ditugun itzulpen-memoriak erabiltzeko aukera du, gero horiek post-editatu ahal izanik... Zerotik abiatzea baino lan azkarragoa gehienetan; eta beste hainbat tresna erabiliz lan hori are gehiago errazten du.

2000. urtean Keith Godfrey-k sortu zuen OmegaT[13], C++ programazio-lengoaia erabiliz. 2001an argitaratu zen lehenengo bertsio libre eta publikoa, honakoan Java lengoaiaz idatzia. Sortutako lehenengo bertsio horretan jabedun formatuko itzulpen-memoriak erabiltzen zituen; testu soila eta HTML formatuekin soilik zen lan egiteko gai; eta segmentazioa paragrafoka baino ez zuen egiten. Ordutik aldaketa ugari egon dira, itzulpen-memoria libreak erabiliz, testu-formatu kopuru handi batekin lan eginez eta segmentazio konplexuago eta eraginkorragoa erabili ahal izanik.

OmegaT-ren garapena SourceForge proiektu gisa egin da, denboran zehar munduko hainbat programatzailek editatu, zabaldu eta hobetu dutelarik. Didier Briel itzultzailearen agindupeko komunitate eta garapen talde garrantzitsua dago, lan handia egiten duena bertsio berrietan lantzen. Komunitate honetan, lengoaia naturalaren prozesamenduan edota beste alorretan dauden informatikari eta programatzaileetaz gain, itzultzaile eta erabiltzaileen iritziak ere oso kontuan hartzen dira, beren beharrak aztertu eta asetzen saiatzeko. OmegaT

13 http://www.omegat.org

ez da adituentzat soilik, edozeinek erabil dezan pentsatua baitago.

Oro har, irismen handia duela esan daiteke. Itzultzaile eta testu mota askori dago egokitua OmegaT. Hizkuntza askotarako dago prest, nahiz eta baliagarritasun batzuk hizkuntza gutxi batzuetarako baizik balio ez duten. Bestalde, fitxategi mota eta kodeketa desberdinak onartzen ditu. Horien artean, proiektu honetarako garrantzi handikoa izango den WikiMedia[14] formatua, Wikipedia eta antzeko Wiki webguneek erabiltzen dutena.

Hasteko, OmegaT-k eskaintzen dituen tresnen artean garrantzitsuak dira esaldi eta hitz konkretuekin erabili daitezken tresna txiki baina erabilgarriak. Adibidez, baditu eskuragarri hiztegi eta glosategi batzuk, izan elebakar edo elebidunak, eta gutxi diren arren, nahi adina gehitzeko aukera eskaintzen dio erabiltzaileari. Gainera, testu-editore orotan iada ikusten ohituta gauden tresnak badira: ortografia zuzentzaileak eta bilatzailea, esaterako. Guzti horiei esker, OmegaT erabiltzen ari den erabiltzaileak itzulpena egiten duen bitartean berehala jakin dezake zer esan nahi duen hitz batek, hiztegiak zer baliokide eskaintzen duen helburu-hizkuntzan, hitz bat gaizki idatzi ote duen eta hitz edo esaldi bat non dagoen jakin dezake. Hauek guztiak eguneroko tresnak direla esan daiteke XXI mendearen hasiera honetan, baina erabilgarriak dira oso hala eta guztiz ere, uneoro erabiltzen direnak.

Hala ere, OmegaT-ren hiru erabilera dira interesgarrienak; honakoak dira: segmentazioa, itzulpen automatikoa eta itzulpen-memoriak.

Segmentazioa. OmegaT-k itzuliko den testua hainbat segmentuetan banatzeko aukera du. Modu honetan, erabiltzaileak banan-bana itzuli ahal izango ditu, orokorrean erraztasuna gehituz prozesuari, itzultzailea esaldi soil bati baino ez baita adi egon beharko. Hala ere, gertuko segmentuak ikusgarri daude eta beraz kontestua ez da galtzen. Horretaz gain, segmentazioak beste tresnei, aurrerago ikusiko diren itzulpen automatikoa eta itzulpen-

14 http://www.wikimedia.org/

memoriei bezala, laguntzazkoa izango zaie, azkarrago eta eraginkorrago ibiliko baitira testu laburragoekin.

Segmentazioa egikaritzeko, OmegaT-k espresio erregularrak erabiltzen ditu. Baditu defektuzko espresio erregular batzuk testuak modu erosoenean zatituko dituztenak: esaldietan. Honela, puntuazio-marka jakin batzuk aurkitzen dituen orotan, esaldia bukatu egin dela suposatuko da eta hor egingo da segmentuen arteko zatiketa. Hala eta guztiz ere, badu aukera erabiltzaileak berak bestelako espresio erregularrak definitzeko eta segmentazioa beste modu batera egiteko, itzultzaileak egokien deritzon moduan. Horretarako erabiltzaileak espresio erregularrak baino ez ditu ezagutu behar, eta ez du zertan programaren kodea aldatu behar, ezta programazioari buruzko ezagutzarik ere ez.

Itzulpen-memoriak. Erabiltzaile batek OmegaT-ko proiektu batean itzulpen bat egiten duen bakoitzean, OmegaT-k hura gorde egiten du datubase baten barruan, fitxategi berezi batzuetan. Horiei itzulpen-memoriak deritze eta bakoitzean esaldi bat eta bere itzulpena edo itzulpenak agertzen dira. Geroago beste testu bat itzultzerakoan antzeko segmenturen bat aurkituz gero, OmegaT-k parekatze lausoen leihotxoan aurkeztuko du itzulpena, antzekotasun ehuneko batekin batera.

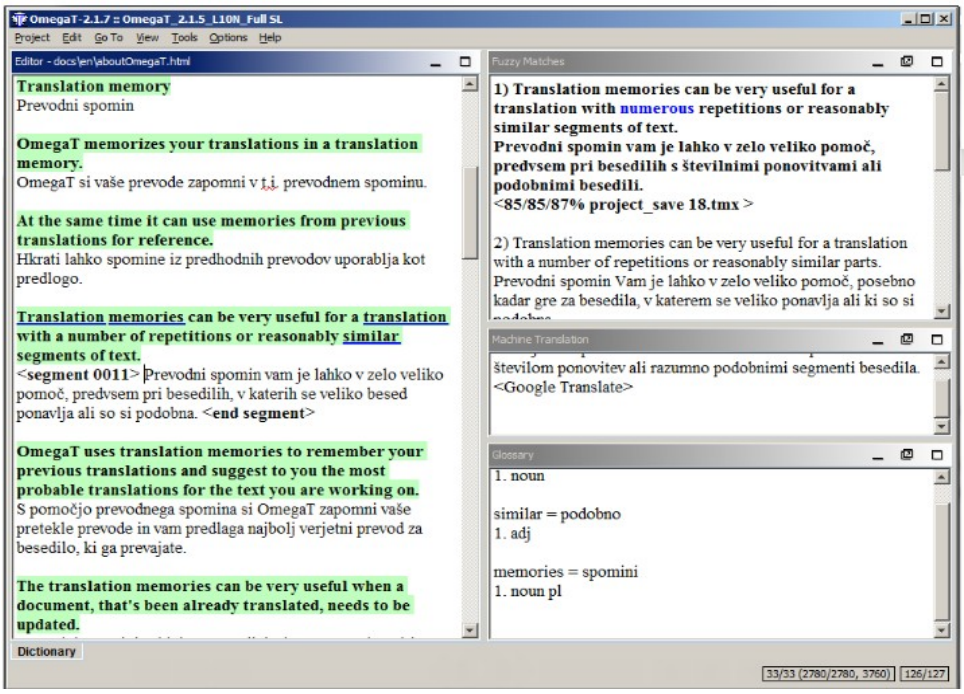
Erabiltzaileak aukera du kanpoko itzulpen-memoriak gehitzeko bere proiektuan, edota proiektu desberdinetakoak biltzea berri bat hastean. TMX formatua erabiltzen du honetarako OmegaT-k eta formatu horretan egon behar dute inportatu edo esportatzen diren itzulpen-memoriak.

Itzulpen automatikoa. Lehen aipatu den bezala, itzulpen automatikoa erreminta oso erabilgarria izan daiteke itzulpenerako laguntzazko aplikazio batean. OmegaT lehenengo aldiz abiaratzean aukera desaktibatuta badago ere, erabiltzaileak itzultzaile automatiko gutxi batzuk dituen zerrenda batetik bat edo gehiago aukera ditzake eta itzultzen ari den segmentuaren proposatutako itzulpen automatikoa agertuko da berehala alboko leiho batean,

beste tresnen antzera. Ctrl + M[*achine translation*] lasterbidearekin uneko segmentuaren itzulpena eskainitako itzulpen automatikoarekin ordeztu ahal izango du erabiltzaileak.

OmegaT-n gure Matxin itzultzailea integratu aurretik, hauek dira OmegaT-k eskaintzen zituen itzultzaile automatikoak: Belazar, errusiera eta bielorrusiera hizkuntza parearen artean itzultzeko ahalmenarekin; Apertium, dozena bat hizkuntza gehiagoren artean itzulpenak egin ditzakeena, batez ere antzeko hizkuntzak diren kasuetan; eta Google Translate, 50 hizkuntzetatik gora dituelarik elkarrekin modu estatistikoan itzultzeko. Tamalez, Google-k bere Google Translate API ordainezko bihurtu zuenetik, hura erabili nahi duen pertsona orok ere ordaindu egin beharko du bere OmegaT-k atzitu ahal izateko.

OmegaT-k aipatu ez den beste aukera interesagarri bat ere eskaintzen du: testuen **post-edizioa**. Aurrerago ikusiko den aurre-edizioan ez bezala, post-edizioan itzulpen automatikoaren emaitza den testu batean aldaketak egiten dira hura hobetzeko. Ondoren, aldaketa horien kudeaketa egoki batekin, informazioa lortu daiteke eta itzultzailean hobekuntzak egiteko erabili.


OmegaT-2.1.7 = OmegaT_2.1.5_L10N_Full SL
Project Edit Go To View Tools Options Help
Editor - docs\en\aboutOmegaT.html
Translation memory
Prevodni spomin
OmegaT memorizes your translations in a translation memory.
OmegaT si vaše prevode zapomni v t.i. prevodnem spominu.
At the same time it can use memories from previous translations for reference.
Hkrati lahko spomine iz predhodnih prevodov uporablja kot predlogo.
Translation memories can be very useful for a translation with a number of repetitions or reasonably similar segments of text.
<segment 0011> Prevodni spomin vam je lahko v zelo veliko pomoč, predvsem pri besedilih, v katerih se veliko besed ponavlja ali so si podobna. <end segment>
OmegaT uses translation memories to remember your previous translations and suggest to you the most probable translations for the text you are working on.
S pomočjo prevodnega spomina si OmegaT zapomni vaše pretekle prevode in vam predlaga najbolj verjetni prevod za besedilo, ki ga prevajate.
The translation memories can be very useful when a document, that's been already translated, needs to be updated.
Fuzzy Matches
1) Translation memories can be very useful for a translation with numerous repetitions or reasonably similar segments of text.
Prevodni spomin vam je lahko v zelo veliko pomoč, predvsem pri besedilih s številnimi ponovitvami ali podobnimi besedili.
<85/85/87% project_save 18.tmx >
2) Translation memories can be very useful for a translation with a number of repetitions or reasonably similar parts.
Prevodni spomin Vam je lahko v zelo veliko pomoč, posebno kadar gre za besedila, v katerem se veliko ponavlja ali ki so si
Machine Translation
številom ponovitev ali razumno podobnimi segmenti besedila.
<Google Translate>
Glossary
1. noun
similar = podobno
1. adj
memories = spomini
1. noun pl
Dictionary
33/33 (2780/2780, 3760)
126/127

***4. Irudia**: OmegaT-ren interfazea*

3.4 Itzulpen automatikoaren ebaluazioa: Asiya

Itzulpen automatikoa egiteko modu ugari daude beraz, eta dagoen tresna kopurua urria ez da. Aukera dago ebaluazioa gizakiek egitea: azken finean, lengoaia naturala erabili eta benetan ulertzen dute; beraz, ziurrenik ebaluazio onargarriena gai honetan adituek emango dute. Hala eta guztiz ere, ez da beti posible hala izatea. Hainbatetan baliteke itzultzaile automatiko baten programatzaileak aldaketa txiki batzuk egiteko asmoa izatea eta horien eragina jakin nahi izatea. Honelako kasu batean, hobe da modu automatikoan egikaritzen den itzulpen automatikoaren ebaluazioa edukitzea, nahiz eta lan erraza ez izan.

Asiya[15] softwarea itzulpen automatikoa ebaluatzeko sortua da, nahiz eta aurrerago azalduko den bezala, beste funtzio batzuk, meta-ebaluazioa bezala, ere badituen (Giménez et al, 2010). Horretarako, probazko dokumentu jakin batzuk beharko ditu: dokumentu originala, jatorrizko hizkuntzan; itzulpen automatikoaren bitartez itzulitako dokumentua; eta pertsonek itzulitako dokumentu bera, erreferentzia bezala erabiltzen dena. Universidad Politècnica de Catalunya (UPC)-ko TALP inbestigazio zentruko Lengoaia Naturalaren Prozesamenduaren taldeak sortu eta mantentzen du Asiya.

Itzulpen automatikoaren ebaluazioa egiteko **metrikak** deritzenak sortu dira. Metrika batek itzulitako testu baten kalitatea neurtu beharko du. Horretarako gizakiontzat baino balio ez duten balio subjektiboak erabili beharko direnez, metrika bakoitzak zer aztertzen duen azaldu beharko du giza iritzien arabera. Gainera, pertsonentzat baliagarria izan behar du emaitzak. Hortaz, esan daiteke metrika on batek emandako emaitzek gizakiok emango genukeenarekin bat datozela.

15 http://nlp.lsi.upc.edu/asiya/#

Metrika automatiko batek bost atributu izan beharko lituzkeela diote Banerjee eta Lavie egileek (Banerjee et al, 2005): giza iritziarekin korrelazio ona, itzulpen automatiko desberdinei sentikorra izatea, kontsistentea emaitza bera lortzeko datu berdinekin, fidagarritasuna antzeko itzultzaileek antzeko puntuazioak lortuko dituztengan, eta corpus domeinu desberdinekin ondo ibili behar da.

Hemen Asiyak erabiltzen dituen metrika edo algoritmo batzuk aurkezten dira, batez ere proiektuan zehar kontuan izan direnak.

BLEU (BiLingual Evaluation Understudy). Sortutako metrika lehenengotarikoa izan zen. BLEU-k itzulpenaren zehaztasuna neurtzen saiatzen da, erreferentziazko testuarekin alderatuz. Horretarako, itzulpenean agertzen den hitz bakoitzeko, erreferentzian hitz bikote edo hitz hirukote bakoitzeko zenbatetan azaltzen den begiratzen du, labur esanda. 0 eta 1 tarteko zenbakia da emaitza. Gero eta altuagoa izan, hitz hitz-bikote edo hitz-hirukote asko azaltzen badira bietan, erreferentzian eta itzulpenean.

NIST. BLEU-ren aldaera. N-grama guztiei pisu berdina eman beharrean, n-grama zuzen bakoitzari pisu bat ematen zaio, balio altuagoa izango duelarik geroz eta urriagoa bada testuan.

GTM (General Text Matcher). Itzulpenaren zehaztasuna eta estalduraren (Precision and Recall) arteko batazbesteko harmonikoa erabiltzen du. Parametro batekin, parekatze luzeak zenbatean "saritzen" diren ezarri daiteke, GTM-ren hiru aldaera sortuz: GTM-10, GTM-20 eta GTM-30.

METEOR (Metric for Evaluation of Translation with Explicit Ordering). GTM-k bezala, batazbesteko harmonikoa erabiltzen du, baina, estaldurari zehaztasuna baino garrantzi handiagoa emanik. BLEU eta antzekoek ez bezala, metrika gutxiek dituzten ezaugarriak ditu: erroen bilaketa, sinonimoak, lerro parekatze zehatzak, etab. Gizakien iritzietara oso hurbil

dagoela kontsideratzen da meta-ebaluazioan.

Lexical Overlap. Itzulpena eta erreferentzia bakoitza bere kabuz tratatzen da. Ondoren, bi multzo lexikalen artean gainjartzea aztertzen da eta horren arabera lortuko da ebaluazioaren emaitza.

WER (Word Error Rate). Distantzian oinarritzen den metrika da. Egin beharreko aldaketetan eta horien distantziak aztertzen ditu – ordezkapenak, txertaketak, ezabatzeak. 0 eta 1 tarteko emaitza itzuliko du, baino "baxuago = hobeto" jarraituz. Hortaz, Asiya-n balio negatiboak ematen zaizkio beste metriken antzera "altuago = hobeto" betetzeko.

PER (Position-independent word Error Rate). WER-en gaineko aldaketa, hitzen arteko ordena kontuan hartzen ez duena.

TER (Translation Edit Rate) eta **TERp** (TER plus). Post-edizioan zenbat aldaketa egin beharko ziren neurtzen du, aldaketa orori balio bera emanez. TERp-ek zabaldu egiten du analisi sintaktikoaren emaitzak erabiliz. Moldaketa desberdinak ditu, hala nola erroen bilaketa, sinonimoak eta parafreseatzea mantenduz edo ez, etabar.

SP (Shallow Parsing). Metrika honek antzekotasun sintaktikoa aztertzen du. Horretarako, kanpoko analizatzaile batek testua aztertu beharko du eta hainbat ezaugarri lortu: hitzen kategoria gramatikalak, hitzen lemak eta hitz multzoak, esate baterako. Horietan oinarrituta ebaluatzen du itzulpen automatikoa ona den.

DP (Dependency Parsing). *Shallow Parsing*-aren antzera, *Dependency Parsing*-ak testuak sintaktikoki aztertuko ditu, baina beste egitura bat erabiliz: dependentzia zuhaitzak. Aztertzen diren dependentzia zuhaitzetan, hitz bakoitzaren kategoria gramatikala eta beste hitzekin duen erlazio sintaktikoa ageri dira. Horren arabera, atal desberdinak aztertu daitezke: erroan dauden hitzen arteko konparazioa, maila jakin batean dauden hitzen

analisia, nodoak edo hostoak aztertu, sintagma mota jakin batera murriztu, etab.

Gainera, horien arteko metriken konbinazioak ere eskaintzen dira. Adibidez, *SP-NIST*-ek, bere izenak dioen bezala, SP eta NIST metrikak biltzen ditu. Nahi bezain konbinazio konplexuak egin daitezke analisietan. Ikusi denez, metrika hauek batzuek hitzen arteko distantzia eta desberdintasunak aztertzen dituzte, eta beste batzuk, berriz, hitzen arteko erlazio sintaktikoak behar dituzte. Horietaz gain Asiya-k baditu hainbat metrika hitzen erlazio semantikoak aztertzen ematen dituztenak beren emaitzak.

Asiya-rekin amaitzeko, merezi du eskaintzen dituen beste bi anailisi motak aipatzea, proiektuan zehar landu ez badira ere. Alde batetik, **meta-ebaluazioa** egin dezake, hau da, ebaluatzaile bat ebaluatu. Bestalde, eta amaitzeko, badu aukera, nahiz eta ez oso garatua, **konfidantza-estimazioak** egiteko, ez dutenak erreferentzia-testurik erabiltzen, baizik eta jatorrizko testua eta itzulpenari bestelako kanpo baliabideak gehituz.

Asiya-ren online bertsio bat badago[16], hainbat aukerekin, oso erabilgarria:

***5. Irudia**: Asiya webgunean*

16 http://asiya.lsi.upc.edu/demo/asiya_online.php

3.5 Itzulpen automatikoaren aurre-edizioa: DiSeg

OmegaT-ri esker egikaritu daiteken post-edizioaz gain, posible da ere testu bat aurre-editatzea itzulpen automatiko hobe bat lortzeko. Bestela esanda, testu bat itzulpen automatikorako aurre-editatzea, emaitza hobeak lortzeko asmoarekin bertan aldaketak sortzea da, baina itzulpen automatikoa gertatu aurretik.

Adibidez, itzultzaile automatikoen emaitzak hobeak izan ohi dira esaldi laburragoekin, luzeekin baino. Hura adibide bat baino ez da, kasu batzuetan beteko dena eta besteetan ez, eta itzultzaile automatikoaren motaren arabera desberdin izan daiteke. Edonola, proiektuari begira, aurre-edizioaren adibide bezala hura konpontzeko modu bat bilatu da: esaldien segmentazioa, luzeenak laburtuz eta sinplifikatuz, itzulpena errazteko asmoarekin.

