eman ta zabal zazu

Universidad del País Vasco

Euskal Herriko Unibertsitatea

BiscayTrad

Euskara batua eta bizkaieraren arteko kode irekiko itzulpen automatikoko sistema.

Itziar Cortés Etxabe
Tutorea: Iñaki Alegria Loinaz

hap

Hizkuntzaren Azterketa eta Prozesamendua Masterreko titulua

lortzeko bukaerako proiektua

2011eko iraila

Saila: Lengoaia eta Sistema Informatikoak

Eskerrak,

*Eleka Ingeniaritza Linguistikoa*ri proiektu hau egiteko emandako aukeragatik eta bereziki *Garbiñe Aranbarri*ri emandako laguntza “linguistikoagatik”,

Francis Tyers eta *Mireia Ginestí*ri, *Apertium* gertutik ezagutzen laguntzeagatik,

eta *Iñaki Alegria*ri master-tesi hau zuzentzeagatik.

Aurkibidea

Irudien aurkibidea

Taulen aurkibidea

1 Sarrera

Euskarazko itzulpen automatikoaren esparruan, euskara batuaren eta euskalki baten artean egin den lehen itzulpen automatikoko proiektua aurkezten da master-tesi honetan. Euskara batua eta bizkaiera bi euskalki gisa hartu dira, alegia, hizkuntza ezberdinak izango balira bezala, eta kode irekiko itzulpen automatikoko sistema berri bat lortu da.

Oinarrian, gertuko hizkuntza-bikoteetarako pentsatuta dagoen *Apertium* teknologia erabili da; hizkuntza-baliabideei dagokienez, *Eleka*n eta *IXA*n euskara batuarekin eta bizkaierarekin egindako beste hainbat lanen oinarri izan diren baliabide linguistikoak erabili dira. Baliabide linguistiko hauek ahalik eta egokien ustiatzeko, morfologikoki konplexuak diren hizkuntzetarako egokiak diren egoera finituko transduktoreetan oinarritutako teknologiarekin konbinatu dira; datuak modu eraginkor eta erdi-automatikoan erabiltzea ahalbidetuz.

Proiektu hau *Eleka Ingeniaritza Linguistikoa*n garatu da. *Eleka*, momentu honetan, euskararekin lotutako hainbat itzulpen automatikoko proiektutan sartuta dago, hala nola, informazio morfologikoa eta estatistika konbinatuz euskara eta espainiera hizkuntza-bikotean arteko itzulpenak egiteko sistema hibrido bat sortzen.

*Eleka*k, *Opentrad*[1] itzulpen automatikoko sistema osatzen duten *Apertium* eta *Matxin* itzulpen motorren garapenean parte hartu zuen. Bertan, hainbat hizkuntza-bikoteen arteko itzulpenak egin daitezke, hasiera batean, espainiar estatuko hizkuntzen arteko itzulpenak egiteko sortua izan bazen ere, orain 20 hizkuntza-bikote baino gehiagoren arteko itzulpenak egiteko aukera ematen digu. Oinarrian, *Opentrad*, kode irekiko itzulpen automatikoa eskaintzen duen sistema irekia da.

Master-tesi honetan aurkezten den prototipoa, irailean Huelvan izango den *SEPLN2011* kongresuan aurkeztuko da *demo* moduan. Gehigarrien atalean, bertan aurkeztutako azalpen grafiko bat eta txosten bat atxikitu dira.

1 http://www.opentrad.com

2 Helburuak

Proiektu honen helburua, euskara batuaren eta bizkaieraren arteko kode irekiko itzultzaile bat sortzea da. Itzulpen automatikoaren arloan euskarak izan duen bilakaera eta dauden tresnak eta orokorrean hizkuntza ezberdinen arteko itzultzaile automatikoak garatzeko dauden kode irekiko aukerak ikusi ostean, *Apertium*en oinarritutako itzultzaile bat sortzea erabaki da. Teknologia honetan oinarrituz eta horrelako sistema bat sortu ondoren egiten den moduan, prototipo baten garapena aurkezten da; web-interfaze baten bidez sistema probatzeko aukera egongo da, eta testu hutsa itzultzeko aukeraz gain web orriak eta dokumentuak itzuli ahalko dira.

Proiektu hau kode irekikoa eta *Apertium*en oinarritutakoa izango denez, bertako gordailutik deskargatzeko moduan egongo da.

Erabiliko diren baliabide linguistikoei dagokienez, esan, hasiera batean pribatuak izango direla; hau da, *demo* moduan jarriko den webguneak erabiliko dituen datu linguistikoak pribatuak izango dira. *Apertium*eko gordailura igoko den guztia, ordea, librea izango denez, datu linguistiko hauek murriztu egingo dira *Matxin* proiektua garatzerakoan egin zen lexiko murrizketa oinarritzat hartuz.