Proiektu honetan **DiSeg** erreminta erabiliko da hori egiteko. Université d´Avignon et des Pays de Vaucluse-eko Laboratoire Informatique d´Avignon-aren TALNE taldeak eta Universitat Institut Pompeu Fabra-ko Universitari de Lingüística Aplicada-ren Iulaterm taldeak sortua da eta gaztelerazko esaldien segmentazioa egin dezake.

4. Baliabideak

4.1 Corpusa

Erabilitako corpusak bi helburu izan ditu: batetik, Asiya programaren moldaketak egin eta gero hauek ondo zebiltzela probak egitea; eta bestetik, Disegekin probak egiteko. Corpusa *NEWStest* izan da, gaztelerazko egunkarietatik lortutako 1.000 esaldi biltzen dituena, bakoitzaren euskararako bi itzulpenekin batera, erreferentzia bezala erabili direnak itzulpen automatikoaren emaitzak ebaluatzeko orduan.

4.2 Matxin

Matxin[17] gaztelania – euskara eta ingelera – euskara itzulpenak egin ditzakeen itzultzaile automatikoa da, EHU/UPVko Donostiako Informatika Fakultateko IXA Taldeak garatutakoa (Alegria et al, 2005). Arauetan oinarritutako itzultzaile automatikoa da. Matxin itzultzaile automatikoa, bere web aplikazioaren bitartez, OmegaT tresnan integratuko da.

17 http://matxin.sourceforge.net/

Gainera, Diseg eta Asiyarekin frogak egiteko ere Matxin erabiliko da aurre-editatutako esaldiak itzuli ditzan.

4.3 Wikipedia

Wikipedia[18] baliabide bezala erabili egin da OmegaT-ren egokitzapena Entziklopedia moduko webgunea da, eduki askekoa eta edozeinek editatu dezakeena. Wikipedia eleanitza denez, hainbat hizkuntzetan dago, baina ez guztietan garapen maila berdinarekin. Horiek izan dira buruan OmegaT egokitzerakoan, hizkuntza batzuetako artikuluak beste batzuetara itzuli direnean. Froga nagusiak Wikipediaren ingelerazko eta gaztelerazko artikuluak euskaratzean zeutzan, ondoren euskal Wikipedian txertatu direnez sortutako artikulu berriak.

4.4 OmegaT

OmegaT[19] aplikazioa itzulpen automatikoa eta bestelako hainbat tresna erabiltzen dituen itzulpenerako laguntzazko softwarea da. Java lengoaiaz idatzia dago eta dohakoa software librea da. Itzultzaileek erreminta arrunt gisako erabili dezaten dago pentsatua, eta itzultzaile profesionalen artean portzentai handi batek erabiltzen du. OpenMT2 proiektuan, alde batetik, OmegaT moldatu da dituen itzultzaile automatikoen artean gaztelania – euskara eta ingelera – euskara itzulpenak egiten dituen Matxin egokitzeko; eta bestetik, Wikipediako artikuluak itzuli eta igotzeko aukera gehitu da. Azkenik, egokitze honen ondoren egindako frogen emaitzak aztertu dira.

18 http://www.wikipedia.org/

19 http://www.omegat.org/en/omegat.html

4.5 Asiya

Asiya[20] kode irekiko erreminta da, itzulpen automatikoaren ebaluazio eta metaebaluazioa egiteko balio duena. Metrika desberdinak erabiltzen ditu emandako itzulpen automatiko bat ebaluatzeko, jatorrizko testua, erreferentziazko baten bat eta metrika horiek erabiliz. OmegaT-ren antzera, Asiya moldatuko da euskara sartzeko metrika batzuetan, IXA Taldeko analizatzaileak eskaintzen duen informazio gehigarriaz baliatuz. Horretaz gain, Asiya erabiliko da ebaluatzaile paperean DiSeg eta aurre-edizioarekin egindako esperimentuetan.

4.6 DiSeg

DiSeg[21] aurre-ediziorako erabili daitekeen segmentatzaile bat da, web aplikazio baten bitartez atzitu daitekeena. Esaldi luze eta konplexuak beste esaldi laburragoetan bana ditzake, itzultzeko errazago gerta daitekeenak. Probak egiteko erabiliko da, corpuseko testu desberdinak banatuz eta horien gainean itzulpen automatikoa aplikatu ondoren, ebaluazioak emaitza hobeak ematen dituen edo ez aztertuz Asiya erabiliz. Esaldi laburragoak era independentean itzulpen automatikoaren emaitzak hobetzea da asmoa.

20 http://nlp.lsi.upc.edu/asiya/

21 http://daniel.iut.univ-metz.fr/~iula/WebDiSeg/index.php

5. OmegaTren egokitzapena

5.1 Sarrera

Proiektuaren lehenengo fase honen helburua bikoitza da. Alde batetik, OmegaT-ren itzultzaile automatikoen zerrendari IXA taldeko Matxin gehitzea, euskararako itzulpenak egin ahal izateko. Bestetik, OpenMT2 proiektuarekin lotuta, ingelerazko edo gaztelaniazko Wikipedietatik artikuluak eskuratu, euskaratu eta euskarazko Wikipediara berriz igotzea, hori dena OmegaT erabiliz, egin ahal izatea. Horri edozein hizkuntzatako Wikipediekin egin ahal izateko ahalmena gehitu egin zaio.

Aipatzekoa da egindako aldaketa guztiak OmegaT trensaren banaketako azpikarpeta jakin batzuetan bakarrik egin direla, *org.omegat.core*, *org.omegat.gui* eta *org.omegat.util* azpian, hain zuzen ere. Horietan daude OmegaT-k erabiltzen dituen kodeak, eta proiektuan zehar hiruretan zehar murgildu behar izan da.

5.2 Itzultzaile automatikoak OmegaT-n

OmegaT-ren kodearen barnean itzulpen automatikoko modulua dugu. Proiektua hasi zenean Google Translate itzultzaile automatikoa baino ez zuen inplementatua, laister Belazar ere gehituko bazen ere. Google Translate-rekin itzulpen automatikoa egitea ahalbidetzen zuen kodea *omegat.gui.exttrans* moduluan zegoen, *GoogleTranslateTextArea.java*-n, hain zuzen ere. Kode zati horrek zati guztiak zituen: erabiltzaile interfazean nola egikaritzen zen, segmentuen bilaketa eta beste moduluekin komunikatzea, Google Translate-ren API-arekin konexioa egitea, testua bidali, erantzuna jaso, hura moldatu, etab.

5.3 Matxin eta SOAP

Lan hau egikaritu ahal izateko, Matxin itzultzaile automatikoa erabili behar da OmegaT aplikaziotik. Baina Matxin osoa sartu ordez, hobeto zen beste itzultzaile automatikoen modura web aplikazio bat erabiltzea. Honela, OmegaT-ren pisua ez zen handituko eta erabiltzaileak Internet konexioa baizik ez zuen beharko. Hortaz gain, Matxin web aplikazio bitartez atzigarri edukitzea beste proiektu eta helburuetarako erabilpenak ere izan ahalko ditu.

Matxin web API bat bezala jartzeko SOAP (*Simple Object Access Protocol*) erabili da. SOAP-ek XML-z idatzitako mezuak elkar trukatu ditzake HTTP protokoloa erabiliz, web zerbitzuak sortu ahal izanik. SOAP-ekin egitea zerbitzu sinple eta erabiltzeko erraza eskaintzea ahalbidetzen du, edozein programazio-lengoaiarekin erabili daitekena.

Beraz, web-zerbitzura konektatu eta eskaera zehatz eta sinple batekin, Matxin-ek eskainitako itzulpena eskuratu daiteke pare bat segundutan.

5.4 Matxin integratu: Lehenengo saiakera

Matxin integratzeko egin zen lehenengo saiakera aipaturiko *GoogleTranslateText-Area.java*-n oinarrituz egin zen. Ideia kodea kopiatzea zen, eta horren gainean lan egitea, beharrezko aldaketak eginez Google Translate-ri deia egin ordez Matxin-i egiteko.

Google Translate-ren kasuan ez bezala, Matxin-ekin lan egiteko web zerbitzua sortu beharko zen eta web dei bat egitea ez zen nahikoa. Web zerbitzua sortu ahal izateko Apache Axis erabili zen, SOAP-ren Java eta C++-erako inplementazioa. Honekin Matxin API-a erabiliko zen.

Lana era motelean egin zen, lehenengo fasean baino ez baitzegoen, eta ikasketa prozesua hasiberria zegoen. Kodea ez zen guztiz berridatzi eta arazo ugari zeuden oraindik alde batera utzi zenean OmegaT-ren bertsio berria agertzean. Hala ere, laguntza handikoa izan zen ikasketa prozesuan: Java programazio lengoaiarekin lehen jorratzea izan zen; Apache Axis, SOAP eta web zerbitzuak zer eta zertarako ziren ikasi zen; eta bai Google Translate-ri deitzen zion kodea eta OmegaT-ren orokortasun asko barneratu egin ziren, aurrerago erabili egin zirenak.

OmegaT-ren bertsio berri bat argitaratu zen. Itzulpen automatikoari dagozkion berrikuntza oso erabilgarriak zekartzan, lana asko erraztuko zuena, eta modu berrira jo egin zen – ordurarte egindako kodea alde batera guztiz utzi gabe, bertan elementu interesgarriak ere baitzeuden.

5.5 Itzultzaile automatikoen eguneraketa

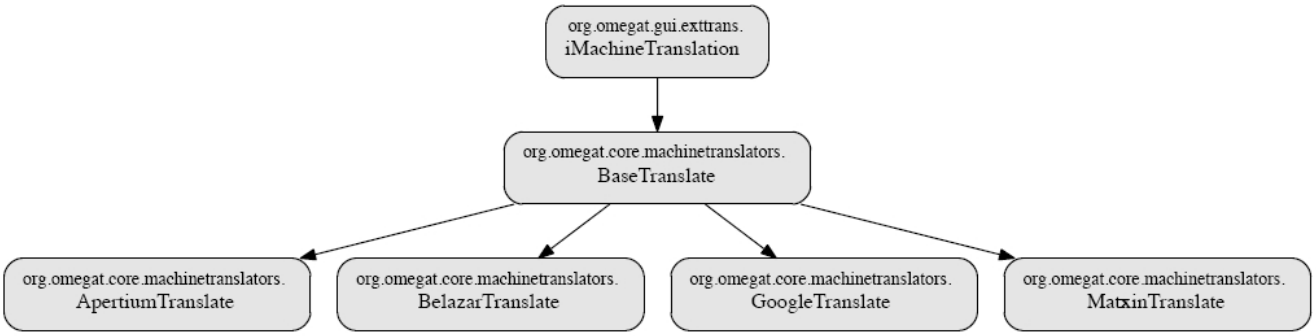
Hasieran OmegaT-ren itzultzaile automatikoen artean Google Translate soilik bazegoen ere, aplikazio honen arrakasta eta erabilgarritasuna direla eta, ez da arraroa suertatzen itzultzaile automatiko gehiago sartzea denbora pasa ahala. Aipatu denez, hurrengoa Belazar izan zen. Eta ondorengoa, Apertium. Apertium-ekin batera, itzultzaile automatikoei dei egiten dien kodean hainbat aldaketa egin ziren, kodea zatitan banatuz, modularitatea erraztuz eta itzultzaile automatiko berrien txertaketari lagunduz.

Alde batetik, *omegat.gui.exttrans* moduluan hiru fitxategi gelditzen dira. Horietako batek, *MachineTranslateTextArea.java*-k, interfazearekin zerikusia duen guztia mantentzen du, lehen *GoogleTranslateTextArea.java*-n zegoenaren antzera, baina nahastu gabe. Beste batek, *MachineTranslationInfo.java*-k, aktibatutako itzultzaile automatikoak zeintzuk diren begiratuko du. Eta hirugarrena, *IMachineTranslation.java*, beste leku batean jarritako azpiklaseek hau erabili dezaten.

Bestalde, itzulpen automatikoari berari dagokion kodea *omegat.core.machinetranslators*-era mugitu da, eta superklase bat ezarri da. *BaseTranslate.java*-k *IMachineTranslation.java*-ren azpiklasea da, hainbat eragiketa komun egingo dituenak: testua hartu uneko segmentutik, erabiltzaile interfazeari zer erakutsi adierazi, erroreren bat badago abisatu, etab.

Beraz, *BaseTranslate.java* gainkargatzen duen objektu bat gelditzen da itzultzaile automatiko bakoitzeko: *GoogleTranslate.java*, *ApertiumTranslate.java* eta *BelazarTranslate.java*. Horietako bakoitzak ez du bere itzultzaile automatikora deia egin eta emaitza jaso baino egin behar.

Honelakoa da, beraz, orain, OmegaT-ren itzultzaile automatikoen egoera:



***6. Irudia**: OmegaT-ren itzultzaileen arkitektura*

5.6 Matxin integratu: Bigarren saiakera

Guzti hau jakinda, Matxin integratzeko bigarren saiakera egikaritu ahal izan da, kasu honetan arrakastarekin. Hasieran egindako kodea eta itzultzaile berriak kontuan hartuta, *MatxinTranslate.java* idaztea posible eginz en.

Hau izan zen kode amaituaren lehen bertsioa. Beltzez, Matxin-i deia.

```
protected String translate(Language sLang, Language tLang, String text)
     throws Exception {
        try {
               String endpoint = "http://siuc05.si.ehu.es/cgi-bin/matxin/translate.cgi";
               String uri = "http://siuc05.si.ehu.es/MATXIN";
               Service service = new Service();
               Call call = (Call) service.createCall();
               call.setTargetEndpointAddress( new java.net.URL(endpoint));
               call.setOperationName( new QName(uri, "EsEu") );
               String ret = (String) call.invoke( new Object[] {text});
            return ret;
        }
        ...        }
}
```

***7. Irudia**: MatxinTranslate.translate, lehenengo bertsioa*

Aurrerago kodearen bertsio gehiago egin badira ere, ikusiko den bezala, hau zen kodearen itxura lehenengo aldiz martxan jartzerakoan arrakastarekin. Kodea sinplea eta ulerterraza da. Zerbitzuaren URL eta URI-a definitzen dira hasteko. Ondoren, Ajax-ek eskaintzen dituen erremintak erabiliz, web zerbitzua definitzen da. Behin konexioa eginda jatorrizko testuaren hizkuntza eta lortu nahi den itzulpenarena definitzen dira – kasu honetan, gazteleratik euskarara soilik planteatu egin zen. Azkenik, deia egiten da itzuli nahi den testua, parametro bezala jaso dena, bidaliz; eta emaitza *ret* izeneko parametroan gordetzen da.

Kode sinple honekin Matxin erabiliz itzulpenak egitea posible zen iada. Hala ere, bestelako aldaketak egin behar izan dira programaren beste lekuetan. Esaterako, *org-.omegat.util.Preferences.java*-n beste itzultzaile automatikoen antzera, Matxin ezartzeko aukera ezarri behar izan zen; eta *org.omegat.Bundle.properties*, *org.omegat.Bundle_es.properties* eta *org.omegat.Bundle_eu.properties* fitxategietan botoi berriei izenak eman zitzaien.

5.7 Matxin integratu: Azken bertsioa

Aurrerago bestelako aldaketak egin behar izan dira. Alde batetik, hizkuntza eta aukerak gehitzeko eta errazago maneiatzeko. Bestalde, Matxin-en API-a ere aldatu egin delako. Gorka lankidearen lanari esker, hona hemen nola gelditu den kodea:

```
protected String translate(Language sLang, Language tLang, String text)
throws Exception {
   try {
        String sourcetargetlang = changeCode(sLang)+changeCode(tLang);
        String endpoint = "http://ixa2.si.ehu.es/matxin_zerb/translate.cgi";
        String uri = "MATXIN";
        String s = text + "\n.é";
        byte[] b = s.getBytes("UTF-8");
```

```
        Service service = new Service();
        Call call = (Call) service.createCall();
        call.setTargetEndpointAddress(endpoint);
        call.setOperationName(new QName(uri, sourcetargetlang));
        call.addParameter("c-gensym3", org.apache.axis.Constants.XSD_BASE64,
        javax.xml.rpc.ParameterMode.IN);
        call.addParameter("c-gensym4", org.apache.axis.Constants.XSD_STRING,
        javax.xml.rpc.ParameterMode.IN);
        call.setReturnType(org.apache.axis.Constants.XSD_BASE64);
        ...
```

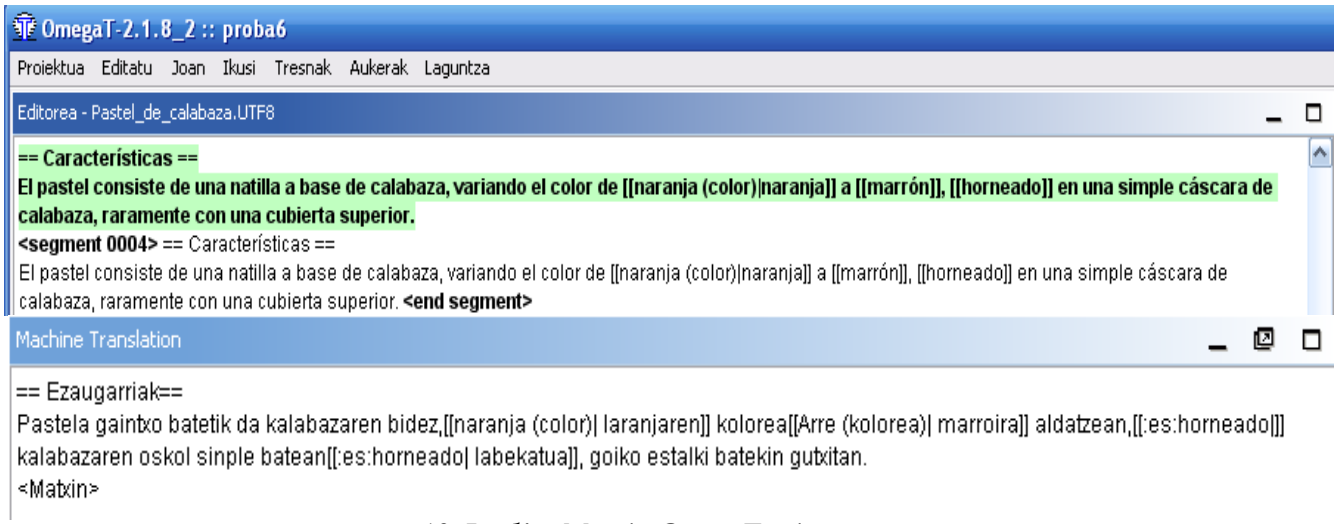
***8. Irudia**: MatxinTranslate.translate, bigarren bertsioa, definizioak*

Kodearen lehenengo zati honetan ageri da nola definitzen diren zerbitzua eta erabiliko dituen parametroak Axis eta SOAP-ek eskaintzen dituzten funtzioak erabiliz. Gehitu egin diren hainbat elementu web zerbitzuaren aldaketak direla eta sartu dira, eta beste batzuk, kodeketarekin eta testuaren formatuarekin zerikusia dutenak – arazoak sortzen dira UTF-8, BASE64, etabarren arteko bihurketak ez badira ongi azaltzen.

Bigarren zati honetan deia egikaritzen da. Beltzez, deia:

```
        ...
        byte[] b2;
        if (sLang.getLanguageCode().equals("ES") &&
        tLang.getLanguageCode().equals("EU")) {
                b2 = (byte[]) call.invoke(new Object[]{b, "wiki_es"});
        }
        else { b2 = (byte[]) call.invoke(new Object[]{b, "wiki"}); }
        String str = new String(b2, "UTF8");
        String tr = str.substring(0, str.length()-5);
        return tr; }
    catch (Exception e) {System.err.println(e.toString());
    return e.toString();
    }
}
```

***9. Irudia**: MatxinTranslate.translate, bigarren bertsioa, deia*


OmegaT-2.1.8_2 :: proba6
Proiektua Editatu Joan Ikusi Tresnak Aukerak Laguntza
Editorea - Pastel_de_calabaza.UTF8
== Características ==
El pastel consiste de una natilla a base de calabaza, variando el color de [[naranja (color)|naranja]] a [[marrón]], [[horneado]] en una simple cáscara de calabaza, raramente con una cubierta superior.
<segment 0004> == Características ==
El pastel consiste de una natilla a base de calabaza, variando el color de [[naranja (color)|naranja]] a [[marrón]], [[horneado]] en una simple cáscara de calabaza, raramente con una cubierta superior. <end segment>
Machine Translation
== Ezaugarriak==
Pastela gaintxo batetik da kalabazaren bidez,[[naranja (color)| laranjaren]] kolorea[[Arre (kolorea)| marroira]] aldatzean,[[:es:horneado|]] kalabazaren oskol sinple batean[[:es:horneado| labekatua]], goiko estalki batekin gutxitan.
<Matxin>

***10. Irudia**: Matxin OmegaT-n integratua*

Emaitza: posible bihurtzen da Matxin erabiltzea itzultzaile automatiko moduan. Menuan Matxin aktibatzea aukeratu eta gero, nahikoa da *Machine Translation* leihoa irekita edukitzea eta itzulpenak bere kabuz agertuko dira. Gainera, OmegaT-k eskaintzen dituen aukerei esker, Ctrl + M lasterbideak defektuz duen efektua leiho hortako testua uneko segmentuan jartzea da, prozesua azkartuz.

5.8 WikiGet

Matxin alde batera utzita, proiektuan OmegaT-rekin erlazionatuta dagoen beste helburua Wikipediarekin zerikusia duena da. Zorte onez, hau bazegoen lehenagotik inplementatua, *org.omegat.util* barneko *WikiGet.java* izeneko kodean. Labur azalduta, *WikiGet*-en lana, erabiltzailearen interfazean dauden beste funtzioetaz baliatuz, MediaWiki formatuko testuak jasotzea da, OmegaT-n erabili daitezen. Horretarako, erabiltzaileak Wikipediako artikulu baten URL baino ez du sartu behar. *WikiGet*-ek, Wikipediak berak eskaintzen dituen API-ak erabiliz, Wikipediatik zuzenean lortzen du artikulua eta testua moldatzen du OmegaT-n ongi azal dadin (kodeketa, etab.).

Beraz, proiektua hasi zenetik zuen OmegaT-k MediaWiki formatuko testuak eskuratu eta editatzeko. Baina ez berriz sarera igotzeko. Hori, beraz, hurrengo pausoetan egin behar zen.

5.9 MediaWiki API-a

Izen hau duen web aplikazioak Wikipediako artikuluak tratatzeko balio duen zerbitzu bat da. Software aplikazioen bitartez Wikipedian dagoen maila altuko informazioa, MediaWiki formatuan, atzitzeko aukera ematen du. API-ari deitzerakoan hainbat parametro definitu daitezke, testuaren formatua eta akzio mota bezala. Azkeneko hau da garrantzitsuena, dozenaka aukera desberdin eskaintzen baititu Wikipediarekin "jolasteko".

Eskaintzen diren funtzio anitz horietatik, gure *WikiSave* sortzeko garrantzitsuenak: Wikipediara konektatu eta erabiltzaile identifikatu bezala kautotzea (*action* = *login*), *cookie*-ak lortu eta software gaiztoa edo malwarea ez dela egiaztatatu (*action* = *query*) eta azkenik artikulu egokian aldaketa egitea (*action* = *edit*).

WikiGet-ek badu post izeneko funtzio bat dei orokorra egiteko erabiltzen duena, eta ondoren baliozkoa izango dena *WikiSave*, Wikipedian gordetzeko kodea, idazterakoan.

5.10 WikiSave

Dagoeneko Wikipediako artikuluak, hau da, MediaWiki formatukoak, lortu daitezkenez, hurrengo pausua hauek editatu eta gero berriz Wikipediara –hizkuntza berrian–

igo ahal izatea izango da. Horretarako, MediaWiki API-ko funtzio sakonagoak erabili beharko dira, baina *WikiGet*-en oinarritu da kodea hala ere.

Matxin integratzeko beste kode zati gehiago moldatu behar izan dira *WikiSave* inplementatzeko, *org.omegat.util.WikiSave.java* sortzeaz gain. Aurreko kasuan bezala, *org.-omegat.Bundle.properties*, *org.omegat.Bundle_es.properties* eta *org.omegat.Bundle_eu.properties* aldatu behar izan dira Wikipedian gordetzeko aukera eskaintzen duten aukerak sartzeko. "Wikipedian gorde" gehitzeko, gainera, beste fitxategi batzuk editatzea beharreazkoa suertatu da: *org.omegat.gui.filelist.ProjectFrame.java*, *org.omegat.gui.main.MainWindow.java*, (zeinetan botoiaren funtzioa definitu behar izan den) *org.omegat.gui.main-MainWindowMenu.java* eta *org.omegat.gui.main.MainWindowMenuHandler.java*.

Beste alde batetik, *WikiGet* editatu egin zen funtzio berri bat gehituz. *WikiGet.post2* funtzioa, *WikiGet.post* jatorrizkoaren berdina egiten du (hau da, MediaWiki API-ari dei bat), baina parametro desberdinak gehitu zaizkio, *WikiSave*-k behar desberdinak baititu eta funtzio desberdinak erabiliko ditu, eta *WikiGet.post* ez dago horretarako prest. Beraz, labur esanda. *WikiGet.post2*-k parametro bezala helbidearentzat *String* bat eta <*String, String*> hash mapa bat izan beharrean, helbidearen *String*-a, <*String, Object*> hash mapa bat eta *cookie*-ekin beste *String* bat eskatuko ditu (*cookie*-ak beharrezkoak baitira Wikipedian aldaketak sorrarazteko).

MainWindow-en, "Wikipedian gorde" botoiak zer egiten duen definitzen da, hau da, *doWikiSave()*-en, *WikiSave*-ren funtzio desberdinei deiak egongo dira. Lehenengo, *WikiSave*-k dituen bost funtzioak azalduko dira banan-bana eta amaitzeko *doWikiSave()*-ren egitura azalduko da.

Hasteko, *getName* funtzioa uneko fitxategiaren (momentuan itzultzen ari dena) izena itzultzen du.

```
public static String getName(String in) {
    String[] split = in.split
    (Pattern.quote(System.getProperty("file.separator")));
    String fileName = split[split.length - 1];
    String name = fileName.substring(0, fileName.length() - 5);
    return name;
}
```

***11. Irudia**: WikiSave.getName*

Jarraian, *getTarget*-ek itzulpen-fitxategiaren izena ezagutuz , fitxategiaren edukia, hau da, itzulpena testu formatuan, itzuliko du. UTF8 motako fitxategia dela egiaztatuko du lehenengo. Gero, formatua egokia izan dadin hainbat eragiketa eman dizkiogu.

```
public static String getTarget(String locRoot, String midName) {
    String translation = new String();
    try {
        String[] split = midName.split (Pattern.quote(System.getProperty("file.separator")));
        String fileName = split[split.length - 1];
        String extension = fileName.substring(fileName.length() - 5);
        if (extension.equals(".UTF8")) {
            File file = new File(locRoot, fileName);
            StringBuilder contents = new StringBuilder();
            BufferedReader reader = new BufferedReader(new FileReader(file));
            String text = null;
            while ((text = reader.readLine()) != null) {
                String UTFtext = new String(text.getBytes(), "UTF-8");
                contents.append(UTFtext).append (System.getProperty("line.separator"));
            }
            if (reader == null) { reader.close(); }
            translation = contents.toString();
            return translation;
        }
        return translation;
    } catch (Exception e) { ... }
}
```

***12. Irudia**: WikiSave.getTarget*

Hurrengoa *connect* da. Honek Wikipediari dei bat egingo dio *login* akzioa adieraziz, eta horrekin batera, erabiltzaile izena eta pasahitza, kautotzeko. OmegaT-ren erabiltzailea arduratu beharko da kontu erabilgarri bat edukitzeaz hemen sartzeko. Behin *login*-a eginda, MediaWiki API-ak *cookie* bat itzultzen du sesioaren informazioarekin. *Cookie* hau garrantzitsua da, Wikipedian aldaketak egin nahi direnean bot baten moduan, saioa hasita dagoenaren *cookie*-a erakutsi beharko baita.

Jarrian, kodea. Beltzez adierazi da MediaWiki API-aren erabilera.

```
public static String connect(String lang, String userName, String password) {
	String address = "http://" + lang.toLowerCase() + ".wikipedia.org/w/api.php";
	String Cookies = "";
	try {
		Map<String, Object> map_login = new HashMap<String, Object>();
		map_login.put("action", "login");
		map_login.put("lgname", userName);
		map_login.put("lgpassword", password);
		map_login.put("format", "xml");
		String res1 = WikiGet.post2(address, map_login, "");
		Pattern p = Pattern.compile("result=(\'|\")NeedToken(\'|\").*token=(\'|\")([^\"]+)
			(\'|\").*cookieprefix=(\'|\")([^\"]+)(\'|\").*sessionid=(\'|\")([^\"]+)(\'|\")");
		Matcher m = p.matcher(res1);
		if (m.find()) {
			map_login.put("lgtoken", m.group(4));
			Cookies = m.group(7) + "_session=" + m.group(10);
			res1 = WikiGet.post2(address, map_login, Cookies);
		}
		p = Pattern.compile("result=(\'|\")Success(\'|\").*lguserid=(\'|\")([^\"]+)(\'|\")
			.*lgusername=(\'|\")([^\"]+)(\'|\").*lgtoken=(\'|\")([^\"]+) (\'|\")
			.*cookieprefix=(\'|\")([^\"]+)(\'|\").*sessionid=(\'|\")([^\"]+) (\'|\")");
		m = p.matcher(res1);
		if (m.find()) {
			Cookies = m.group(13) + "_session=" + m.group(16);
			Cookies += "; " + m.group(13) + "UserName=" + m.group(7);
			Cookies += "; " + m.group(13) + "UserId=" + m.group(4);
			Cookies += "; " + m.group(13) + "Token=" + m.group(10);
```

```
            } else {
                    Cookies = "";
                    String message = StaticUtils.format(OStrings.getString ("TF_WIKI_USER_FAILED"),
                            new Object[]{userName});
                    JOptionPane.showMessageDialog(Core.getMainWindow().getApplicationFrame(),
                            message, Ostrings.getString("TF_WIKI_SAVE_TITLE"),
                            JOptionPane.INFORMATION_MESSAGE);
            }
            } catch (Exception e) {
                    System.out.println("error: " + e.toString());
            }
       return Cookies;
}
```

***13. Irudia**: WikiSave.connect*

Laugarrena *doWikiEdit* da. Honek Wikipedian aldaketa egikarituko du MediaWiki API-a erabiliz. Lehenengo *query* akzio bat egikaritzea beharrezkoa da, *cookie*-az gain pauso hau eman behar baita editatzeko baimena eskuratzeko eta malwarea ez dela egiaztatzeko. Behin abisu hori emanda, *edit* akzioa egiteko prest dago.

Kode hau ere, beraz, aurkezten da jarraian. Berriz ere, *post* funzioak beltzez markatu egin dira.

```
   public static void doWikiEdit(String name, String translation, String lang, String Cookies) {
       try {
           String address = "http://" + lang.toLowerCase() + ".wikipedia.org/w/api.php";
           String token = "";
           String message = StaticUtils.format
              (OStrings.getString("TF_WIKI_SAVE_FAILED"), new Object[]{name});
           translation = "{{OpenMT-2}}\n" + translation;
           translation.getBytes(OConsts.UTF8);

           Map<String, Object> map_login = new HashMap<String, Object>();
           map_login.put("action", "query");
           map_login.put("intoken", "edit");
```

```
            map_login.put("titles", name);
            map_login.put("prop", "info");
            map_login.put("format", "xml");
            String res1 = WikiGet.post2(address, map_login, Cookies);

            Pattern p = Pattern.compile("edittoken=(\'|\")([^\"]+)(\'|\")");
            Matcher m = p.matcher(res1);
            if (m.find()) {
                token = m.group(2);
            }

            Map<String, Object> map = new HashMap<String, Object>();
            map.put("action", "edit");
            map.put("token", token);
            map.put("title", name);
            map.put("text", translation);
            map.put("format", "xml");
            String res = WikiGet.post2(address, map, Cookies);

            p = Pattern.compile("result=(\'|\")Success(\'|\")");
            m = p.matcher(res);

            if (m.find()) {
                message = StaticUtils.format(OStrings.getString("TF_WIKI_SAVE_SUCCESS"),
                      new Object[]{name});
            }
            JOptionPane.showMessageDialog(Core.getMainWindow().getApplicationFrame(), message,
                    OStrings.getString("TF_WIKI_SAVE_TITLE"), JOptionPane.INFORMATION_MESSAGE);

            return;
        } catch (Exception e) {
            System.out.println("error: " + e.toString());
        }
    }
```

***14. Irudia**: WikiSave.doWikiEdit*

Azkeneko funtzioa, *sendTMX*, hurrengo puntuan azalduko da, Wikipedia editatzearekin zerikusi zuzena ez baitu.

Ondoren, *MainWindow.java*-ko *doWikiEdit* azalduko da. Hasteko, OmegaT-ren uneko proiektua gordetzen eta konpilatzeko agindua ematen da. Honen arrazoia, lehen ikusi den *getTarget*-ek itzulpenaren fitxategi osoa hartu behar duela da, eta hori eguneratuta egoteko, gordetzea eta konpilatzea burutuak behar dute egon. Gero, elkarrizketa leihoekin erabiltzaileak eragiketa egiaztatuko du eta bere Wikipediako erabiltzaile izena eta pasahitza sartuko ditu.

Aipatu den eta azaltzear dagoen *sendTMX* egikaritzen da. Jarraian, *connect* erabiliz *cookie*-a lortu egiten da eta igoko diren fitxategien zerrenda egiten da (proiektu batean fitxategi bat baino gehiago egon baitaiteke, erabilgarritasunaren ikuspuntutik erosoena ez bada ere). Bakoitzarekin *getTarget* eta *getName* egiten dira. Erabiltzaileari fitxategi zehatz hau ere gorde nahi duen galdetzen zaio bada-ezpada ere eta zein izenburuarekin gordeko den galdetzen zaio. Amaitzeko, *doWikiEdit*-i deitzen zaio, artikulua igo egin dadin.

Kodean, beltzez markatu dira *WikiSave*-ri deiak.

```
public void doWikiSave() {
	try {
		Core.getProject().saveProject();
		Core.getProject().compileProject(".*");
	} catch(Exception ex) {
		Log.logErrorRB(ex, "TF_COMPILE_ERROR");
		Core.getMainWindow().displayErrorRB(ex, "TF_COMPILE_ERROR");
		return;
	}
	JTextField userField = new JTextField();
	JPasswordField passwordField = new JPasswordField();
	Object[] Fields = new Object[]{OStrings.getString("TF_WIKI_USER_PROMPT"),Ostrings.getString
		("TF_WIKI_USER"), userField, OStrings.getString("TF_WIKI_PSW"), passwordField};
	 int r = JOptionPane.showConfirmDialog(Core.getMainWindow().getApplicationFrame(), Fields,
		 OStrings.getString("TF_WIKI_USER_TITLE"), JOptionPane.OK_CANCEL_OPTION);
```

```
    if (r == JOptionPane.OK_OPTION) {
            WikiSave.sendTMX(userField.getText(), Core.getProject().getProjectProperties().
                    getProjectName());
            String projectTarget = Core.getProject().getProjectProperties().getTargetRoot();
            String targetLan = Core.getProject().getProjectProperties().getTargetLanguage().
                    getLanguageCode();
            String Cookies = WikiSave.connect(targetLan, userField.getText(),
                    passwordField.getText());

        if ((Cookies != null && !Cookies.equals("")) && (projectTarget != null) &&
            (projectTarget.trim().length() > 0)) {
             List<String> files = new ArrayList<String>(256);
             StaticUtils.buildFileList(files, new File(projectTarget), true);
             if (files.isEmpty()) {
                 return;
             }

             for (String filename : files) {
                 String translation = org.omegat.util.WikiSave.getTarget(projectTarget, filename);
                 if (!translation.isEmpty()) {
                     String sourceName = org.omegat.util.WikiSave.getName(filename);
                     String name = JOptionPane.showInputDialog(this,
                             StaticUtils.format(OStrings.getString("TF_WIKI_SAVE_PROMPT2"),
                                  targetLan.toLowerCase(), sourceName),
                             OStrings.getString("TF_WIKI_SAVE_TITLE"),
                             JOptionPane.OK_CANCEL_OPTION);

                     if (name != null && name.length() > 0) {
                         WikiSave.doWikiEdit(name, translation, targetLan, Cookies);
                     }
                 }
             }
        }
    }
}
```

***15. Irudia**: MainWindow.doWikiSave*

5.11 Post-edizioaz baliatu Matxin-entzat

Amaitzeko, azken funtzio bat eman nahi izan zaio OmegaT-ri. Egindako aldaketei esker, erabiltzaileak Matxin-en itzulpen automatikoetaz baliatu daiteke eta gero horiek post-editatu. Baina OpenMT2 proiektuaren parte bezala, beste erabilpen bat ikusten zaio itzulpen automatikoaren post-edizio honi. Hain zuzen ere, gure IXA Taldeko ordenagailuetara bueltan bidaltzea emaitza hori, post-edizioan egindako aldaketak aztertu eta horiekin Matxin hobetzeko.

Matxin honela hobetzea ez da proiektuaren parte, baina emaitzak bidaltzea, bai. Horretarako, metodo sinple bat ezarri dugu: OmegaT aldatu honekin itzulpen bat Wikipedian gordetzen den bakoitzean, proiektuan une hartaraino OmegaT-k automatikoki sortutako itzulpen memoria (*Translation Memory*, TMX formatukoa) hartu eta bidali egingo da. Hortaz arduartzen da *WikiSave*-ren bostgarren funtzioa:

```
public static void sendTMX(String userName, String projectName) {
    String TMXname = Core.getProject().getProjectProperties().getProjectRoot() +
           Core.getProject().getProjectProperties().getProjectName() + OConsts.OMEGAT_TMX +
           OConsts.TMX_EXTENSION;

    try {
        File TMXfile = new File(TMXname);
        StringBuilder contents = new StringBuilder();
        BufferedReader reader = new BufferedReader(new FileReader(TMXfile));

        String text = null;
        while ((text = reader.readLine()) != null) {
            String UTFtext = new String(text.getBytes(), "UTF-8");
            contents.append(UTFtext).append(System.getProperty("line.separator"));
        }
        if (reader == null) {
            reader.close();
        }
```

```
            String address = "http://ixa2.si.ehu.es/glabaka/OpenMT-OmegaT/uploadfile.php";
            Map<String, Object> map_upload = new HashMap<String, Object>();
            map_upload.put("username", userName);
            map_upload.put("project", projectName);
            map_upload.put("TMXtext", contents.toString());
            String res1 = WikiGet.post2(address, map_upload, "");
            System.out.println("SendTMX: " + res1);
            return;
        } catch (FileNotFoundException e) {
            JOptionPane.showMessageDialog(Core.getMainWindow().getApplicationFrame(),
               Ostrings.getString("TF_WIKI_CREATE_FILES"), OStrings.getString("TF_WIKI_SAVE_TITLE"),
               JOptionPane.INFORMATION_MESSAGE);
        } catch (Exception e) {
            System.out.println("error: " + e.toString());
        }
    }
```

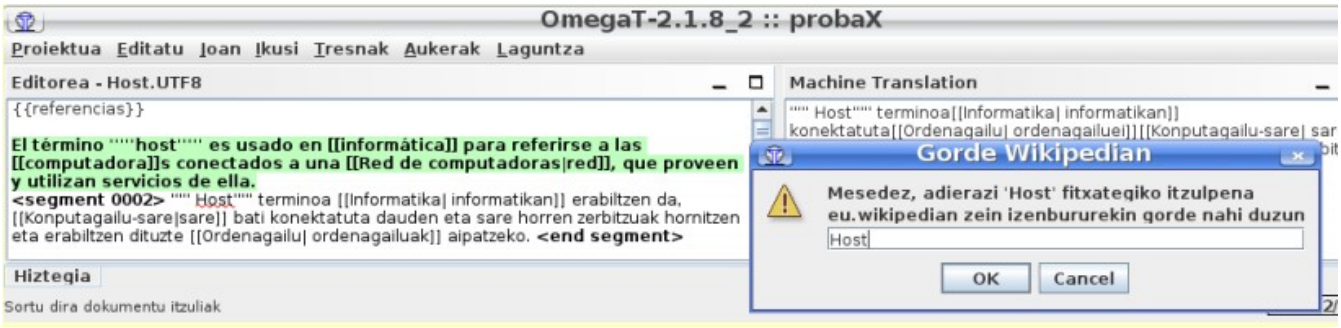
***16. Irudia**: WikiSave.sendTMX*

5.12 Hainbat arazo konpondu

OmegaT-ren egokitzapenean azken pausu bezala, bidean alde batean utzitako hainbat arazo aztertu eta, ahal izan zenean, zuzendu egin ziren. Formatu arazoak (bai dokumentuekin eta baita MediaWiki formatuarekin ere), API-ak erabiltzen, konexioak ezartzen, etab.