Laburbilduz, master-tesi honen helburua, euskara batua eta bizkaiera bi hizkuntza balira bezala tratatuz, kode irekikoa izango den itzulpen automatikoko sistema bat sortu eta interneten edonork deskargatzeko moduan jartzea izango da. Gehigarrietan, esperimentu honetatik lortutako sistema instalatzeko beharrezko argibideak jarriko dira, edonork hau instalatzeko aukera izan dezan.

3 Aurrekariak

Proiektu honen testuinguruan kokatu asmoz, txosten honetan zehar aipatuko diren hainbat kontzepturi buruzko sarrera bat egingo da. Hala nola, itzulpen automatikoa, hizkuntzaren prozesamendurako erabiltzen diren teknologiak eta horrelako lanetarako erabili ohi diren baliabideei buruzko xehetasunak emango dira.

3.1 Itzulpen automatikoa

Itzulpen automatikoa, *Elhuyar zientzia eta teknologiaren hiztegi entziklopedikoa*n[2] jartzen duen moduan, *giza itzultzaile batek parte hartu gabe, hizkuntza batean idatzitako testu bat beste hizkuntza batean idatzitako testu baliokidera itzultzeko erabiltzen den teknika-multzoa da*. Teknika-multzo hau, azken urteotan gorakada izan duen arloa dela esan daiteke.

Urteak atzera eginez eta itzulpen automatikoaren hastapenei erreparatuz, honi buruz idatzi diren hainbat liburu aipa ditzakegu, horien artean (Beesley, Kenneth R., et al. 2003). Bestalde, eta urteotan zehar itzulpen automatikoaren inguruan egin diren lanen biltegi moduan, *Machine Translation Archive*[3] artxibategi elektronikoa aipa daiteke.

Itzulpen automatikoaren bilakaeran bi estrategia nagusi sortu dira: corpusetan oinarritutako itzulpen automatikoa (*CBMT, Corpus Based Machine Translation*) eta erregeletan oinarritutako itzulpen automatikoa (*RBMT, Rule Based Machine Translation*). Atal honetan, estrategia hauen xehetasunak eta hauen harira sortutako proiektuak azalduko dira.

Itzulpen automatikoari buruzko atal hau lantzeko (Mayor, A. 2007) eta (Labaka G., 2010) hartu dira erreferentziatzat.

2 http://zthiztegia.elhuyar.org/

3 http://www.mt-archive.info

3.1.1 CBMT. Corpusetan oinarritutako itzulpen automatikoa

Corpusetan oinarritutako itzulpen automatikoaren oinarri dira corpus parekatu elebidun eta handiak; honen adibide, *OLI* (*Ordenagailuz Lagundutako Itzulpeneko*) tresnekin lor daitezkeen itzulpen-memoriak dira. Ezagutza linguistikoa corpusetatik bertatik hartzen da, beraz, ez da erregeletan oinarritutako sistemetan moduan iturburu- eta xede-hizkuntzan adituak diren hizkuntzalariak behar sistema hauek martxan jartzeko. Sistema hauek, ordea, konputazionalki oso konplexuak eta karga handikoak diren prozedurak erabiltzen dituzte; corpus handiak izateaz gain, hauek makina ahaltsuetan manipulatzea beharrezkoa da.

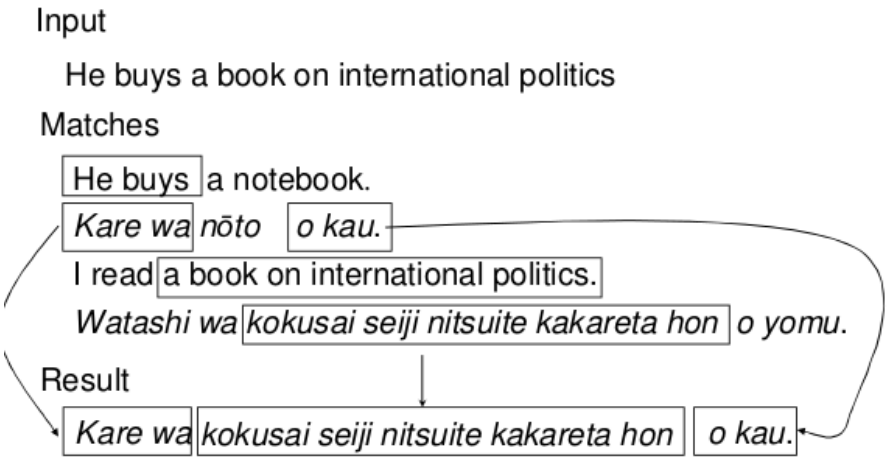
Mota honetako itzulpen automatikoa bi estrategia ezberdinetan sailka daiteke: adibideetan oinarritutako sistemak eta teknika estatistikoetan oinarritutako sistemak.