Euskal Wikipediako Unai Fernández de Betoño-ren laguntzarekin Wikipediaren funtzionamenduari buruzko hainbat arazo konpondu ziren eta euskararako moldatutako OmegaT erabiltzeko prest geratu zen.

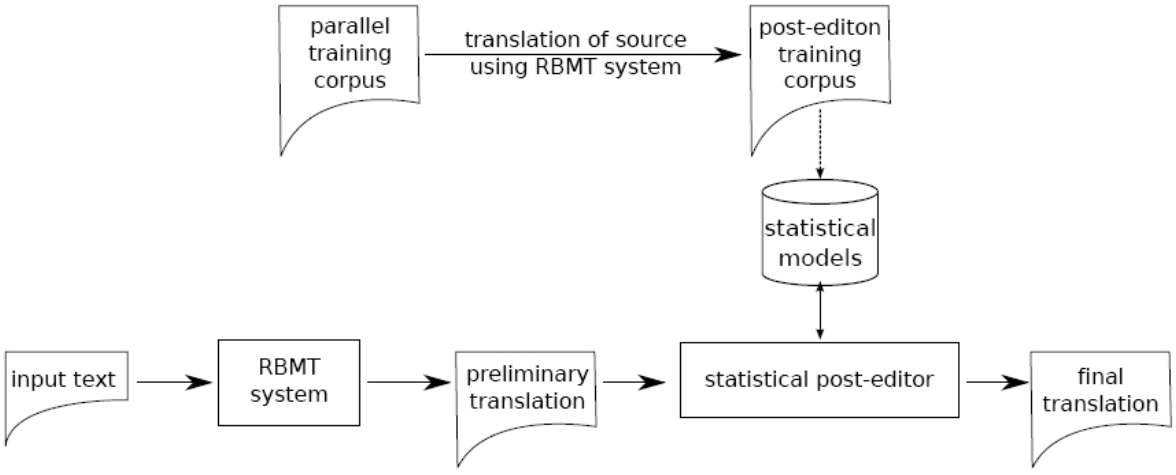
Hala eta guztiz ere, badira hobekuntza posible batzuk, atal egokian azalduko direnak.


OmegaT-2.1.8_2 :: probaX
Proiektua Editatu Joan Ikusi Tresnak Aukerak Laguntza
Editorea - Host.UTF8
{{referencias}}
El término """host""" es usado en [[informática]] para referirse a las [[computadora]]s conectados a una [[Red de computadoras|red]], que proveen y utilizan servicios de ella.
<segment 0002> """ Host""" terminoa [[Informatika| informatikan]] erabiltzen da, [[Konputagailu-sare|sare]] bati konektatuta dauden eta sare horren zerbitzuak hornitzen eta erabiltzen dituzte [[Ordenagailu| ordenagailuak]] aipatzeko. <end segment>
Hizteqia
Sortu dira dokumentu itzuliak
Machine Translation
""" Host""" terminoa[[Informatika| informatikan]]
konektatuta[[Ordenagailu| ordenagailuei]][[Konputagailu-sare| sar
Gorde Wikipedian
Mesedez, adierazi 'Host' fitxategiko itzulpena eu.wikipedian zein izenbururekin gorde nahi duzun
Host
OK
Cancel

***17. Irudia**: Artikulu bat Euskal Wikipediara igotzen OmegaT erabiiliz*

5.13 Erabilera erreala eta emaitzak

2010. urteko azaroan hasi egin zen euskal Wikipediarekin elkarrekiko proiektua, zeinarekin bai Matxin eta baita OmegaT ere proban jarriko ziren. OpenMT2 proiektu honetan, erabiltzaileei informatikaren arloko artikuluak itzul ditzaten eskatzen zaie gaztelaniatik euskarara, OmegaT erabiliz, Matxin-en itzulpen automatikoen gainean post-editatuz eta amaitzeko euskal Wikipediara igoz, prozesuan, azaldu den bezala, sortu den itzulpen memoria guri bidaliz (ikus 18. Irudia). Informatikako artikuluak hiru motetan sailkatu ziren, luzeak, tartekoak eta motzak, erabiltzaileak orientatzeko.



***(aurreko orrialdean) 18. Irudia**: Post-editore estatistiko baten ohiko arkitektura*

Horretarako, eta proiektuaren parte, erabiltzaile-gida bat sortu egin da, OmegaT deskargatu eta instalatzeko argibideekin, OmegaT erabiltzeko azalpen orokorrekin, eta zehatzagoak Matxin eta Wikipediarekin zerikusia duten atalentzat, guretzat interesgarrienak direnak. Gida hori dokumentu honen Eranskin bezala aurkitu daiteke.

Lan boluntario honetan 36 pertsonek hartu dugu parte. 2011ko ekaina eta 2012ko otsailaren artean, biak barne, 100 artikulu itzuli egin dira guztira, 50204 itzulitako hitzetako itzulpen memoriak sortuz. Bolondresek lan aipagarria egin zuten, Matxin-ek oraindik ez baititu kalitate handiko itzulpen automatikoak sortzen, artikulu luzeak zailagoak dira (hala ere erdiak baino gehiago itzuli egin dituzte artikulu luzeak), Wikipediako artikuluen metadatuen formatuak zailak gertatzen dira eta euskara eta informatika hizkuntza eta lexiko minoritarioak direlarik, batez ere.

Hurrengo grafikoan proiektuan parte hartu duen jendearen lana ikusi daiteke, artikuluak itzuli ahala IXA-ra iristen ziren itzulpen-memoria eta hitzen bolumena zein azkar hazten zen argi ageriz.

***19. Irudia**: OpenMT2 proiektuan Wiki-artikuluetan itzulitako hitzen kopurua*

Hala ere, Matxin-en itzulpen berriak aztertuz eta aurrekoekin alderatuz, emaitzak onak eta desiragarriak izan dira. Bolondresak Matxin itzultzaile automatikoarekin eta OmegaT itzulpenerako laguntzazko tresnarekin pozik egon dira eta eskaintzen duten laguntza aipatu egin dute. Gainera, Matxin-i proiektuan zehar sortutako itzulpen memoriak gehitzerakoan eta itzulpen automatikoaren ebaluaziotik pasa ondoren, %10aren inguruko hobekuntza nabaria aurkitu egin da.

6. Asiya-ren egokitzapena

6.1 Sarrera

Asiya itzulpen automatikoak ebaluatzeko gaitasuna duen software librea da. Perl-ez idatzita dago programaren zati handiena, baina erabiltzen dituen hainbat modulu eta programa beste programazio lengoaietan daude.

Ebaluaketak egiteko hainbat metrika erabiltzeko aukera eskaintzen du, eta testu bat, bere itzulpen automatikoa eta erreferentziazko pertsona batek egindako beste itzulpen bat erabiliz, itzulpen automatikoa ebaluatzen dutenak. Bakoitzak parametro desberdinak eta era desberdinean neurtzen ditu, eta konbinatu egin daitezke emaitza kontsistenteak lortu ahal izateko.

Metrika horietako batzuk hizkuntzarekiko independenteak dira, hau da, berdin dabiltza edozein hizkuntzatan. Adibidez, BLEU motako metrikak, itzulpenaren eta erreferentziaren artean hitzen distantziak neurtzen dituztenak, ez dute zertan hizkuntza kontuan hartu behar. Baina kontu gramatikaletan sartzen direnak ezingo dute edozein hizkuntzarekin egin lan.

Asiya-k metrika asko dakartza, eta gehienek aldaera anitz dituzte. Baina ezin dira beti denak erabili. Gutxi batzuk dira independenteak, baina beste asko, ez. Eta horien artean gehienak ingelererako soilik balioko dute, agian gaztelania eta katalanerako ere... Badira asko, baita ere, hizkuntza baterako ere ez daudenak prest.

Proiektuaren zati honetan helburua metrika horietako batzuk hartu eta emandako analizatzaile bat erabiliz, euskarara moldatzea da.

6.2 Asiya martxan jartzea

Asiya programa abiatzeak uste baino denbora gehiago hartu egin du, dozenaka aplikazio, azpiprograma, pakete eta modulu instalatu behar izan baitira. Gainera, hori egin zen ordenagailua jarri berria zen eta ez zuen ezer ere ez aurretik instalatua. Beraz, paketeen arteko behar eta dependentziak zirela tarte, denbora luze bat behar izan zen Asiya instalatzeko soilik.

Bestalde, zailtasunak egon ziren softwarearen funtzionamendu zehatza ulertu eta abian jartzeko, erabiltzaile-gida tekniko bat dakarren arren. Asiya modu normalean erabiltzeko konfigurazio-fitxategi bat sortu eta erabili behar da, zeinetan egin nahi diren ebaluazioei buruzko informazioa idatzi egin behar den: zeintzuk diren fitxategi originala, itzulpen automatikoa eta erreferentzia edo erreferentziak; zein metrika onartu nahi diren eta zeintzuk erabili; testuen formatua zehaztu; bestelako aukerak; etabar.

Azkenik, bestelako zailtasunak ere agertu egin dira kodea aldatzen zen bitartean frogak egiterakoan funtzionamendua egokia zen edo ez egiaztatzeko, edota utzitako trazak jarraitzeko. Horrek esperimentuak behin eta berriro egitea eskatzen du, kodearen zati txikiak

begiratuz bakoitzean, edo konfigurazioaren xehetasunak aldatuz, edota traza desberdinen bila. Asiya-k exekuzio bakoitzean hainbat fitxategi sortzen ditu: hainbat, tarteko datuak eta analisiak gordetzen dituztenak, eta besteren bat amaierako ebaluazioaren emaitzekin. Arazoa exekuzio bakoitzean egiten duen lehenengo gauza horiek bilatzea da, analisiak ez daitezen errepikatu dagoeneko egin badira. Tarteko eta amaierako emaitza horiek eskura izatea ondo dator erabiltze arruntarekin, baina oztopo bat dira lan hau egitean. Kodea aldatzeko saiakera egin zen, baina aldaketa konplexuegiak ekarriko zituen, kodean nahiko finkatua baitzegoen zati hori. Gainera, fitxategi horiek begiratzea baliagarria zen zenbaitetan. Beraz, zama hori mantentzea erabaki zen. Frogak egitean, beharrezkoa zenean fitxategiak borratzen ziren, eta ahal zenean, programa gelditu fitxategiak sortu aurretik.

6.3 Hizkuntzarekiko independenteak diren metrikak

Hurrengo metrikak, Asiya-k erabili ditzazkeenak eta hizkuntzarekiko independenteak liratekenak dira, hau da, hauek erabiltzea lortu egin dela euskarazko testuekin, euskararako analizatazilerik edo sartu gabe.

Kasu gehienetan, beren aldaera desberdinak ere onartu egiten dira.

- PER, WER eta TER
- BLEU
- NIST
- GTM
- Lexical Overlap

Azkenik, teorian METEOR eta ROUGEren aldaera batzuk arazorik gabe ibili beharko lirateke Asiya-ren dokumentazioaren eta kodeako iruzkinen eta definizioen arabera, baina praktikan ez zebiltzan, eta, beraz, ez dira erabili.

6.4 Euskarazko analizatzaile sintaktikoa

Asiya moldatzeko erabiliko den tresna euskararako analizatzaile sintaktiko bat da. Kodea Perl-ez idatzita dago eta *analizatu_eu.pl* du izena. Errutina nagusiak, *analizatu*-k, testu bat jasotzen du eta web aplikazio bati deituz analisia egikaritzen du eta XML formatuan jaso. XML horretan beharrezkoa izango den informazio sintaktiko guzti hau dago:

- Termino edo hitz bakoitza desberdinduta
  - Hitz bakoitzaren lema (*Gaude* -> *Egon*, *Analizatzailearen* -> *Analizatzaile*...)
  - Hitz bakoitzaren kategoria gramatikala (*Aditzlaguna*, *Izen arrunta*...)
  - Hitz bakoitzaren kasu gramatikala (*Genitiboa*, *Absolutiboa*...)
- Hitzen arteko dependentziak
  - Zein hitzen artean
  - Dependentzia mota
  - Kasu gramatikala
- *Chunking*-a, edo sintagmak bilatzea
  - Zein hitzek eratzen duten
  - Zein hitz den erroa
  - Kasu gramatikala

Orokorrean erabiltzeko funtzio bat eskaintzen du *analizatu_eu.pl*-k informazio hortatik nabaria den hainbat lortzeko, *interpretatu_xml()*. XML dokumentuan oinarrituz, honakoak antolatzen ditu testu hutseko formatuan:

```
hitza|lema|kategoria|chunk-a
```

Jarraian, *interpretatu_xml*. Beltzez, *analyzed* parametroaren jarraipena, horrek gordetzen baitu emaitza.

```
sub interpretatu_xml() {
    my $xml = shift;
    my $input = shift;
    my $output = "";
    my $parser = new XML::LibXML;
    my $ana_doc = $parser->parse_string($xml);
    my @sarrerak = split(/\n\n/, $input);
    my $i = 1;
    foreach my $k (@sarrerak) {
        my $analyzed = "";

        my $key = $k;
        $key =~ s/\s+$//;
        $key =~ s/^\s+//;
        $key =~ s/\s+/ /;

        my @nodes = $ana_doc->getDocumentElement->findnodes("//wf[\@wid='w" . ($i) . "']");
        my $kk = $nodes[0]->textContent if ($#nodes >= 0);
        my $t = $nodes[0];

        while ($#nodes >= 0 && $key =~ s/^\s*\Q$kk\E//) {
            $analyzed .= &encode_reserved($t->textContent, 1) . "|";
            my $wid = $t->getAttribute('wid');

            my @terms;
            @terms = $ana_doc->getDocumentElement->findnodes("//term[./span/target/\@id='$wid']");
            if (defined $terms[0]) {
                my $kat = $terms[0]->getAttribute('pos');

                $kat =~ s/^[A-Z]\.([A-Z]+)(\-[A-Z]+)?$/\1/;
                $kat =~ s/^[A-Z]\.([A-Z]+)_IZEELI$/IZEELI/;

                $analyzed .= &encode_reserved($terms[0]->getAttribute('lemma'), 0) . "|" . $kat . "|" .
                        $terms[0]->getAttribute('tid');
            }
```

```
            else {
                $analyzed .= &encode_reserved($t->textContent, 1) . "|NoKat|O";
            }
            $analyzed .= " ";

            $i++;
            @nodes = $ana_doc->getDocumentElement->findnodes("//wf[\@wid='w" . ($i) . "']");
            $kk = $nodes[0]->textContent if ($#nodes >= 0);
            $t = $nodes[0];
        }

        my @chunks = $ana_doc->getDocumentElement->findnodes("//chunk");
        foreach my $c (@chunks) {
            my $phrase = "B." . $c->getAttribute('phrase');
            my @elems = $c->findnodes("./span/target");
            foreach my $t (@elems) {
                my $tid = $t->getAttribute('id');
                if ($analyzed =~ s/\|$tid( |$)/|$phrase$1/) {
                    $phrase =~ s/^B\./I./;
                    $analyzed =~ s/\|$tid( |$)/|$phrase$1/g;
                }
            }
        }
        $analyzed =~ s/\|t[0-9]+( |$)/|O$1/g;
        $k =~ s/ .\s*$//;
        $analyzed =~ s/\s+\.\|\.\|NoKat\|O\s*$//;
        $analyzed =~ s/^\s+//;
        $analyzed =~ s/\s+/ /;

        $output .= "\n\n" if ($output ne "");
        $output .= $analyzed;
    }
    return $output;
}
```

***20. Irudia**: interpretatu_xml*

Hala ere, aukeratu dugun metrikaren batek ere dependentzia analisitik lortutako informazioaz baliatu daiteke edo baliatu behar du. Horregatik, bigarren funtzio bat sortu da,

interpretatu_dep_xml(), honetan oinarrituz, beste parametro batzuk eskuratuko dituenak XML dokumentutik. Funtzio nagusiari hautazko parametro bat gehituz analisi "arrunta" edo dependentzia analisia egingo den aukeratu daiteke.

```
identifikadorea|hitza|kategoria|dependentzia|dependentzia-kategoria
```

Hau da emaitza. Berriz ere, *analyzed* markatu egin da:

```
sub interpretatu_dep_xml() {
    my $xml = shift;
    my $input = shift;
    my $output = "> ( \n";
    my $parser = new XML::LibXML;
    my $ana_doc = $parser->parse_string($xml);
    my @sarrerak = split(/\n\n/, $input);
    my $id = 1;
    foreach my $k (@sarrerak) {
        my $analyzed = "";

        my $key = $k;
        $key =~ s/\s+$//;
        $key =~ s/^\s+//;
        $key =~ s/\s+/ /;

        my @nodes = $ana_doc->getDocumentElement->findnodes("//wf[\@wid='w" . ($id) . "']");
        my $hitza = $nodes[0]->textContent if ($#nodes >= 0);
        my $t = $nodes[0];

        while ($#nodes >= 0 && $key =~ s/^\s*\Q$hitza\E//) {
            my $wid = $t->getAttribute('wid');
            $analyzed .= $id . "\t" . $hitza . "\t";
            my @terms;

            @terms = $ana_doc->getDocumentElement->findnodes("//term[./span/target/\@id='$wid']");
            if (defined $terms[0]) {
                my $kat = $terms[0]->getAttribute('pos');
```

```
            $kat =~ s/^[A-Z]\.([A-Z]+)(\-[A-Z]+)?$/\1/;
            $kat =~ s/^[A-Z]\.([A-Z]+)_IZEELI$/IZEELI/;
            $analyzed .= $kat . "\t";

            my $tid = $terms[0]->getAttribute('tid');
            my @depe = $ana_doc->getDocumentElement->findnodes("//dep[\@to='" . ($tid) . "']");
            if (defined $depe[0]) {
                    my $nodeF = $depe[0]->getAttribute('from');
                    $nodeF =~ s/t//;
                    my $catNode = $depe[0]->getAttribute('rfunc');
                    $analyzed .= $nodeF . "\t" . $catNode . "\n";
            }
            else {
                    $analyzed .= "*\n";
            }
        }
        else {
            $analyzed .= "NoKat\t*\n";
        }
        $id++;
        @nodes = $ana_doc->getDocumentElement->findnodes("//wf[\@wid='w" . ($id) . "']");
        $hitza = $nodes[0]->textContent if ($#nodes >= 0);
        $t = $nodes[0];
    }
    $analyzed =~ s/\|t[0-9]+(\)\n|$)/|\*$1/g;
    $output .= $analyzed;
}
    $output .= "\n)\n>";
return $output;
}
```

***21. Irudia**: interpretatu_dep_xml*

Egin behar izan zen beste aldaketa bat, *analizatu_eu.pl* fitxategitik (Perl *script* bat dena) *analizatu_eu.pm*-ra bihurtzea (hau da, Perl *module*), honela Asiya-k erabiltzen duen formatuetara egokitzeko, objektuei orientatutako paradigman oinarritzen baita, Perl-eko moduluak erabiliz.

6.5 TERp eta METEOR-en ezinezkotasuna

Aukeratu ziren bi metrika aztertzen ziren heinean arazo bat agerizko egin zen: bai METEOR-en eta baita TERp-en ere, analisiaren kodea ez zen Asiya-ren kodearen parte, besteekin gertatzen zen ez bezala. Horren ordez, aparte instalatutakoen artean daude eta horrek arazo asko ekartzen ditu. Hasteko, Perl-ez idatzita ez daudenez, ikasketa prozesua asko luzatzeko arrisku nabaria zegoen.

Gainera bai METEOR eta bai TERp-en lizentziak etab. ez dira Asiya-ren berak, baizik eta kodeak beste urruneko unibertsitateetan egin dituzte: METEOR, Carnegie Mellon University-koa da; eta TERp Unversity of Mariland-ekoa.

Azkenik, baliabideen aurretik denbora nahikoa ez zen izango hau aurrera eramateko. Beste bi metrikekin (Shallow Parsing eta Dependency Parsing) hasita iada, metrika bat moldatzea uste baino zailagoa zela ikusten zen, eta laurak denboran amaitzea oso zaila izango zela ikusten zen, plangintzatik ateraz.

Hori dena dela eta, proiektuaren zuzendariarekin bilera bat egin eta gero, bi metrika hauekin aurrera ez jarraitzea erabaki egin zen.

6.6 Shallow Parsing

Shallow Parsing-ak (azaleko analisi sintaktikoa) analisi sintaktikoaren hainbat elementu hartzen ditu bai itzulpen automatikoak sortutako testutik eta baita

erreferentziazkoaz eta horien arabera ebaluazioaren emaitza bat kalkulatuko du erabiltzen dituen algoritmoetan oinarrituz.

Tartean, SVMTool tresna erabiltzen du testu batetik analisi sintaktikoa formatu jakin batzuetan lortzeko. Hainbat tarteko fitxategi sortzen dira testu bakoitzarekin, ondoren ebaluazioan erabiliko direnak. Kodea sakonki aztertu eta gero, *FILE_parse* metodoan gertatzen dela aurkitu zen, eta bost dokumentu sortzen ziren, denak formatu berarekin baina elementu desberdinekin:

- **wp**file: hitza (**w**ord) eta kategoria (**p**os – part of speech)
- **wpl**file: hitza (**w**ord), kategoria (**p**os) eta lema (**l**emma)
- **wc**file: hitza (**w**ord) eta chunk-a (**c**hunk)
- **wpc**file: hitza (**w**ord), kategoria (**p**os) eta chunk-a (**c**hunk)
- **wplc**file: hitza (**w**ord), kategoria (**p**os), lema (**l**emma) eta chunk-a (**c**hunk)
- conllfile: zenbagarren hitza, hitza, lema, kategoria (sinplifikatua) eta kategoria

Behin hau jakinda, aipaturiko *FILE_parse* aldatzea baino ez zen gelditzen. Saiakera asko eta gero, hau da emaitza. Beltzez markatu da euskarazko soilik analisia non hasten eta amaitzen den.

```
sub FILE_parse {
   my $input = shift;
   my $parser = shift;
   my $tools = shift;
   my $L = shift;
   my $C = shift;
   my $iter = 0;
   my ($w, $p, $l);

   if (exists($SP::rLANG->{$L})) {
      my $wplcfile = $input.".$SP::SPEXT.wplc";
      if ((!(-e $wplcfile)) and (!(-e "$wplcfile.$Common::GZEXT"))) {
```

```
my $wpfile = $input.".$SP::SPEXT.wp";
my $wplfile = $input.".$SP::SPEXT.wpl";
my $conllfile = $input.".$SP::SPEXT.conll";
if ((!(-e $wpfile)) and (!(-e "$wpfile.$Common::GZEXT"))) {
   open(INPUT, "< $input") or die "couldn't open input file: $input\n";
   open(WPFILE, "> $wpfile") or die "couldn't open output file: $wpfile\n";
   open(WPLFILE, "> $wplfile") or die "couldn't open output file: $wplfile\n";
   open(CONLLFILE, "> $conllfile") or die "couldn't open output file: $conllfile\n";

  if ($L eq "eu") {
            my $wcfile = $input.".$SP::SPEXT.wc";
            my $wpcfile = $input.".$SP::SPEXT.wpc";
            open(WCFILE, "> $wcfile") or die "couldn't open output file: $wcfile\n";
            open(WPCFILE, "> $wpcfile") or die "couldn't open output file: $wpcfile\n";
            open(WPLCFILE, "> $wplcfile") or die "couldn't open output file: $wplcfile\n";

            while (my $line = <INPUT>) {
                   chomp($line);
                   my $analisia0 = analizatu_eu::analizatu($line);
                   my @analisia1 = split(/ /, $analisia0);
                   my @RESwp;
                   my @RESwpl;
                   my @RESwc;
                   my @RESwpc;
                   my @RESwplc;
                   for (my $i = 0; $i<@analisia1; $i++)
                   {
                          my @analisia2 = split(/\|/, $analisia1[$i]);
                          push(@RESwp, $analisia2[0]." ".$analisia2[2]);
                          push(@RESwpl, $analisia2[0]." ".$analisia2[2]." ".$analisia2[1]);
                          push(@RESwc, $analisia2[0]." ".$analisia2[3]);
                          push(@RESwpc, $analisia2[0]." ".$analisia2[2]." ".$analisia2[3]);
                          push(@RESwplc, $analisia2[0]." ".$analisia2[2]." ".$analisia2[1]."
                                 ".$analisia2[3]);
                   }

                   print WPFILE join ("\n", @RESwp)."\n\n";
                   print WPLFILE join ("\n", @RESwpl)."\n\n";
                   print WCFILE join ("\n", @RESwc)."\n\n";
```

```
                        print WPCFILE join ("\n", @RESwpc)."\n\n";
                        print WPLCFILE join ("\n", @RESwplc)."\n\n";
                        my $wcount = 1;
                        foreach my $line (@RESwpl) {
                                ($w, $p, $l) = split(/\s/, $line);
                                print CONLLFILE "$wcount\t$w\t$l\t".substr($p, 0, 1)."\t$p\t_\n";
                                $wcount++;
                        }
                        print CONLLFILE "\n";
                        $iter++;
                }
                close(WCFILE);
                close(WPCFILE);
                close(WPLCFILE);
        }
        else {
         ...
        }
         close(INPUT);
         close(WPFILE);
         close(WPLFILE);
         close(CONLLFILE);         }

    my $wcfile = $input.".$SP::SPEXT.wc";
    if ((!(-e $wcfile)) and (!(-e "$wcfile.$Common::GZEXT"))) {
       if ((!(-e $wpfile)) and (-e "$wpfile.$Common::GZEXT")) { system("$Common::GUNZIP $wpfile.
          $Common::GZEXT"); }
       if (!($L eq "eu")) {
          ...
       }
         my $wpcfile = $input.".$SP::SPEXT.wpc";
         SP::FILE_merge_BIOS($wpfile, $wcfile, $wpcfile);
         system("$Common::GZIP $wpfile");
         if ((!(-e $wplfile)) and (-e "$wplfile.$Common::GZEXT")) { system("$Common::GUNZIP
            $wplfile.$Common::GZEXT"); }
         SP::FILE_merge_BIOS($wplfile, $wcfile, $wplcfile);
         system("$Common::GZIP $wplfile");
         system("$Common::GZIP $wpcfile");
         system("$Common::GZIP $wcfile");
```

```
            system("$Common::GZIP $conllfile");
        }
    }
  }
  else { die "[ERROR] Shallow parser not available for '$L' language!!\n"; }
}
```

***22. Irudia**: SP.FILE_parse*

6.7 Dependency Parsing

Dependency Parsing-aren funtzionamendua Shallow Parsing-arenaren antzekoa da. *FOREST_parse* metodoak testu baten dependentzia analisia lortuko du zuhaitz moduan. Zuhaitzak (itzulpen automatikoarena eta erreferentziena, bat edo gehiago izan) bestelako algoritmoei pasako zaie, zeintzuek itzulpen automatikoaren ebaluazio bat egingo dute eta emaitza bat sortu.

Aurrekoan bezalaxe, Asiya-k tresna propio bat du dependentzien analisia egiteko, DParser (*Dependency Parser*) izenekoa. Gure analizatzaileak dependentziak ere lortzen ditu, baina hura deskribatzerakoan azaldu den bezala, XMLa interpretatzen duen aplikazioak ez zuen defektuz hori ere itzultzen. Horretarako gehitu egin zaio dependentziak ere lortzen dituen irakurketa bat egin ahal izatea.

Oro har, Dependency Parsing-en euskararako dependentzia analisiaren zuhaitza lortzeko kodea sinple eta laburragoa dirudi, biek izan dutelarik zailtasuna formatu desberdinak lortzeko (SP-aren fitxategiak eta DP-aren zuhaitzak), baina bigarren honek analizatzailearen interpretazioa gehitu beharraren zailtasun gehitua izan du.

Aurrekoan bezala, beltzez markatu dira euskarazko analisia egiten den tartearen hasiera eta amaiera.

```
sub FOREST_parse {
   my $file = shift;
   my $kind = shift;
   my $tools = shift;
   my $verbose = shift;
   my $language = shift;
   my @FOREST;
   my $forest = $file.".$DP::DPEXT.".($kind?"d":"c")."-forest";

   if (!(-e $forest)) {
      if (-e "$forest.$Common::GZEXT") { system("gunzip $forest.$Common::GZEXT"); } #...should it be in
a .gz, extract it.
      else {
         if ($verbose > 1) { print STDERR "running minipar parser [$file -> $forest]\n"; }

        if  ((defined($language)) and ($language eq $Common::L_EUS)) {
              my $analisia;
              my @analisiak;
              open(FILE, "< $file") or die "couldn't open input file: $file\n";
              open(X, "> $forest") or die "couldn't open output file: $forest\n";
              while (my $lerroa = <FILE>) {
                     $analisia = analizatu_eu::analizatu($lerroa, 1);
                     push (@analisiak, $analisia);
              }
              print X join ("\n", @analisiak)."\n\n";
              close(X);
              close(FILE);
        }
        else {
              ...
        }
      }
   }

   open(AUX, "< $forest") or die "couldn't open file: $forest\n";
   while (my $line = <AUX>) {
      chomp($line);
      if ($line=~/\> \(/) {
         my @tree;
```

```
        my $STOP = 0;
        while (!$STOP) {
           my $l = <AUX>;
           chomp($l);
           if ($l =~ /^\)$/) { $STOP = 1; }
           else {
              $l =~ s/\(//g;
              $l =~ s/\)//g;
              $l =~ s/\~ //g;
              my @entry = split(/\t/,$l);
              my $cat = $entry[2];
              my $catNode = "";
              if (scalar(@entry) > 4) { $catNode = $entry[4]; }
              if ($catNode ne ""){ $cat=~/(\S*)$/; $cat=$1; $entry[2] = $cat; push(@tree, \@entry); }
               }
        }
        push(@FOREST, \@tree);
      }
   }

   close(AUX);

   system("$Common::GZIP $forest");

   return \@FOREST;
}
```

***23. Irudia**: DP.FOREST_parse*

6.8 Azken pausoak

Alde batetik, proiektuaren hurrengo atalean Asiya ere erabili denez, bien artean aldaketaren bat egin behar izan da kodean han eta hemen, hobekuntza txiki batzuk. Horretaz gain, eranskin gisa atxikitu da erabiltze-gida labur bat Asiya-ren ebaluazio-konfigurazioa nola egin azalduz (bestelako xehetasunetan sartu barik).

Orokorrean, hala ere, OmegaT ez bezala Asiya ez dago edozein erabiltzaile arrunt batentzat pentsatua, Lengoaia Naturalaren Prozesamenduaren eta informatikaren ezagutzak dituen norbaitentzat baizik. Programan egin diren aldaketak ere ez daude edozein erabiltzailerentzat pentsatuta.

Hortaz, eranskin gisa gehitu den gida hori konfigurazioan soilik murgiltzen da. Izatez, Asiya-ren instalazioa eta martxan jartzea ongi azalduta daude Asiya-ren dokumentazio teknikoan, baina konfigurazioaren azalpena ez zait guztiz egokia iruditu. Beraz, lan hau jarraitu dezakeen norbaitentzat dago batez ere zuzendua gida hori.

7. DiSeg-ekin esperimentuak

7.1 DiSeg erabiltzea

DiSeg aplikazioa esaldi bat edo gehiago segmentu edo azpiesaldi desberdinetan banatzeko ahalmena du. Jarraian egikarituko den esperimentuan, corpus-eko esaldiak, gazteleraz daudenak (DiSeg gaztelerazko esaldietarako dago prestatua) banatu edo segmentatuko dira. Matxin itzultzaile automatikoarekin gazteleratik euskarara itzuliko dira esaldi segmentatu horiek. Ondoren, Asiya erabiliz itzulpen automatikoaren ebaluazioa egingo da, bi kasuak konparatuko dira.

Esperimentuaren helburua, beraz, Matxin-ekin euskararako itzulpen automatikoaren emaitzak hobetzen diren ikustea da, esaldiak laburragoak bihurtu direnean. Espero da hala izatea eta itzulpenaren ebaluazioan hobekuntza ikusi ahal izatea. Hala ere, egin nahi diren esperimentuen kopurua oso handia da eta ez da espero guztiak bukatu ahal izatea. Hortaz, etorkizunerako lan dotorea egongo da esperimentu hauek zabaltzen eta Matxin horien inguruan hobetu nahian, dagokion atalean azalduko den bezala.

(hurrengo orrialdean) ***24. eta 25. Irudiak****: DiSeg eta Asiya esperimentuetan erabiltzen*

el instituto pío baroja acogió ayer una prueba atlética para recaudar fondos para el país africano .

Bidali Berrezarri

Segmented text:

el instituto pío baroja acogió ayer una prueba atlética
para recaudar fondos para el país africano.

Segmented text with POS tags:

el(el)_D instituto(instituto)_N pío(pío)_A baroja(baroja)_N acogió(acoger)_V ayer(ayer)_R una(uno)_D prueba(prueba)_N atlética(atlético)_A
para(para)_S recaudar(recaudar)_V fondos(fondo)_N para(para)_S el(el)_D país(país)_N africano(africano)_A .(.)_F [s]

All XML tags:

```
<XML>
  <S>
    <seg rule="rule1">
      <sn>
        <espec_ms>
          <j_ms role="head">
            <term cat="DA0MS0" lem="el">el</term>
          </j_ms>
        </espec_ms>
        <grup_nom_ms role="head">
          <n_ms role="head">
            <term cat="NCMS000" lem="instituto">instituto</term>
```

```
----------------------------------------------------------------------------------------------------
ASIYA v0.1
(C) 2010. TALP, Technical University of Catalonia (written by Jesus Gimenez)
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
CONFIGURATION OPTIONS
----------------------------------------------------------------------------------------------------
ASIYA_HOME = /[redacted]/Asiya
language pair = (Spanish, case sensitive) -> (Basque, case sensitive)
----------------------------------------------------------------------------------------------------
input format = raw
output format = mmatrix
granularity = sys
source = ./fitx/adibideak.es.tok
----------------------------------------------------------------------------------------------------
system_set = { adibideak.MT.eu }
reference_set = { adibideak.eu0 }
metric_set = { BLEU, GTM-2, NIST, -PER, -TER, -WER }
----------------------------------------------------------------------------------------------------
[METRICS] computing 'system vs. reference' scores (one vs. all)...
adibideak.MT.eu - adibideak.eu0 [BLEU..NIST..WER..PER..TERp..GTM..]
Printing evaluation report...
SET         SYS              BLEU      GTM-2      NIST     -PER     -TER     -WER
==========================================================================================
UNKNOWN_SET adibideak.MT.eu   0.08377707   0.24754673   2.83572112  -0.81954887  -0.88721805  -1.02255639
[FINISHED]
```

7.2 Esaldi osoekin frogak

Hasteko, esaldi osoekin egikaritu dira frogak, hau da, DiSeg erabili gabe. Horretarako, corpus-a hartu egin da eta Asiya-tik pasa egiten saiatu egin da euskaraz badabiltzan metrikekin, Asiya-ren atalean aipatu direnak, hain zuzen ere.

Metrika arruntekin arazorik ez da egon. Proiektu honen parte bezala moldatu egin diren bi metrikekin, analisi sintaktikoa erabiltzen zutenak, Shallow Parsing eta Dependency Parsing, bai egon direla arazoak.

Hasteko, analizatzaileak benetan denbora asko hartzen du esaldi bakoitzeko. Eskaera bat egiten zaion bakoitzean hasieraketa asko egin behar omen ditu, horrek moteltzen du. Arazoa, zera da: bai Asiya eta baita perl-ez idatzitako analizatzaileari deitzen dion *analizatu_eu.pm*-k ere esaldiak banan bana bidaltzen dizkio analizatzaileari. Hortaz, prozesua azkarragoa izan ziteken arren, esaldi bakoitzeko 30 segundu inguru iraun dezake lasai asko. Kontuan harturik gutxienezko kasuan, gutxi gora-behera, 3000 esaldi aztertu behar direla (corpus-eko 1000 esaldiak, esaldi "osoekin", DiSeg-etik pasa eta gero, eta erreferentziazkoak, bakarra erabiltzen bada eta ez biak), denbora gora doa kontrolik gabe.

Hori konpontzeko kodea aldatzeko saiakera egin da, baina modu erraz eta azkarrean egiteko modurik ez denez aurkitu, saihestea erabaki da. Horren ordez, hasiera batean bi metrika hauek erabiltzeko ez da corpus osoa erabiliko, totalaren %10a diren 100 ausazko esaldi baizik. Gehiago aztertzeko aukera irekia dago etorkizunerako, eta proiektuaren dokumentazioa entregatu eta gero baina defentsa egin aurretik lagin hau zabaltzea proposatu egin da.

Ausaz aukeratze hori ez da egitan guztiz ausazkoa izan. Horren ordez, esaldiak

hogeinaka hartu dira bost multzoetan. Corpus-ean mota desberdinetako esaldiak daudenez, eta hasierakoak laburragoak izanda eta amaierakoak lerro askotakoak, honela anizkotasuna bermatzeaz gain, esaldi motz edo luzeekin desberdintasunak ageri diren ikusi ahal izango da.

Gainera, beste arazo larri bat sortu da esperimentuak egikaritzen ziren bitartean. Ikaslearen irismenetik kanpo gelditzen den analizatzaileak berak arazoak eman egin ditu, hainbat esaldi ez baititu ongi aztertzen. Laguntzarekin ere ez da lortu oraindik zergatia aurkitzea. Analizatzaileak XML dokumentu huts bat itzultzen zuen esaldi horiekin eta programak ez zekien nola interpretatu, analisi guztia bertan behera utziz eta ordu asko galaraziz esaldi 'txarrak' identifikatzeko lanean. Beraz, esaldiak ausaz aukeratu diren arren, horietako portzentai bat ez zebilen eta beste inguruko esaldiekin ordezkatu egin dira (ingurukoak, aipatu den hurbiltasuna ez apurtzeko).

7.3 DiSeg-etik pasa diren esaldiekin frogak

Ondoren, atal honen zati interesgarrienarekin hastea posible izan da.

Hasieran, corpus-eko esaldiei formatu egokiago bat ezarri egin zaio, esaldiak ongi banatuz. Geroago hau eskuz desegin da, baina esaldiak DiSeg-ek banatu ondoren ere, originalak zeintzuk ziren gogoratzea ahalbidetzen duen.

Behin gaztelaniazko esaldi horiek moldatuta, DiSeg-etik pasa egin dira. Esaldien kopurua ia %250an igo da, 1000 ingurutik ia 2500era. DiSeg-ek formatu marka batzuk gehitzen dituela ikusi egin da, adibidez, barra etzanak kakotxen aurrean eta marra baxua zenbaki batzuk eta hainbat erpresiorekin. Formatu-marka horiek ezabatu eta gero, esaldi guzti hauek Matxin-ek itzuli ditugu.

Matxin-etik lortutako itzulpenak gorde egin dira eta formatua aldatu behar izan zaie bueltan, corpus-ean datozen eta aurreko pausoan erabili egin diren formatu bera izan zezaten. Posible zen formatua mantentzea eta erreferentziazkoak aldatzea erraz, baina hala egin izan balitz, analisi osoa egin behar izango zen berriz horientzako, eta azaldu den bezala, horrek denbora gehiegi hartuko luke.

Bestalde, arrazoi berdinak direla eta, erreferentziak ez aldatzearren, kasu honetan horiek ere segmentatuz, eskuz egin behar izan da DiSeg-ek banatutako eta Matxin-ek itzulitako esaldiak berriz batzeko lana, originalekin bat etortzeko eta konparazioak modu eraginkorrean egin ahal izateko Asiya-rekin. Hau eskuz egindako lan mekanikoa baino ez izan arren, denbora hartu egin du.

Azkenik, aurreko pauso berdinak errepikatu egin dira: corpus osoa metrika "arruntentzat", eta lehengo ausazko esaldi berak metrika "sintaktikoentzat".

7.4 Emaitzak

Metrika "arruntekin" emaitzak. Honakoak erabili dira: BLEU arrunta, 2-mailako GTM, NIST, PER, TER, WER (azken hauek emaitza negatiboa itzultzen dute, "altuago = hobeto" mantentzeko) eta Lexical Overlap barruko 4 aldaera erabili egin dira. Bakoitza bi erreferentzia desberdinekin probatu da eta ondoren batazbestekoak atera dira. Horiek konparatu dira.