Adibideetan oinarritutako itzulpen automatikoko sistemak

Adibideetan oinarritutako sistema estatistikoa 1984an proposatu zen lehenengo aldiz (Nagao M., 1984). Sistema honen ikasketa metodoa esaldien antzekotasunean oinarritzen da. *Nagao*k, estrategia hau erabiltzeko egin beharreko hiru ataza nagusiak finkatu zituen:

- esaldi-zatiak adibide errealetako datu-base batekin parekatzea
- lotutako esaldi-zatiei dagozkien itzulpen-zatiak identifikatzea
- zati hauen konbinazioa itzulpen gisa ematea

Hona hemen Sato eta Nagaok 1990ean hau azaltzeko emandako adibide bat:



3.1. irudia: Adibideetan oinarritutako itzulpen automatikoko sistemaren funtzionamenduaren adibidea

Bertan, *input*aren itzulpena lortzeko, lehen eta bigarren esaldietako zati egokiak identifikatzen dira. Emaitza osatzeko, zati hauek egoki ordenatu eta esaldia sortzen da; itzulpena, beraz, zati horien konbinaziotik sortzen da.

Adibideetan oinarritutako sistemetan, ikasketa prozesuan, iturburu- eta xede-hizkuntzako esaldiak parekatu behar dira: bertan, esaldiko zati bakoitza zein beste zatiren itzulpena den identifikatu behar da. Horretarako, hainbat baliabide linguistiko erabil daitezke, adibidez, *Pangloss*[4] sistema (Brown, 2000; Nirenburg et al., 1994; Brown, 1996) sortzeko erabilitakoak: hiztegi elebidunak, analizatzaile morfologikoak eta sinonimoen zerrendak.

Adibideetan oinarritutako itzulpen automatikoko estrategia erregeletan oinarritutakoa baino egokiagoa izan daiteke kasu batzuetan, adibidez, itzulpen-erregelak sortzea oso zaila denean. Hala ere, zalantzan jarri izan da adibideetan oinarritutako

4 Pangloss interlingua bidezko sistema Southern California, New Mexico State eta Carnegie Mellon Unibertsitateetan garatu zen (Nirenburg, 1995).

itzulpen automatikoak bakarrik, itzulpen-prozesu osoa burutu zezakeenik, eta gai honen inguruko saiakera gutxi egin dira.

Estatistikan oinarritutako itzulpen automatikoko sistemak

Estatistikan oinarritutako itzulpen automatikoa, gehien ikertzen den itzulpen automatikoko sistema da; azkenaldian, gai honen inguruan argitaratu den artikulu kopuruak igoera nabarmena izan du.

Orokorrean, estatistika hutsean oinarritutako sistemek emaitzen kalitatea hobetzeko lanetan eragozpenak izan ditzakete. Adibidez, domeinu ezberdinetan ematen dituzten emaitzak kalitate ezberdinekoak izaten dira. Gainera lengoaia naturalaren prozesamenduan egindako ikerketez ez dute etekinik ateratzen eta entrenamendurako erabilitako corpus tamaina handitu arren, oso zaila da emaitzen kalitatea hobetzea. Horregatik, sistema estatistiko hutsak erabiltzeaz gain beste teknika batzuekin konbinatutako sistemekin esperimentuak egin dira.

Itzulpen automatikoko sistema estatistiko hauen helburua, oinarrian, bolumen handiko testu elebidunak hartu eta bertatik lortutako informazio estatistikoan oinarrituz, itzulpen sistema bat sortzea da. Beraz, funtsean, ez dago bestelako baliabide linguistikoen beharrik, baina morfologikoki konplexuak diren hizkuntzekin lan egiterako orduan interesgarria da etiketatzaile morfologiko batek emandako emaitzak ikasketa prozesuan sartzea; lema bakoitzak forma ezberdin askotan flexionatu daiteke eta honek iturburu- eta xede-hizkuntzako hitzen parekatzea zailtzen du.

Corpuseko esaldi guztiei dagokien prozesamendu linguistikoa eginez, hau da, morfologikoki etiketatuz, faktoreekin lan egiten duten itzulpen modeloak sor daitezke. Adibide adibide gisa ditugu ingelesa eta txekiarrarekin egindako esperimentua (Bojar O., 2007) eta ingelesa eta alemaniarrarekin egindakoa (Holmqvist et al., 2007) .

Corpusen tamaina handia izanik eta testu honi guztiari gehitzen zaion informazio morfologikoa kontutan hartuta, entrenamenduaren ondoren lortutako modeloek sistemaren eraginkortasunaren galera ekartzen dezake. Beraz, morfologikoki aberatsak diren hizkuntzen arteko itzulpenak egiteko informazio morfologikoa eta estatistika

konbinatu nahi badira, corpusaren tamaina ditugun baliabideen arabera mugatu beharko da.