	BLEU	GTM-2	NIST	-PER	-TER	-WER	Ol	Pl	Rl	Fl
Originalak, 1. erreferentzia	0,134	0,221	5,556	-0,505	-0,685	-0,738	0,357	0,520	0,522	0,517
Originalak, 2. erreferentzia	0,108	0,202	5,006	-0,552	-0,732	-0,782	0,326	0,492	0,492	0,486
Originalak, batazbestekoa	**0,122**	**0,211**	**5,281**	**-0,529**	**-0,708**	**-0,760**	**0,341**	**0,506**	**0,507**	**0,502**
Segmentatuak, 1. erreferentzia	0,084	0,170	4,513	-0,608	-0,764	-0,831	0,292	0,424	0,468	0,442
Segmentatuak, 2. erreferentzia	0,059	0,149	3,893	-0,652	-0,819	-0,874	0,254	0,384	0,423	0,397
Segmentatuak, batazbestekoa	**0,071**	**0,160**	**4,203**	**-0,630**	**-0,791**	**-0,852**	**0,273**	**0,404**	**0,446**	**0,419**
ALDAKETA	-0,051	-0,051	-1,078	-0,101	-0,083	-0,092	-0,068	-0,102	-0,061	-0,083
PORTZENTAIA	**%-41,8**	**%-24,2**	**%-20,4**	**%-19,1**	**%-11,7**	**%-12,1**	**%-19,9**	**%-20,2**	**%-12,0**	**%-16,5**

***26. Irudia**: Emaitzak metrika arruntekin*

Ondoren, metrika "sintaktikoekin" emaitzak. Bai Dependency Parsing eta bai Shallow Parsing aztertu egin dira, bakoitza bi Lexical Overlap desberdinekin alderatuz.

	DP-Oc(*)	DP-Ol(*)	SP-Oc(*)	SP-Op(*)
Originalak, 1. lagina	0,7500	0,2482	0,6255	0,6906
Originalak, 2. lagina	0,0000	0,0000	0,0561	0,0545
Originalak, 3. lagina	0,3750	0,1853	0,4262	0,4965
Originalak, 4. lagina	0,5589	0,1150	0,4682	0,5134
Originalak, 5. lagina	0,3690	0,1062	0,4734	0,5133
Originalak, GUZTIRA	**0,4106**	**0,1309**	**0,4099**	**0,4537**
Segmentatuak, 1. lagina	0,5125	0,2294	0,4782	0,5370
Segmentatuak, 2. lagina	0,3889	0,1217	0,3499	0,3850
Segmentatuak, 3. lagina	0,0000	0,0000	0,0248	0,0203
Segmentatuak, 4. lagina	0,2857	0,1308	0,4506	0,4951
Segmentatuak, 5. lagina	0,2522	0,1249	0,4183	0,4486
Segmentatuak, GUZTIRA	**0,2879**	**0,1214**	**0,3444**	**0,3772**
ALDAKETA	-0,1227	-0,0096	-0,0655	-0,0765
PORTZENTAIA	**-%29,8894**	**-%7,3058**	**-%15,9868**	**-%16,8554**

***27. Irudia**: Emaitzak metrika sintaktikoekin*

Argi gelditzen denez, edozein kasutan emaitzak okertu direla esan daiteke.

Zer espero zen? Corpus-eko esaldiak DiSeg-etik pasa eta gero, esaldi kopuru altuagoa lortu egin da, baina laburragoak, luzera guztira mantentzen baitzen. Matxin bezalako arauetan oinarritutako itzultzaile automatiko bat eraginkorragoa da esaldi motzekin, konplexutasun sintaktiko txikiagoa baitute. Hori dela eta, esaldi laburragoak pasatzerakoan itzulpen hobeak egitea espero zen.

Zer gertatu da? Emaitzak alderantzizkoak dira: metrika ororekin emaitzak txarrera joan direla ageri da. Metrika guztiekin gertatu da, eta egindako frogen batazbestekoaren arabera, emaitzak ia %20an okertu dira. Zergatik ez da hobetu, espero zen bezala? Hainbat faktore eta arrazoi daude hau honela izateko.

Esaldiak banatzeak onurak ekarriko dituela suposatzen da. Baina, noski, segmentatze-prozesuak, hau da, testuaren aurre-edizioak, hainbat arazo ere ekarri ditu, zehatz-mehatz aurreikusteko zailak kasu batzuetan.

Hasteko, DiSeg-ek esaldiak banatu eta Matxin-ek itzuli eta gero, esan den bezala, Asiya-ra pasa aurretik berriz batu egin behar dira, erreferentziazko itzulpenekin konparatu ahal izateko. Hau eskuz egin beharreko lana da oraingoz, ez baitugu oraindik garatu moduren bat automatikoki egiteko. Matxin-en itzulpenak ez daudenez prest prozesu zehatz hau jasateko eta eskuz egitean balitekenez faktoreren bat kontuan ez hartu izatea, ez da harritzekoa prozesu honetan esaldi batzuek esanahia edo koherentzia galtzea, eta gero Asiya-n ebaluazioen emaitzak aurrekoak baino okerragoak izatea.

Bestalde, esaldiak banatzerakoan, DiSeg-ek konjuntzio zehatz batzuk bilatzen ditu eta horietan zatitu, bigarren zatian hasieran geldituko delarik. Honek Matxin-i arazoak emateko aukera dago, itzulpen ez oso zehatzak eraginez. Baina ez da arazo bakarra, DiSeg-ek aldaketa

gehiago eragiten dituela testuan aurkitu baita. Lehen azaldu da karaktere batzuk gehitzen dituela zenbait kasuetan, eta konpondu ahal izan da. Baina badira bestelakoak ere, esaterako, atzizkiren bat agertzen bada, zuriune batez bananduko du. Honelako aldaketak ezin ziren aurreikusi, DiSeg-en funtzionamendu eta arau zehatzak ez baitziren hasieran ezagutzen, esperimentuak hasi aurretik. Honek guztiak ere aldaketak sortarazten ditu amaieran Asiya-ra iritsiko diren testuen gainean, horien ebaluazioen emaitzak jaitsiz.

Adibide bat aipaturiko arazoak errazago ikusteko.

El sábado a las 8 voy a vestirme y a salir a la calle para comprar unos zapatos nuevos.

Honela segmentatuko da. "Sábado" eta ondoren datozen hitzak elementu edo hitz bakar bat bezala hartu egin du DiSeg-ek, eta Matxin-ek honelako zeozer aurkituko badu, ez du itzulpen on bat egingo, ez baitu hura ulertuko. Ondoren, "vestirme" bitan zatitzeak ere itzultzaile automatikoari arazoak emango dizkio. Azkenik, Matxin-en barne funtzionamendua ezagutzen ez dudan arren, baliteke "y" edo "para"-z hasten diren esaldiekin lan ona ez egitea.

El sábado_a_las_8 voy a vestir me
y a salir a la calle
para comprar unos zapatos nuevos.

Azkenik, froga gehiago egin beharko liratekela uste da, emaitza agian sendoagoak lortzeko, eta honelako okertzea zehazki zerk ekartzen duen bilatzeko. Hau etorkizunerako lanen proposizioaren parte bezala kontsideratu egin da.

8. Ondorioak eta etorkizunerako lanak

8.1 Ondorioak

Proiektu honetan lan asko egin da, baina zalantzarik gabe emaitzekin gustora gaude; proiektuaren irismena ongi bete egin da, gaiari buruzko asko ikasi da, erreminta baliagarri batzuk euskararako moldatu dira eta hainbat esperimentu egikaritu ahal izan dira.

Hasteko, Lengoaia Naturalaren Prozesamenduari eta Itzulpen Automatikoari buruz gauza asko ikasi egin dira. Horien barruan gai desberdinak trebatu egin dira eta ez bakar bat. Ikuspuntu zabal batetik aztertu dira landu egin diren arloak, horietan gehiago sakonduz eta espezializatuz baina ikuspegi orokorra galdu gabe. Horretaz gain, arrunta dena baino proiektu zabal eta konplexuagoa izanik, honen plangintza eta kudeaketak lan zaila egitea behartu dute. Beraz, proiektu baten garapenean ere ikasi egin dela esan daiteke.

OmegaT tresna hainbat zentzuetan zabaltzeari esker, bere erabilgarritasuna handitu egin da. Euskarara itzulpenak egiten dituen Matxin itzultzaile automatikoa gehitu zaionez, orain euskal komunitatean testuak gaztelania edo ingeleratik euskarara itzuli nahi dituztenek itzulpenean laguntzazko tresna ahaltsu, dohaneko eta libre hau izango dute eskuragarri.

Bestalde, Wikipediako artikuluak editatu eta igotzeko eman zaion ahalmenari esker, beste erabiltzaile komunitate bati ere OmegaT-ren ateak ireki zaizkio: beste hizkuntzetako Wikipedietatik berenera itzultzeko nahai duten guztiena, hain zuzen ere. Bi erabilera horiek batuz, Euskal Wikipediarekin kolaborazioan OpenMT2 proiektuan, erabilpen asko izatea espero da.

Asiya, itzulpen automatikoaren ebaluatzailea, euskaraz aukera gehiago izateko ere moldatu da. Itzulpen automatikoaz itzulitako testu bat eskuz egindako erreferentziazko testu berarekin konparatzean dauden metrikak aztertu egin dira euskaraz zaintzuk erabili daitezken jakiteko. Gainera, metriken artean analisi sintaktikoa erabiltzen dutenen artean bi, azaleko analizatzaile sintaktikoa eta dependentzien analisi sintaktikoa, euskarazko analizatzaile bat erabili ahal izateko aldatu egin dira. Honek ez du OmegaT-ren erabilera bera orokorrean, baino IXA Taldean eta Lengoaia Naturalaren Prozesamenduaren munduan erabilgarria izatea espero da.

Azkenik, egin diren esperimentuak eta horien emaitzak interesgarriak izan dira. Alde batetik, OmegaT, Matxin eta Euskal Wikipediarekin egin diren frogetan, OmegaT eta Matxin ongi erabiltzen direla ikusi da, pertsona askok erabili dute, hainbat itzulitako artikulu igo dira Wikipediara eta Matxin itzultzailea hobetzeko balio izan duten itzulpen-memoria asko bildu dira. Bestalde, Asiya eta DiSeg aplikazioa erabiliz aurre-edizioaren gaineko esperimentuak egin dira. DiSeg-ekin aurre-editatu ditugun testuen itzulpen automatikoak, espero zenaren kontra, hobeak ez direla ebaluatu du Asiya-k. Hala eta guztiz ere, esperimentu hauek hasiera bat baino ez direla esan daiteke eta aurretik bide asko dagoela ere.

Oro har, beraz, pozik gaude proiektuaren ondorio hauekin. Ikaslearen formazioan modu garrantzitsu batean laguntzeaz gain, komunitateari laguntzazkoak izango zaizkion emaitzak lortu direla esan daiteke, bain tresna erabilgarrien itxuran eta baita esperimentuen emaitzekin. Hauen gainean etorkizunean lan asko egin daitezkela espero da.

8.2 Etorkizunerako lana OmegaT-n

- OpenMT2 proiektua egikaritu eta gero eta emaitzak positiboak direlarik, bidea irekia gelditzen da esperimentu gehiago egiteko. Horrekin, Matxin-ek eskuragarri duen itzulpen-memoria kolekzioa handituko litzateke eta aldi berean Euskal Wikipediako artikulu kopurua handituko litzateke.

- Wikipediara hobe egokitzea. Oraindik badira formatu-arazoak, batzuk OmegaT-n eta beste batzuk Matxin-en. Adibidez, MediaWiki formatuko hainbat marka eta etiketa hobeto identifikatu eta tratatzea itzulpen prozesuan laguntzeko, edo, lengoaia desberdinetarako erabili nahi bada, beste karaktere kodeketak erabili ahal izatea (arabiera, txinera, etab.).

- OmegaT-ren bertsio egonkor bat sarean jarri dugun arren, eta dozenaka erabiltzailek instalatu ahal izan duten arren, egindako aldaketak ez dira OmegaT bertsio nagusian azaldu eta sartu. Oso interesgarria litzateke laister OmegaT-ren garapen taldearekin harremanetan jartzea eta hori egiten saiatzea. Honela, OmegaT-ren bertsio berriak jaisten dituen edonork erabili ahal izango ditu proiektu honetan sartutako berrikuntza eta hobekuntzak: Matxin erabili ahal izatea eta artikuluak Wikipediara igo ahal izatea.

- Matxin oro har motela ere ez den arren, beste itzultzaile automatikoekin alderatuz segundu gutxi batzuk gehiago behar dituela nabari da. Hau hobetzeko modu bat bilatzea posible litzateke.

8.3 Etorkizunerako lana Asiya-n

- METEOR eta TERp metrikak moldatzeko beren garatzaile izan diren unibertsitateetako taldeekin kontaktuan jartzea eta kodea moldatzeko prest egotea izango zen hurrengo pausoa. Gure analizatzaileari esker, ez luke arazo handirik egon beharko.

- Bestelako metrikak ere "euskaratzea" posible ote den aztertzea interesgarria izango zen.

- Programaren konplexutasun anitzak direla eta, badira oraindik hainbat hobekuntza egin daitezkenak lan eta denbora gehiago eskaintzen bazaio Asiya-ri eta bere euskarazko analizatzailearekin duen elkarrekintzari, eraginkorrago izateko, arazoak izan baitira arlo honetan, analizatzailearen huts egiteak Asiya-n modu txarrean zabaltzen baitziren (esaldi batekin huts egitean, martxan dauden guztiak egiten dute huts horrekin batera). Gainera, oso motela gertatzen da elkarrekintza hau testu luzeekin.

- Programaren erabilgarritasuna handitzeko lehenengo pauso bat verbose (-v) aktibatuta dagoenean mezu gehiago erakustea izango zen. Kodea komentatu baita baina ez da erabiltzailearentzat iruzkin erabilgarririk gehitu. Harago joanez, oso ideia ona izango zen Asiya-rentzat interfaze on bat diseinatzea, oraingo egoeraren zailtasunak (terminala erabiltzea, fitxategiak eskuz mugitzea, konfiguraziorako fitxategi editatu beharra...) saihesteko.

8.4 Etorkizunerako lana DiSeg-ekin

- Esperimentu hauek lehenengoak izan dira eta gehiago egitea probetxuzkoa litzateke. Bere atalean azaldu den bezala, hainbat arazo zirela eta analizatzailearekin (alde batetik, denbora falta, luzea delako; eta bestetik, onartzen ez zituen esaldiak), ez da corpus osoa sintaktikoki ebaluatu, %10eko lagin bat baizik. Berehalako lana izan daiteke hau zabaltzen saiatzea, esperimentuekin emaitza geroz eta zehatzagoak lortuz honela.
- Bestalde, lan asko hartu du ere corpus-eko esaldiak DiSeg-etik pasatu eta gero, eta itzulpena egin eta banatutako esaldiak berriz batzea ebaluazioa egiteko. Hortaz, mota honetako esaldiak berriz batzeko aplikazio bat erabilgarria litzatekela pentsatu egin da, DiSeg-en kontrakoa egiten duena hain zuzen ere.
- Horri guztiari emaitzak nahi zirenak ez direla izan gehitzen zaio, bere atalean azaldu egin den bezala. Beraz, froga gehiago eginda erroreak finkatzea errazagoa egingo dela eta hobekuntza bilatu ahal izango dela uste da.

Bibliografia

Beste argitalpenak

Unai Cabezón Lumbreras-en argitalpenak:
http://ixa.si.ehu.es/Ixa/Argitalpenak/kidearen_argitalpenak?kidea=1269250121

OpenMT-2: Wikipedia eta Itzulpen Automatikoa biak elkarri laguntzen
Alegria, I., Cabezón, U., Ferandez de Betoño, U., Gonzalez, G., Iturbe, M., Labaka, G., Sarasola, K. eta Zubiaga, A., 2011

Matxin-Informatika: versión del traductor Matxin adaptada al dominio de la Informática
Alegria, I., Cabezón, U., Labaka, G., Mayor, A. eta Sarasola, K., 2011.

Reciprocal Enrichment Between Basque Wikipedia and Machine Translation
Alegria, I., Cabezón, U., Ferandez de Betoño, U., Labaka, G., Mayor, A. eta Sarasola, K. eta Zubiaga, A., 2012 (submited / bidalita).

Artikuluak

Asiya: An Open Toolkit for Automatic Machine Translation (Meta)-Evaluation
Giménez, J. eta Màrquez, L., 2011

METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments
Banerjee, S. eta Lavie, A., 2011

An Open Architecture for Transfer-based Machine Translation between Spanish and Basque
Alegria, I., Díaz de Ilarraza, A., Labaka, G., Lersundi, M., Mayor, A., Sarasola, K., Forcada, M., Ortiz, S. eta Padró, L., 2005.

Webguneak

IXA Taldea (UPV/EHU Euskal Herriko Unibertsitatea)
http://ixa.si.ehu.es/Ixa

Machine translation (Wikipedia: The Free Encyclopedia)
http://en.wikipedia.org/wiki/Machine_translation

Wikipedia: Multilingual statistics (Wikipedia: The Free Encyclopedia)
http://en.wikipedia.org/wiki/Wikipedia:Multilingual_statistics

WikiMedia
http://www.wikimedia.org/

Ethnologue: Languages of the World
http://www.ethnologue.org

Matxin: an open-source transfer machine translation egine
http://matxin.sourceforge.net/

OmegaT: The free (GPL) translation memory tool
http://www.omegat.org/en/omegat.html

Wikiproiektu: OpenMT2 eta Euskal Wikipedia (Wikipedia: Entziklopedia askea)
http://eu.wikipedia.org/wiki/Wikiproiektu:OpenMT2_eta_Euskal_Wikipedia

Java SE Technical Documentation
http://docs.oracle.com/javase/

SOAP: Simple Object Access Protocol, W3C-n
http://www.w3.org/TR/soap/

MediaWiki API documentation page
http://en.wikipedia.org/w/api.php

Asiya: An Open Toolikt for Automatic Machine Translation (Meta-)Evaluation
http://nlp.lsi.upc.edu/asiya/

Meteor: Automatic Machine Translation Evaluation System
http://www.cs.cmu.edu/~alavie/METEOR/

TER-Plus: TERp: Translation-Edit-Rate plus
http://www.umiacs.umd.edu/~snover/terp/

General Text Matcher (GTM)
http://nlp.cs.nyu.edu/GTM/

Perl: The Perl Programming Language
http://www.perl.org/

TAPE Testu-analisirako PERL erremintak
http://www.unibertsitatea.net/blogak/testuak-lantzen/

DiSeg: A Discourse Segmenter for Spanish
http://daniel.iut.univ-metz.fr/~iula/WebDiSeg/index.php

Eranskinak

Jarraian, bi eranskinak irakurri ahal izango dira: OmegaT-ren instalazio eta erabiltze eskuliburua eta Asiya-rekin proiektuan egin diren bezalako ebaluaketak egiteko gida labur bat.

Eskuliburua

eu.wikipedia aberasteko artikuluak espaineratik euskarara itzultzen OmegaT(+Matxin)-ekin

Instalazioa-eskuliburua

+

Erabiltzailearen-eskuliburua

1. OmegaT martxan jarri
2. Wikipediako artikulu bat inportatu
3. Matxin-en itzulpen automatikoa lortu eta editatu
4. Wikipedian gorde

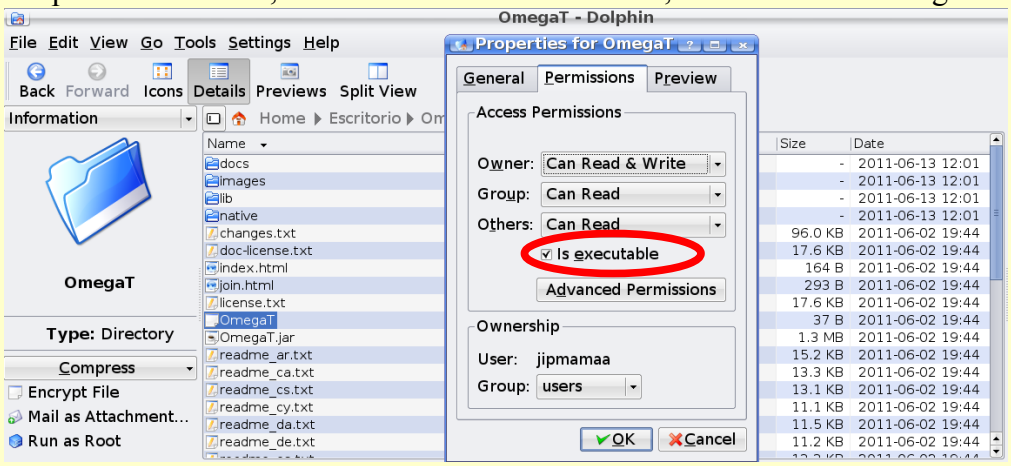
Instalazioa-eskuliburua

1. OmegaT (Matxin erabitzeko egokituta) duen **konprimitutako fitxategia lortu** hauetako helbide batetik:

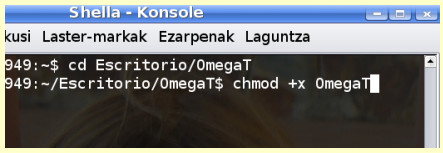
   http://ixa2.si.ehu.es/glabaka/OmegaT/OpenMT-OmegaT.zip
   http://ixa2.si.ehu.es/glabaka/OmegaT/OpenMT-OmegaT.tar.bz2

   Fitxategia destrinkotu, nahi duzun katalogoan.

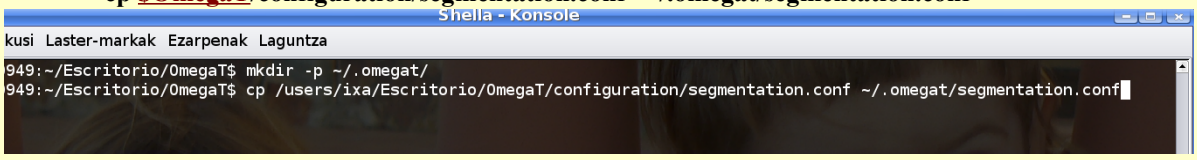
2. *(Linux eta Mac OS bakarrik)*
   OmegaT fitxategiari **exekutatzeko baimena eman**.
   (OmegaT fitxategia dago destrinkotzean sortutako OmegaT katalogoaren barruan).
   Honetarako bi aukera daude:
   1. Fitxategiaren gainean eskuineko botoia sakatu, eta agertzen den menuan aukeratu "Propietateak".
      Propietateen leihoan, "Baimenak" erlaitza aukeratu, eta bertan "Exekutagarria" hautatu.



   2. Terminal bat ireki, joan OmegaT katalogora, eta bertan OmegaT fitxategiari exekutatzeko baimenak eman ondoko komandoarekin:
      chmod +x OmegaT



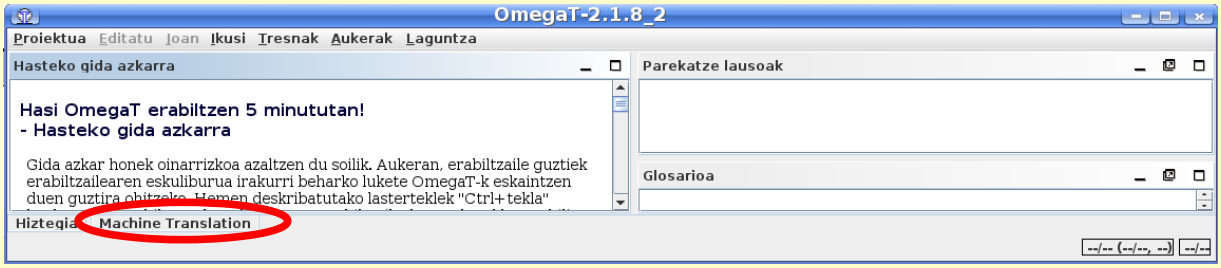
3. OmegaT-ren **segmentazio-erregelak kopiatu**, terminalean ondoko aginduak erabiliz.
   (Ondoko aginduetan $OmegaT ordeztu dagokion path osoarekin)
   1. Windows:
      OmegaT martxan jarri (deskonprimitutako karpetako **OmegaT.bat** fitxategia) eta berriro itxi.
      Kopiatu **configuration\segmentation.conf** fitxategia
      deskonprimitutako karpetatik
      C:\Documents and Settings\%username%\Application Data\OmegaT karpetara
   2. Linux shell:
      mkdir -p ~/.omegat/
      cp $OmegaT/configuration/segmentation.conf ~/.omegat/segmentation.conf



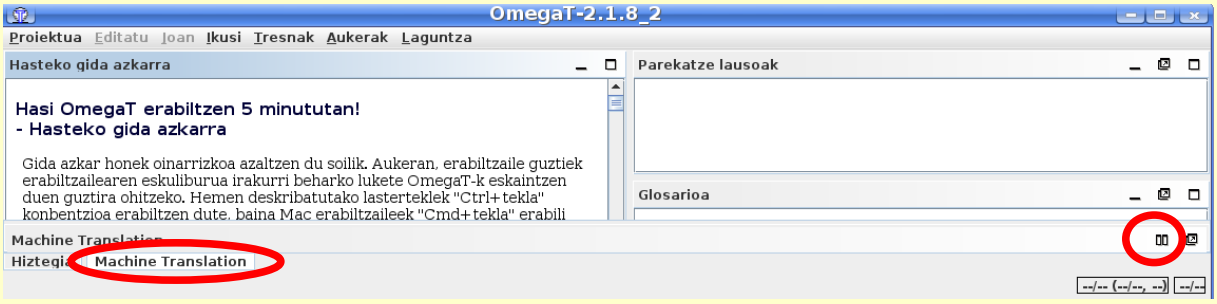
3. Mac OS shell:
   mkdir -p ~/Library/Preferences/OmegaT
   cp $OmegaT/configuration/segmentaion.conf ~/Library/Preferences/OmegaT/segmentation.conf

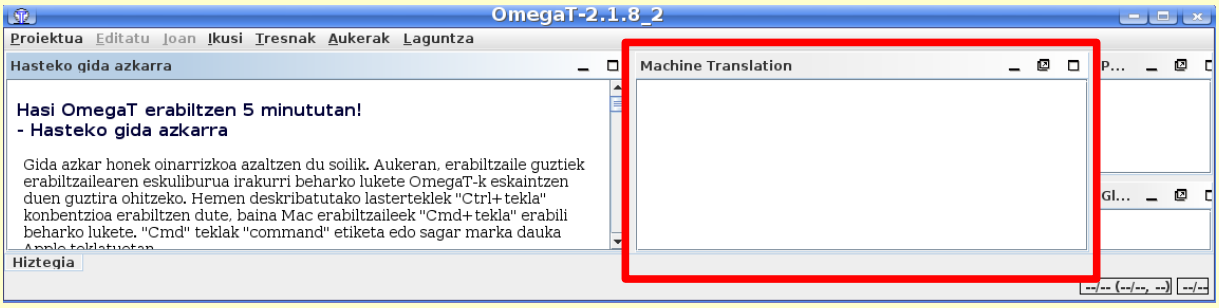
4. **OmegaT martxan jarri**
   (Linux eta Mac OS) OmegaT
   (Windows) OmegaT.bat
   fitxategian klik bikoitza egin.

5. **"Machine Translation" leihoa aktibatu**.
   OmegaT lehen aldiz erabiltzen denean itzulpen automatikoaren leihoa ez da aktibatzen, eta horren ordez "Machine Translation" botoia agertzen da behealdean ("Hiztegia" botoiaren ondoan).



"Machine Translation" botoian klik eginez edo gainean jarriz, itzulpenaren leihoa, hutsik, azalduko da.
Bi zutabetxo duen ikonoa selekzionatuz itzulpen automatikorako leihoa eskuin aldean finkatuko da, lanean hasi ahal izateko.



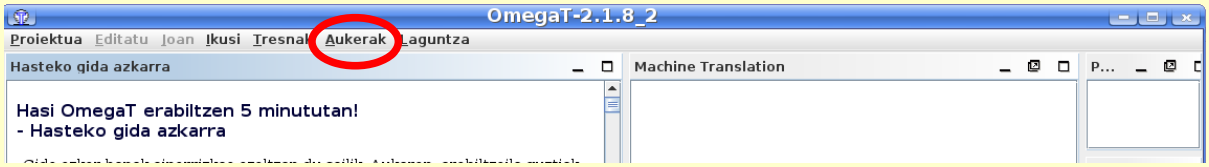


Erabiltzailearen-eskuliburua

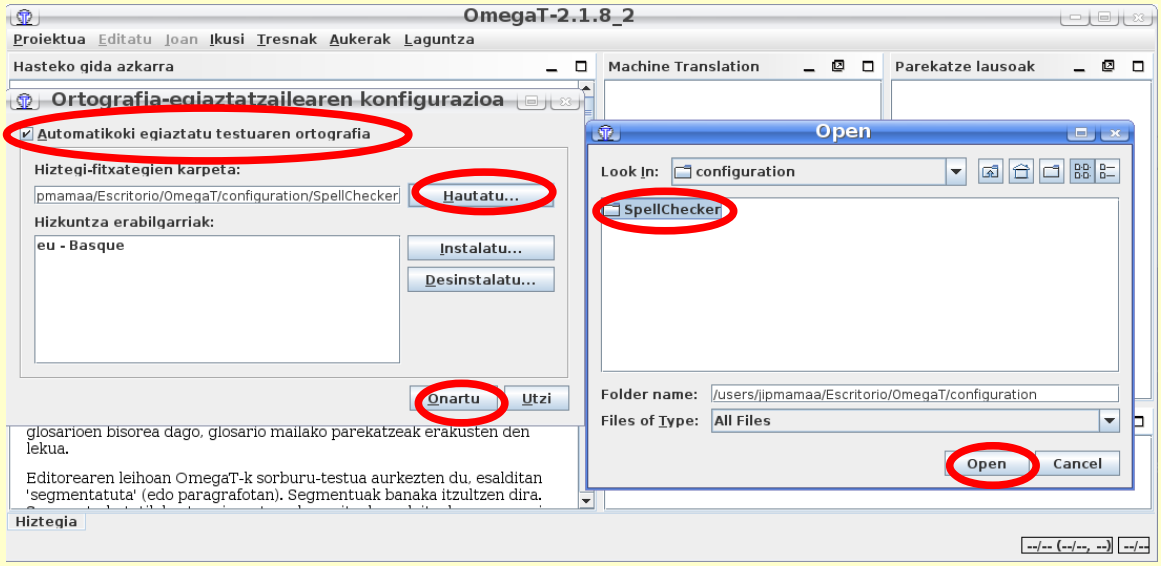
1. OmegaT martxan jarri

1. **OmegaT egikaritu** terminaletik:
   ./OmegaT

2. Euskararako Xuxen **ortografia-egiaztatzaile gaitu**.
   Horretarako, OmegaT tresnaren menuan "Aukerak" menua zabaldu eta bertan "Ortografia-egiaztatzailea" hautatu.
   (OmegaT menua) Aukerak > Ortografia-egiaztatzailea



Zabaltzen den Ortografia-egiaztatzailearen konfigurazioaren leihoan:
1. "Automatikoki egiaztatu testuaren ortografia" markatu
2. "Hautatu" botoia sakatu,
   1. $OmegaT/configuration/SpellChecker katalogoa bilatu eta hautatu
      (Gogoratu $OmegaT dela OmegatT destrinkotu dugun katalogoa)
   2. "Open" botoia sakatu.
3. "Onartu" botoia sakatu



3. **Itzulpen-automatikoa gaitu**:
   (OmegaT menua) Aukerak > Machine Translate > Matxin

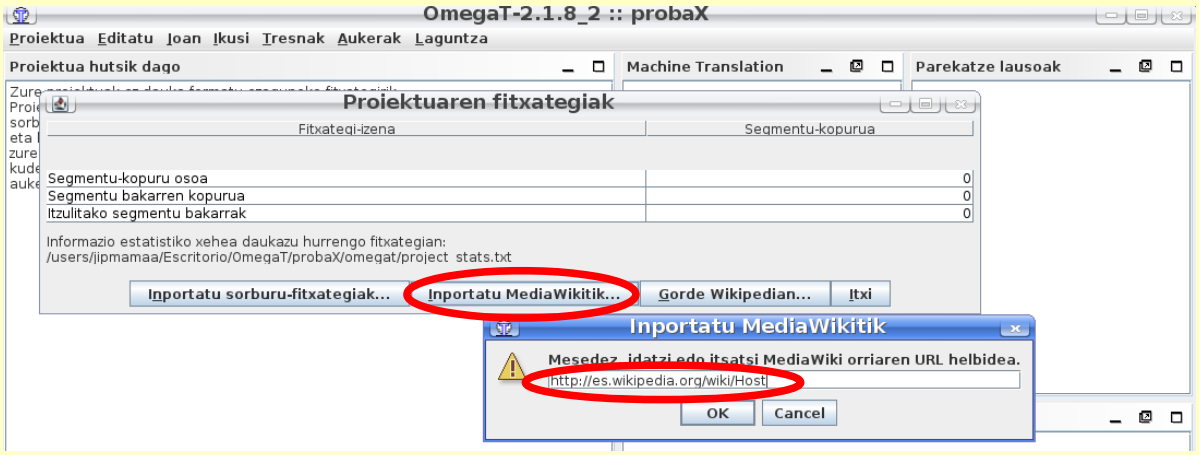
4. OpenMT_proiektua **proiektua ireki**:
   (OmegaT menua) Proiektua -> Ireki (edo **Ctrl+O**)
   Bilatu deskonprimitutako katalogoan **OmegaT/OpenMT_proiektua/**
   eta Onartu botoia sakatu.
   "Proiektuaren fitxategiak" leihoa agertuko da. Jarraitu irakurtzen hurrengo puntua

2. Wikipediako artikulu bat inportatu

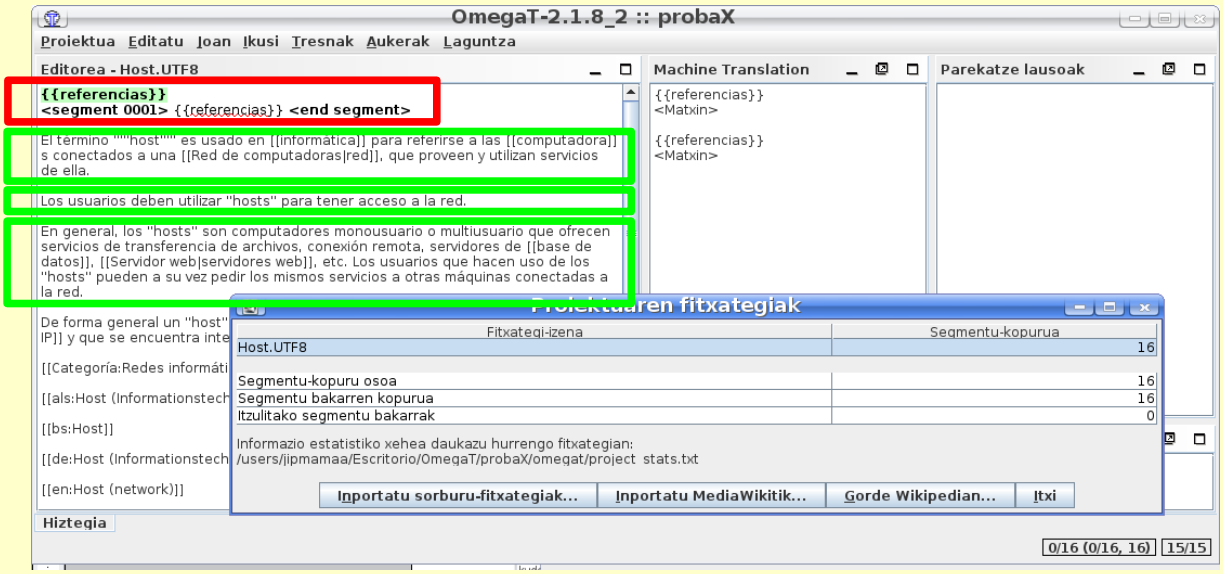
1. Proiektua hutsik badago, **Wikimediako artikulu bat hartu.**
   Horretarako bi aukera daude
   1. Proiektua irekitzen ari bagara, “Proiektuaren fitxategiak” leihoa agertuko da. Lehio horretan “Inportatu MediaWikitik” botoia klikatu.
   2. OmegaT tresnaren menuan “Proiektua” menua zabaldu eta bertan “Inportatu MediaWikitik” klikatu.
      (OmegaT menua) Proiektua -> Inportatu MediaWikitik

   eta testu-kutxan idatzi itzuli nahi den artikuluaren URL osoa
   (adib. http://es.wikipedia.org/wiki/Host)

   Oharra: Kontuz berbideratzeekin (adib. http://es.wikipedia.org/wiki/host), hauek ez baitute ezer itzultzeko



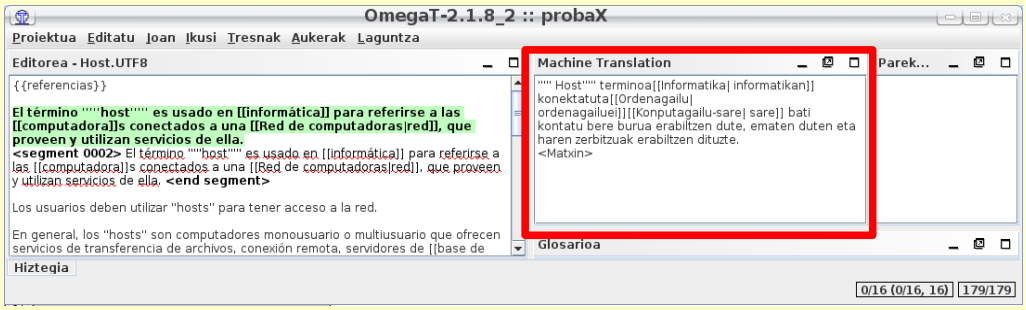
2. OmegaT-k testua hainbat segmentutan banatzen du, eta lehenengoan sartzen da. **Segmentuz aldatzeko**, klik bikoitza egin daiteke beste batean, edo *Enter* sakatu.


OmegaT-2.1.8_2 :: probaX
Proiektua Editatu Joan Ikusi Tresnak Aukerak Laguntza
Editorea - Host.UTF8
{{referencias}}
<segment 0001> {{referencias}} <end segment>
El término "host" es usado en [[informatica]] para referirse a las [[computadora]] s conectados a una [[Red de computadoras|red]], que proveen y utilizan servicios de ella.
Los usuarios deben utilizar "hosts" para tener acceso a la red.
En general, los "hosts" son computadores monousuario o multiusuario que ofrecen servicios de transferencia de archivos, conexión remota, servidores de [[base de datos]], [[Servidor web|servidores web]], etc. Los usuarios que hacen uso de los "hosts" pueden a su vez pedir los mismos servicios a otras máquinas conectadas a la red.
Machine Translation
{{referencias}}
<Matxin>
{{referencias}}
<Matxin>
Parekatze lausoak
Proiektuaren fitxategiak
Fitxategi-izena
Segmentu-kopurua
Host.UTF8 16
Segmentu-kopuru osoa 16
Segmentu bakarren kopurua 16
Itzulitako segmentu bakarrak 0
Informazio estatistiko xehea daukazu hurrengo fitxategian:
/users/jipmamaa/Escritorio/OmegaT/probaX/omegat/project stats.txt
Inportatu sorburu-fitxategiak...
Inportatu MediaWikitik...
Gorde Wikipedian...
Itxi
Hiztegia
0/16 (0/16, 16)
15/15

Oharra: ***Enter*** ordez, ***TAB*** tekla erabili nahi bada horretarako,.***Aukerak*** menuko lehenengo botoia sakatu:
Aukerak > Erabili TAB aurrera joateko

3. Matxin-en itzulpen automatikoa lortu eta editatu

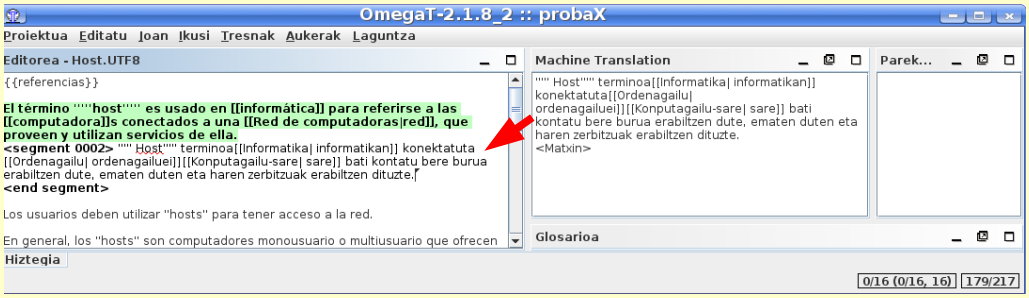
1. "Machine Translation" izeneko leihoan Matxinek eskainitako itzulpena agertzen da.