Corpusetan oinarritutako itzulpen automatikoko sistemetan, ezagutza linguistikorik gabe sistema entrenatu daitekeen arren, informazio morfologikoa konbinatu daitekeela ikusi dugu. Aurrerago azalduko den 3.1. taulan, euskararako egindako esperimentuen emaitzak ikus daitezke: estatistika hutsean oinarritutako sistemak, erregeletan oinarritutakoak eta estatistika eta informazio morfologikoan oinarritutakoen arteko ebaluazio bat da.

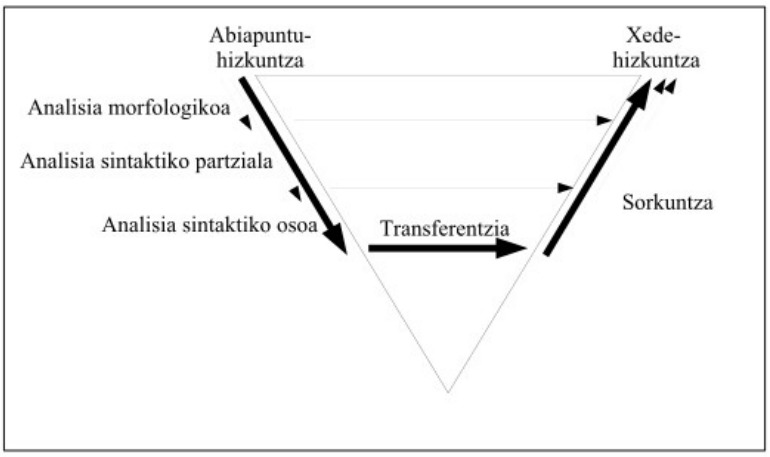
(Labaka G., 2009)n itzulpen automatikoko sistema hibridoak nolakoak izan daitezkeen azaltzen du: itzulpen-baliabideen aberasketa, itzulpen motor ezberdinen arteko hibridazioa eta post-edizio automatiko bidezko hibridazioa. Aipatutako esperimentuetan egindako hibridazio mota, lehenengoa da, itzulpen-baliabideak aberastuz sistema estatistikoa entrenatzea hain zuzen ere.

3.1.2 RBMT. Erregeletan oinarritutako itzulpen automatikoa

Corpusetan oinarrituta sistemetan ez bezala, itzultzaile hauen garapenerako beharrezkoa da iturburu- eta xede-hizkuntzako ezagutza linguistikoa izatea, bi hizkuntzen arteko loturak erregelen bidez kodetzeko; honek, bi hizkuntzetan adituak diren hizkuntzalarien beharra suposatzen du. Honez gain, horrelako sistemak martxan jartzeko orduan beste zailtasun batzuk aurkitu ohi dira (Hutchins J., 2006), hala nola, erregela gramatikalen konplexutasuna, hiztegien konplexutasuna eta hautapen-murrizpenak, kolokazioak, aditz-esapideak, izenordainak, esaldi luzeen egitura konplexua eta beste zenbait arazo semantiko. Sortu behar diren modulu eta hizkuntza-baliabideak konplexuak eta sortzen nekezak diren arren, sistema hauetan, sortutako baliabideak beste RBMT proiektuetarako berriz erabil daitezke (Bond F. et al., 2011). Bestalde, horrelako sistema konplexu bat garatzeko garaian, itzulpenaren eremua gai jakin batera mugatuta garapena errazagoa izateaz gain, lortzen den emaitza ere hobea izango da.

Erregeletan oinarrituz itzulpen automatikoko sistemen garapenerako hiru modu ezberdin aurki ditzakegu: itzulpen zuzena egiten duten sistemak, transferentzian

oinarritutako sistemak eta interlingua bidezko hurbilpenak. Hauek azaltzeko, hona hemen, Vauquoisek erabili zuen diagrama:



3.2. irudia: Vauquoisen triangelua

Diagrama honetan, erregeletan oinarrituz egin daitezkeen hiru estrategiekin, iturburu- eta xede-hizkuntzetan egin beharko diren analisi eta sorkuntza lanak irudikatzen ditu. Adibidez, itzulpen zuzena egiten denean azaleko analisi eta sorkuntza bat erabiliko da transferentzia erregela konplexuekin konbinatuz; *interlingua* bidezko hurbilpen batean, aldiz, transferentzia erregelak ez dira erabiliko. Analisirako eta sorkuntzarako erabiliko diren datuak, ordea, konplexuagoak izango dira eta lan eskerga izango da datu hauek prestatu eta era eraginkor batean martxan jartzea. Aipatutako bi teknika hauen artean, transferentzian oinarritutako beste sistemak aurki daitezke, analisian eta sorkuntzan erabiliko den informazioa eta transferentzia erregelen arteko oreka bat lortu asmoz. Oreka hori, iturburu- eta xede-hizkuntzen arteko hurbiltasunaren araberakoa izango da. Oso ezberdinak diren hizkuntza-bikoteek, transferentzia landuago bat beharko dute; gertukoek aldiz, gutxiago,