OmegaT-2.1.8_2 :: probaX
Proiektua Editatu Joan Ikusi Tresnak Aukerak Laguntza
Editorea - Host.UTF8
{{referencias}}
El término """host""" es usado en [[informática]] para referirse a las [[computadora]]s conectados a una [[Red de computadoras|red]], que proveen y utilizan servicios de ella.
<segment 0002> El término """host""" es usado en [[informática]] para referirse a las [[computadora]]s conectados a una [[Red de computadoras|red]], que proveen y utilizan servicios de ella. <end segment>
Los usuarios deben utilizar "hosts" para tener acceso a la red.
En general, los "hosts" son computadores monousuario o multiusuario que ofrecen servicios de transferencia de archivos, conexión remota, servidores de [[base de
Machine Translation
""" Host""" terminoa[[Informatika| informatikan]] konektatuta[[Ordenagailu| ordenagailuei]][[Konputagailu-sare| sare]] bati kontatu bere burua erabiltzen dute, ematen duten eta haren zerbitzuak erabiltzen dituzte.
<Matxin>
Parek...
Glosarioa
Hiztegia
0/16 (0/16, 16)
179/179

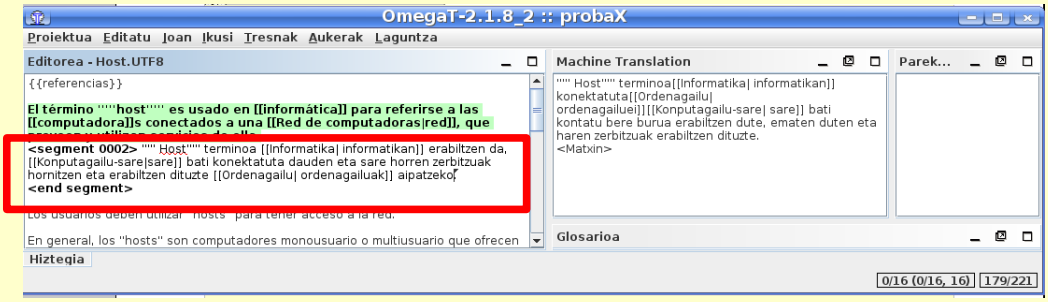
Oharrak:

- Itzulpen automatikoa aktibatzeko (dagoeneko ez badago): *Aukerak -> Itzulpen automatikoak -> Matxin*
- Beste itzultzaile batzuetako itzulpenak ere agertuko dira, hala aukeratu bada.
- Itzultzaile batek itzulpenak eskainiko ditu soilik aktibatuta badago eta jatorri- eta helburu-hizkuntzak tratatu ahal baditu.

2. Itzulpen automatikoa uneko segmentuaren ordez jartzeko: ***Ctrl+M***.

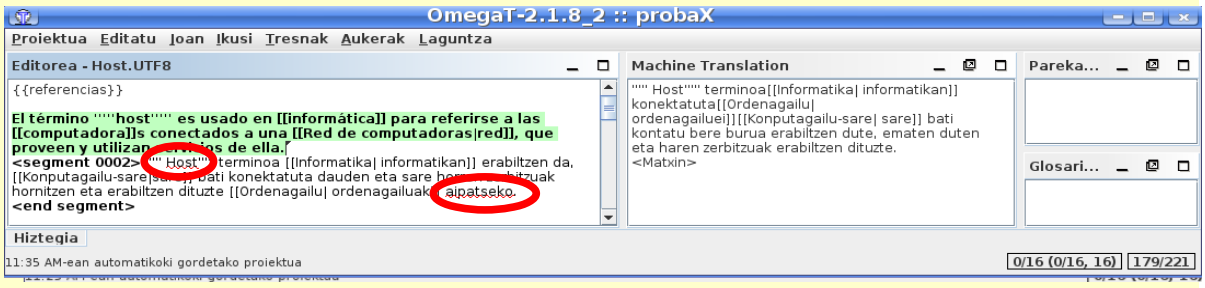

OmegaT-2.1.8_2 :: probaX
Proiektua Editatu Joan Ikusi Tresnak Aukerak Laguntza
Editorea - Host.UTF8
{{referencias}}
El término """host""" es usado en [[informática]] para referirse a las [[computadora]]s conectados a una [[Red de computadoras|red]], que proveen y utilizan servicios de ella.
<segment 0002> """ Host""" terminoa[[Informatika| informatikan]] konektatuta [[Ordenagailu| ordenagailuei]][[Konputagailu-sare| sare]] bati kontatu bere burua erabiltzen dute, ematen duten eta haren zerbitzuak erabiltzen dituzte.
<end segment>
Los usuarios deben utilizar "hosts" para tener acceso a la red.
En general, los "hosts" son computadores monousuario o multiusuario que ofrecen
Machine Translation
""" Host""" terminoa[[Informatika| informatikan]] konektatuta[[Ordenagailu| ordenagailuei]][[Konputagailu-sare| sare]] bati kontatu bere burua erabiltzen dute, ematen duten eta haren zerbitzuak erabiltzen dituzte.
<Matxin>
Parek...
Glosarioa
Hiztegia
0/16 (0/16, 16)
179/217

3. Automatikoki itzulitako paragrafoa zuzentzeko, “Editorea” leihoan editatuko dugu.


OmegaT-2.1.8_2 :: probaX
Proiektua Editatu Joan Ikusi Tresnak Aukerak Laguntza
Editorea - Host.UTF8
{{referencias}}
El término """host""" es usado en [[informática]] para referirse a las [[computadora]]s conectados a una [[Red de computadoras|red]], que
<segment 0002> """ Host""" terminoa [[Informatika| informatikan]] erabiltzen da, [[Konputagailu-sare|sare]] bati konektatuta dauden eta sare horren zerbitzuak hornitzen eta erabiltzen dituzte [[Ordenagailu| ordenagailuak]] aipatzeko.
<end segment>
En general, los "hosts" son computadores monousuario o multiusuario que ofrecen
Machine Translation
""" Host""" terminoa[[Informatika| informatikan]] konektatuta[[Ordenagailu| ordenagailuei]][[Konputagailu-sare| sare]] bati kontatu bere burua erabiltzen dute, ematen duten eta haren zerbitzuak erabiltzen dituzte.
<Matxin>
Parek...
Glosarioa
Hiztegia
0/16 (0/16, 16)
179/221

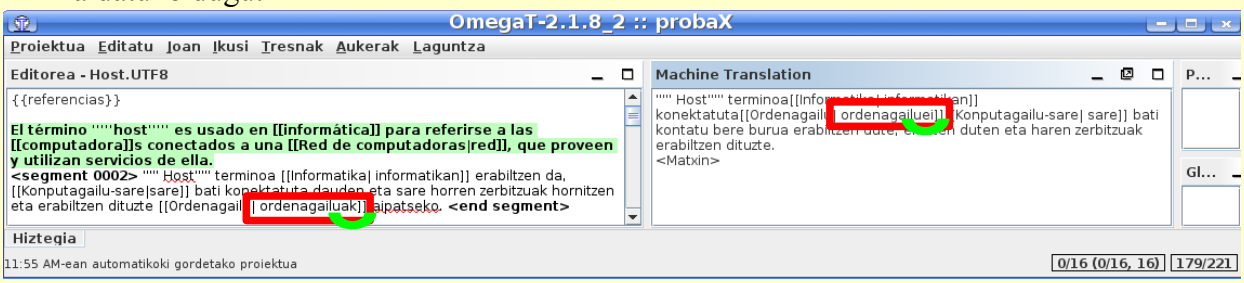
4. Bertan hitzak eta hitz-multzoak zuzendu, gehitu eta kendu daitezke, eta baita lekuz aldatu. Edozein testu-editore batean bezala lan egin dezakegu: Copy-Paste (Ctrl-C,Ctrl-V), Ctrl-X...

5. Gainera, ortografia-egiaztatzailea aktibatuta badugu, euskaraz okerrak diren hitzak gorriz azpimarratuko ditu. Hitz oker baten gainean eskuineko botoia sakatzen badugu zuzenketa posibleak eskaintzen dizkigu.

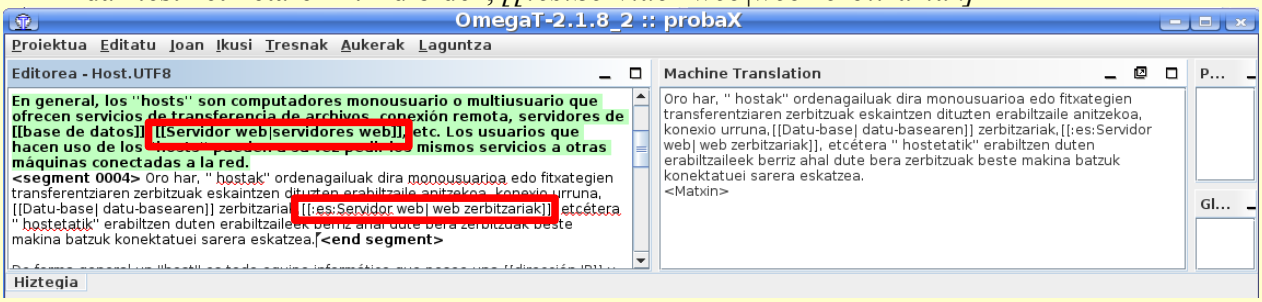

OmegaT-2.1.8_2 :: probaX
Proiektua Editatu Joan Ikusi Tresnak Aukerak Laguntza
Editorea - Host.UTF8
{{referencias}}
El término """host""" es usado en [[informática]] para referirse a las [[computadora]]s conectados a una [[Red de computadoras|red]], que proveen y utilizan
<segment 0002> """ Host""" terminoa [[Informatika| informatikan]] erabiltzen da, [[Konputagailu-sare|sare]] bati konektatuta dauden eta sare
hornitzen eta erabiltzen dituzte [[Ordenagailu| ordenagailuak]] aipatzeko.
<end segment>
Machine Translation
""" Host""" terminoa[[Informatika| informatikan]] konektatuta[[Ordenagailu| ordenagailuei]][[Konputagailu-sare| sare]] bati kontatu bere burua erabiltzen dute, ematen duten eta haren zerbitzuak erabiltzen dituzte.
<Matxin>
Pareka...
Glosari...
Hiztegia
11:35 AM-ean automatikoki gordetako proiektua
0/16 (0/16, 16)
179/221

6. Wikipediaren artikuluetako estekak kortxete bikoitza artean agertzen dira.
   - Estekaren erreferentzia eta erakutsi nahi den testua berdinak badira, kortxete artean testu hori agertuko da. Adibidez, *[[computadora]]s*
   - Estekaren erreferentzia eta erakutsi nahi den testua ez ba badira berdinak, kortxeteen artean agertzen dira. Adibidez: *[[Red de computadoras|red]]*
     - "|" ikurraren ezkerrean Wikipediako esteka, eta
     - "|" ikurraren eskuinean artikuluan erakusten den testua.
   - Espainierazko estekari dagokion euskarazko esteka badago, euskarazkoa agertuko da. Adibidez: *[[Konputagailu-sare|sare]]*, edo *[[Ordenagailu| ordenagailuei]]*

Euskaraz agertu beharreko testua zuzendu nahi badugu, "|" ikurraren eskuinean dagoena aldatuko dugu.



Espainierazko estekari dagokion euskarazko estekarik ez badago, espainierazkoa agertuko da ":es:" etiketarekin. Adibidez, *[[:es:Servidor web|web zerbitzariak]]*



7. Segmentu batean egindako aldaketak gordetzeko, segmentu horretatik atera behar da (*Enter* sakatuz, edo saguarekin klik bikoitza eginez beste segmentu batean)

   Adi ibili: fitxategiaren azkeneko segmentua editatu ondoren, Wikipedian gorde aurretik, segmentutik atera behar da. Bestela, aldaketak ez dira gordeko.

5. Wikipedian gorde

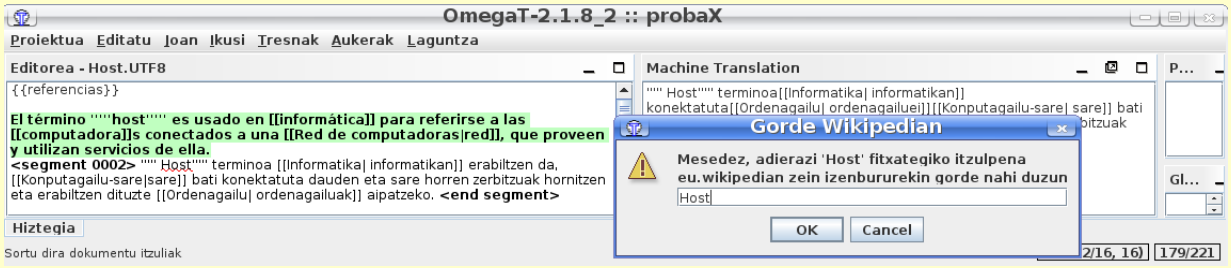
1. Ez ahaztu **azkeneko segmentuan egindako aldaketak gorde**tzen

2. Wikipedian gorde aurretik, **“dokumentu itzuliak” sortu**ta egon behar dira proiektuan:
   Proiektua -> Sortu dokumentu itzuliak (edo ***Ctrl+D)***

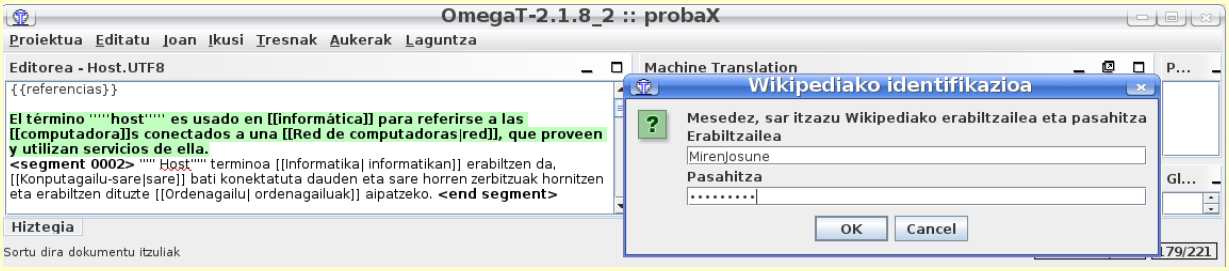
3. Behin dokumentuak sortuta eta aldaketak egin eta gero igo nahi bada, ez da beharrezkoa berriz dokumentuak sortzea – nahikoa da proiektua gordetzea:
   Proiektua -> Gorde (edo ***Ctrl+S)***

4. **Wikipedian gorde**tzeko
   Proiektua -> Gorde Wikipedian

   1. Sistemak artikulu hori zein izenarekin gordeko den eskatzen du.


OmegaT-2.1.8_2 :: probaX
Proiektua Editatu Joan Ikusi Tresnak Aukerak Laguntza
Editorea - Host.UTF8
Machine Translation
{{referencias}}
El término '''''host''''' es usado en [[informática]] para referirse a las [[computadora]]s conectados a una [[Red de computadoras|red]], que proveen y utilizan servicios de ella.
<segment 0002> '''' Host'''' terminoa [[Informatika| informatikan]] erabiltzen da, [[Konputagailu-sare|sare]] bati konektatuta dauden eta sare horren zerbitzuak hornitzen eta erabiltzen dituzte [[Ordenagailu| ordenagailuak]] aipatzeko. <end segment>
Gorde Wikipedian
Mesedez, adierazi 'Host' fitxategiko itzulpena eu.wikipedian zein izenbururekin gorde nahi duzun
Host
OK
Cancel
Hiztegia
Sortu dira dokumentu itzuliak
179/221

   2. Eta Wikipediako identifikazioa ere eskatzen du.
      Wikipedian erregistratzeko:
      http://eu.wikipedia.org/w/index.php?title=Berezi:SaioaHasi&type=signup&returnto=Azala


OmegaT-2.1.8_2 :: probaX
Proiektua Editatu Joan Ikusi Tresnak Aukerak Laguntza
Editorea - Host.UTF8
Machine Translation
{{referencias}}
El término '''''host''''' es usado en [[informática]] para referirse a las [[computadora]]s conectados a una [[Red de computadoras|red]], que proveen y utilizan servicios de ella.
<segment 0002> '''' Host'''' terminoa [[Informatika| informatikan]] erabiltzen da, [[Konputagailu-sare|sare]] bati konektatuta dauden eta sare horren zerbitzuak hornitzen eta erabiltzen dituzte [[Ordenagailu| ordenagailuak]] aipatzeko. <end segment>
Wikipediako identifikazioa
Mesedez, sar itzazu Wikipediako erabiltzailea eta pasahitza
Erabiltzailea
MirenJosune
Pasahitza
OK
Cancel
Hiztegia
Sortu dira dokumentu itzuliak
179/221

Itzuli beharreko artikuluak, zerrenda hauetatik aukera ditzakezu:

- Laburrak
- Ertainak eta luzeak

Tutorialak:

- Wikipediako formatuak ondo ezagutzen ez badituzu begiratu txuleta edo tutorial hau
- OmegaT-k funtzio eta aukera asko ditu, batzuk sinpleak eta beste batzuk dexente konplexuak. Horiei guztiei buruz, eta itzulpen-memoriei buruz, tutorialak badaude *omegat/docs/eu* azpidirektorioan. OmegaT-koek programa ondo erabiltzeko konplexutasuna dela eta tutorialak irakurtzea gomendatzen dute. OmegaT irekitzerakoan bertan tutorial oso azkar bat aurkezten du. Beraz, hemen azaldutakoaz gain, OmegaT ondo erabiltzeko eta bere aukera guztiei probetxu ateratzeko, komenigarria da tutorial zabalagoak irakurtzea. Adib. *http://leuce.com/translate/omegat.html*

Asiya nola konfiguratu

Asiya-ren eskuliburu teknikoak erabilerari buruzko gauza asko azaltzen baditu ere, erabiltzeko orduan zaila iruditu zitzaidan. Kontuan hartu behar da lehenengo aldiz ari nintzela, oraindik Lengoaia Naturalaren Prozesamendua ikasgaia eman gabe nuela, baino hala eta guztiz ere, Asiya-rekin lehenengo aldiz jorratzen den edozeinentzat lehenengo aldia zaila dela esango nuke.

Asiya instalatzeko eskuliburu teknikoaren pausoak baino ez dira jarraitu behar, adierazten dituen beste modulu guztiak ere instalatuz – horrek denbora asko eraman dezakela ikusi egin da. Ondoren, hurrengo pausoak jarraitzea gomendatzen da azkar abiaratzeko. Geroago posible izango da gehiago murgildu eta aukera desberdinekin trebatzeko eta inbestigatzeko.

Modu normalean exekutatzeko, Asiya-ri konfigurazio fitxategi bat pasa behar zaio, zeinetan datu guztiak emango zaizkion. Posiblea da fitxategian are datu gehiago ematea, edo alderantziz, exekutatzerakoan gauzak beste modu batean egiteko adieraztea komandoan bertan. Baino proiektu honetan zehar, hau iruditu zait modu azkarrena.

adibidea1.config

```
# ikurraz hasitako lerroak iruzkinak dira

# Testu mota, 'raw' (testu hutsa). guri 'xml'-rekin ez zaizkigu emaitza onak atera.
input=raw

# Jatorrizko eta helburu hizkuntzak.
srclang=es
trglang=eu

# Fitxategien kokapena. Metrika batzuek tarteko fitxategiak sortzen dituztenez,
# denak aparteko direktorio batean gordetzea gomendatzen da.
src=./fitx/adibideak.es
sys=./fitx/adibideak.MT.eu
ref=./fitx/adibideak.eu0

# Metrika multzoak. Hainbat aukera desberdin ezarri daitezke eta gero bat aukerau.
metrikak1=DP-Oc(*) DP-Ol(*) SP-Oc(*) SP-Op(*)
metrikak2=BLEU GTM-2 NIST -PER -TER -WER
metrikak3=Ol Pl Rl Fl

# Ebaluatuko den itzulpena eta erreferentzia. Bat baino gehiago jartzeko aukera ere bai.
hautagaia=adibideak.MT.eu
erreferentzia=adibideak.eu0
```

Ondoren, honelako komando batekin exekutatu daiteke. Aukerak daude “set”-ak aldatzeko, baino erosoena konfigurazio fitxategia aldatzea deritzot.

```
perl ./Asiya.pl -v -eval single -metric_set metrikak1 -system_set hautagaia -reference_set erreferentzia adibidea1.config
```

Emaitzak konfigurazioa dagoen direktorioaren azpiko *scores* sortuko den azpidirektorio batean gordeko dira.