Transferentzian oinarritutako sistemen artean *Apertium* kode irekiko itzulpen automatikoko motorra aztertuko da. Aipatu bezala, erregeletan oinarritutako sistema bat da eta kasu honetan, gertukoak diren hizkuntza-bikoteen arteko itzulpenak egiteko sortu zen; proiektuaren hastapenetan, espainiar estatuko hiru hizkuntzen arteko itzulpenak egiteko sortu zen: espainiera, galegoa eta katalana. *Apertium*en erabiltzen den itzulpen

automatikoko estrategia (Canals-Marote et al. 2001; Garrido-Alenda et al. 2003)n deskribatuta dago. Esan bezala, motor honen arkitektura azaleko transferentzian oinarritutako zortzi moduluk osatzen dute. (Corbí-Bellot et al., 2005)en jartzen duen moduan, proposatzen den estrategia nahikoa da espainiera, katalana eta galegoaren arteko itzulpen automatikoak emaitza txukunak eman ditzan.

> *"this strategy is sufficient to achieve a reasonable translation quality between related languages such as es, ca or gl. "*

Hizkuntza-bikote hauetarako erabiltzeaz gain, gaur egun, *Apertium*eko gordailuan hainbat hizkuntza-bikoteen arteko itzulpenak egiteko baliabideak aurki ditzakegu. Hauen artean, euskaratik espainierara itzulpenak egiteko sistema aurki dezakegu, *Erdaratu*[5] izeneko proiektuaren barne. Gordailuko *incubator* direktorioan, aldiz, euskarazko baliabideak erabiliz sortutako beste hainbat sistema aurki daitezke, hala nola, euskara-ingelesa eta euskara-frantsesa hizkuntza-bikoteen arteko itzulpen automatikoko sistemak.

Apertium, itzulpen automatikoa eskaintzen duen kode irekiko *Opentrad* sistemaren parte da. *Opentrad* proiektuaren helburua, estatu espainiarreko hizkuntza nagusietarako kode irekiko itzulpen automatikoko sistemak sortzea zen. Helburu honekin garatu zen *Apertium*, gertukoak diren hizkuntza-bikoteentzat, transferentzia sintaktiko partziala erabiliz.

Matxin, *Opentrad*ek barne hartzen duen bigarren teknologia da, espainiera euskara norabiderako sortua, transferentzia sintaktiko osoa burutuz. Espainieratik euskarara itzulpenak egiteko prestatu zen arren, hizkuntzarekiko independente izateko egin beharreko aldaketen azterketa lana egina dago (Mayor, A., 2009). Beraz, egunen batean *Matxin* beste hizkuntzetarako erabili ahal izateko egin beharreko aldaketen azterketa egina dago.

Matxin, espainieratik euskarara itzulpenak egiten dituen lehen itzulpen automatikoko sistema da. Proiektu honen garapena, *IXA*ko informatikari eta hizkuntzalarien artean garatu zen, *Matxin* prototipotik abiatuz sistemaren birdiseinatze

5 http://www.erdaratu.eu

eta birprogramatze bat eginda (Mayor, A., 2007). Hauek dira informatikari eta hizkuntzalariek elkarlanean burutu zituzten ataza nagusiak:

- Sistemak prozesatzen dituen datu-egituren diseinu berria.
- Estandarizazioa. Moduluen arteko datu-fluxuaren, baliabide linguistikoen eta erregelen formatuak XML lengoaiaz definitu genituen. Hiztegien formatua proiektuko espezifikaziotara egokitu genuen eta hiztegien kontsultarako Apertiumeko egoera finituetako transduktoreak erabili. Honetarako, aurrerako azalduko den *lttoolbox* erreminta erabiltzen du normalean *Apertium*ek.
- Preposizio eta funtzio sintaktikoen itzulpenaren desanbiguaziorako moduluaren eraikuntza.
- Datu linguistikoen hedapena eta egiaztapena.
- *Freeling*en hobekuntza, batez ere mendekotasun-analizatzailearen eraikuntzarekin lotuta.
- Kodearen berriro inplementatzea C++ lengoaian.
- Erroreen arazketa eta zuzenketa.
- Banaketa kode irekiko lizentzia batekin

(Mayor, A., 2007)n sortutako *Matxin* prototipo horretatik abiatuz, ordea, prototipo horretan oinarritutako lan gehiago ere egin dira *IXA* ikerketa taldean eta *Eleka Ingeniaritza Linguistikoa*n.

*IXA*n *Matxin*en prototipo hartatik abiatuz, informatika eremuarekin lotutako itzulpenak egiteko beste prototipo bat prestatu dute, irailean Huelvan ospatuko den *SEPLN2011* kongresuan erakusteko.

*Eleka*n, aldiz, *Matxin* eta orokorrean itzulpen automatikoari dagokionez, hainbat lan ezberdin garatzen ari gara. Hurrengo atalean, lan hauen azalpen txiki bat ematen da.

3.1.3 *Eleka* eta itzulpen automatikoa

Momentu honetan Elekan, erregeletan eta estatistikan oinarritutako itzulpen automatikoko lanak egiten dira; estrategia bakoitza bere aldetik landu izan den arren, orain hauen konbinaketekin esperimentuak egiten ari gara. Alde batetik, *Matxin* eta itzulpen sistema estatistikoaren arteko konbinaketa batean lanean dihardugu espainieratik euskararako norabidean, batak zein besteak ematen dituen emaitzak hizkuntza-modelo batekin ebaluatuz emaitza onenak automatikoki lortu asmoz. Beste alde batetik, sistema estatistikoaren hibridazioarekin lanean ari gara, sistema hauek ikasteko oinarri gisa erabiltzen diren corpusak morfologikoki etiketatuz eta informazio guzti horrekin ikasketa automatikoa gauzatuz. Hau da, aurretik aipatutako faktoreekin lan egiten duten itzulpen modeloekin esperimentuak egiten gabiltza.

Esperimentu hauen harira eta sistema ezberdinen ebaluazioa egiteko, giza itzultzaile batek ausaz aukeratutako eta automatikoki sistema ezberdinek itzulitako ehun esaldien itzulpenak aztertu ditu. Sistema ezberdinen arteko ebaluazioa, giza-itzultzailearen iritzia eta ebaluazio automatikoaren emaitza kontutan hartuz egin da. Giza-itzultzaileak, itzulpenen ebaluazioa egiteko, hauen zuzentasuna eta ulermena neurtu ditu; ebaluazio automatikorako, aldiz, BLEU ebaluatzeko metodoa erabili da erreferentziazko corpus baten kontra (Papineni et al., 2002). Hurrengo taulan, jasotako emaitzen laburpena azaltzen da:

	Giza-ebaluazioa		**Ebaluazio automatikoa**
	Ulermena (1-4)	**Zuzentasuna (1-8)**	**BLEU**
(1) SMT + Informazio morfologikoa I	**2.52**	**4.48**	**0.6615**
(2) SMT + Informazio morfologikoa II	2.40	4.27	0.5105
(3) Baseline++ (SMT)	2.28	4.34	0.4428
(4) Baseline (SMT)	1.89	3.15	0.3501
(5) Baseline+ (SMT)	1.85	3.37	0.3348
(6) Google Translate	1.54	2.41	0.0752
(7) Matxin (RBMT)	2.05	3.29	0.0703

3.1. taula: Itzulpen automatikoko sistema ezberdinen arteko ebaluazioa. Giza-ebaluazioa eta ebaluazio automatikoa

Espainieraz idatzitako 100 esaldi hartuta, taulan ikusten den moduan, 7 sistema ezberdinekin itzuli dira. Iturburu-hizkuntzako esaldi bakoitza eta hauen 7 itzulpenak dokumentu batean bildu eta giza-itzultzaile batek itzulpen guztiak banan-banan ebaluatu ditu. Ebaluazioa egiten zuen bitartean, ez zekien esaldi bakoitza zein itzulpen sistemari zegokion. Horrela, sistema ezagunengatik zuen iritziak ez zuen bere emaitzetan eraginik izango. Ebaluaziorako, giza-itzultzaileak bi irizpide jarraitu zituen: batetik laura baloratu zuen esaldiaren ulermena, eta batetik zortzira ebaluatu zuen esaldiaren zuzentasuna.

Sistemei dagokienez, *Baseline* izenekoa, *Eleka*n daukagun oinarrizko SMT sistema da. Hau hobetuz joan da, bere bertsioak diren *Baseline*+ eta *Baseline*++ sortuz.

Emaitzak erakusteko orduan, BLEU balioaren arabera egin da sailkapena. Hala ere, taula honetan argi ikus daitekeen bezala, giza-itzultzailearen ebaluazioa eta ebaluazio automatikoaren emaitza ez datoz guztiz bat. BLEU balioaren arabera, *Matxin* azken postuan gelditu da *Google Translate*ren sistema estatistikoaren atzetik. Bestalde, giza-itzultzailearen ebaluazioaren araberako ordenaketa bat egiten bada, *Matxin*

laugarren postuan egongo litzateke eta *Google Translate* azken postuan geratuko litzateke.

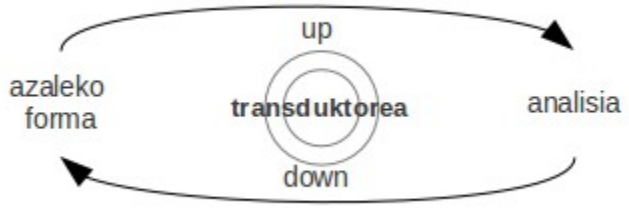
Horrelako ebaluazioak egitea interesgarria da egiten diren esperimentuen emaitzak ulerkorrak eta zuzenak diren jakiteko. Baita erabilitako tekniken artean emaitzarik onenak ematen dituztenak detektatzeko ere.

3.2 Hizkuntzaren prozesamendurako teknologiak

Hizkuntza naturalaren prozesamenduan, analisi eta sorkuntza morfologikorako erabil daitezkeen teknologia eta software ezberdinak aurki ditzakegu. (Beesley et al. 2003) liburuak egoera-finituko morfologia zer den eta nola erabil daitekeen azaltzen du. Liburu hau, era berean, *Koskenniem*ek proposatutako bi-mailatako morfologian dauka oinarria.

Atal honetan, hizkuntza naturalaren prozesamenduan oinarri diren ideia hauek erabiliz sortutako erremintak azalduko dira.

Lttoolbox, hizkuntzaren prozesamendu lexikalerako erabiltzen den erreminta da: analisi eta sorkuntza morfologikoa ahalbidetzen duen tresna da. Erreminta honekin, egoera finituko transduktoreak (*FST, Finite State Transducers*) sortzen dira, non hitzen analisi morfologikoa lortzeaz gain alderantzizko prozesua ere lor daitekeen. Analisi morfologikoaz hitz egitean, hitz batek duen lema eta bere kategoria morfologiko posible guztiak lortzeaz ari gara. Alderantziz den kasuetan, sorkuntza morfologikoan, lema eta analisi bat abiapuntutzat hartuz, azaleko forma edo hitz bat lortzea da helburua.



3.3. irudia: Transduktorearen funtzionamendu orokorra

*Apertium*ek analisi eta sorkuntza morfologikoa egiteko *lttoolbox* erreminta erabiltzen du. Honek hizkuntzalariek eskuz landutako hiztegiak konpilatu eta egoera-finituko transduktoreetan bihurtzen ditu. Hiztegiak sortzerakoan, sarrera bakoitzaren ezkerreko (l) eta eskuineko (r) zatia definitzen dira, adibidez:

```
<e>
      <p>
        <l>berri</l>
        <r>berri<s n="adj"/><s n="izo"/></r>
      </p>
      <par n="ADJK_40"/>
</e>
```

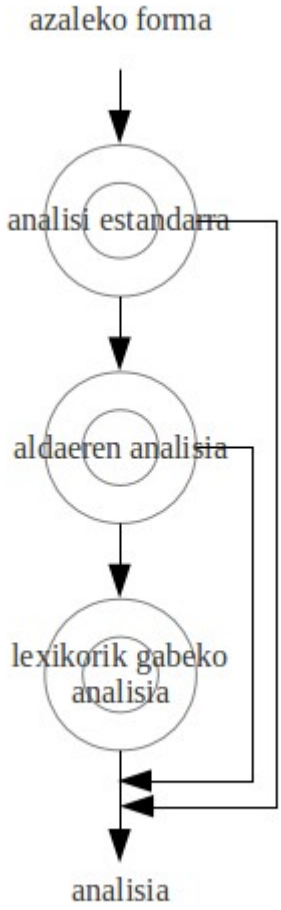
Ezkerreko aldean azaleko forma definitzen da, eta eskuinekoan analisi morfologikoa. Hiztegi hau left-to-right (LR) edo right-to-left (RL) norabideetan konpilatu ahal izango da. LR norabidean konpilatutako hiztegiak analisirako balioko du; RL norabidekoak sorkuntzarako.

Apertiumen euskarazko hiztegi bat daukagula suposatuz, horrela konpilatuko litzateke analizatzaile morfologikoa:

```
lt-comp lr apertium-eu-eu_bis.eu.dix eu.bin
```

Hiztegi hauek sakonago aztertuta, hizkuntza deskribatzen duten gramatikak direla esan daiteke. Hiztegiko sarrera bakoitza, adibidean *berri*, paradigma bati lotuta dago (*ADJK_40*). Paradigma honek, honekin loturik dauden sarrera guztien kasu posibleak bilduko ditu, jarraitze-klase bat izango balitz bezala.

(Alegria, I. et al., 1996)n euskararen analizatzaile morfologikoa egiteko erabilitako datu base lexikalaren ezaugarriak eta hau prozesatzeko erabilitako modua azaltzen da eta nola *Xerox* (Beesley et al. 2002) erreminta erabiliz, datu-base lexikaleko informazioa transduktore ezberdinetan bihurtzen den. Baita bi-mailatako morfologian oinarrituz landutako erregelen xehetasunak ere. Analisi morfologikoaren prozesua ere azaltzen da bertan, analisi bat lortzeko zein modulu erabiltzen diren azalduz.



3.4. irudia: Euskararen analizatzaile morfoloaikoa

Irudi honetan, euskarazko azaleko forma baten analisia lortzeko beharrezko hiru pausuak irudikatu dira. Lehendabizi, analisi estandarra egiten da. Emaitzarik lortu ez bada, aldaera linguistiko bat ote den begiratzen da. Azkenik, oraindik emaitzarik lortu ez bada, lexikorik gabeko analisi bat egiten da. Beraz, fluxu hau jarraitzeko hiru transduktore erabiltzen dira: sarrera estandarren transduktorea, sarreren aldaerak analizatzen dituen transduktorea eta azkenik, lexikorik gabeko analisiak ematen dituen transduktorea.

Analizatzaile morfologiko honetan oinarrituz sortutako lehen bi aplikazio komertzialak *Xuxen zuzentzaile ortografikoa*[6] eta *EUSLEM* lemmatizatzaile-etiketatzailea (Aduriz et al., 95) sortu ziren; beraz, hasiera batean euskararako sortu ziren prozesamendu automatikoko tresnak *Xerox* erremintan zuten oinarria. Erreminta honen erabilera, ordea, mugatua da. Daukan lizentzia dela eta *Xerox* erreminta erabiliz sortutako edozein fitxategi erabiltzen duen produktu batekin etekina ateraz gero, etekin horren portzentaje bat *Xerox* enpresari ordaindu behar zaio. Bilatzaileetan integratzeko ere mugak ditu, ezin baitira *Xerox* erremintarekin sortutako transduktoreak bilatzaileen garapenean erabili.

Xerox alde batera utzita, transduktoreak modu eraginkorrean maneiatzeko beste tresna bat aurkezten da txosten honetan. *Foma* (Hulden M., 2009), egoera-finituko transduktoreak sortzeko konpilatzaile, programazio lengoaia eta C liburutegia da. *Xerox*ekin sortutako transduktoreekin bateragarria eta GPL lizentziaduna.

Bateragarritasun hau dela eta, euskarako sortutako hainbat baliabideetan transduktoreak maneiatzeko erreminten migrazioa egin da. Momentu honetan, *Eleka*n lanean ari garen *Eletagger* proiektuak erabiltzen diren oinarrizko transduktoreak, *Xerox*en formatutik *Foma*kora pasatakoak dira; baita zuzenean *Foma*n sortutakoak ere. Gainera, azken aldian egindako hainbat erremintetan *Foma* erabiliz sortu dira beharrezko transduktoreak. Adibide gisa, *XuxenB* bizkaierarako zuzentzaile ortografikoa (Alegria I., et al., 2010) garatzeko *Foma* erremintarekin sortutako transduktoreak erabili ziren.

*Lttoolbox*ek erabiltzen dituen hiztegiak azaltzeko erabilitako adibide berdinarekin jarraituz, horrelakoa izango litzateke *berri* hitzaren definizioa *Foma*k erabiltzen dituen gramatika batean:

```
berri<adj>:berri  ADJK_40;
```

Master-tesi honen garapenean zehar, morfologikoki konplexuak diren egoera-finituko transduktoreekin lan egiteko beste teknologia bat ezagutzeko aukera sortu da:

6 http://www.xuxen.net/

HFST, Helsinki Finite-State Transducer Technology (Lindén, K., et al., 2009). Aurrerago zehatzago azalduko den arren, esan, transduktoreekin lan egiterako orduan ezagutzen ez genituen aukerak eskaini dizkigula *sorftware* honek. Gainera, *HFST*ko transduktoreak *Foma* erabiliz sortutakoekin konbinatu eta hauek manipulatzeko aukera ematen du.

*HFST*k *Apertium*ekin lan egiteko *hfst-proc* exekutagarria erabiltzen du, eta honek, transduktoreari dei egiterako orduan, sarrerako letra larriz dauden ala ez kontutan hartu behar den esateko aukera ematen du. *Foma* erabiliz sortutako transduktoreetan, hitzen hasierako letrak larria izan behar duenean, analisian hori islatuta geratzeko datuak gramatiketan txertatu behar izan dira. Transduktoreak zuzenean *HFST* erremintarekin sortu balira, hau ez litzateke beharrezkoa izango, analisirako behintzat. Adibide moduan, hona hemen *hfst-proc* exekutagarriarekin egindako proba batzuk:

```
$ echo "ETXEKO" | hfst-proc -c eu-eu_bis.automorf.hfst
^ETXEKO/ETXEKO$
echo "ETXEKO" | hfst-proc eu-eu_bis.automorf.hfst
^ETXEKO/ETXEKO<n>$
```

Esperimentu honetarako *Foma* erremintarekin sortutako transduktoreak erabili dira, eta ondoren, erreminten ezberdintasunak ikusi eta proba gehiago egiteko *HFST* formatura bihurtu dira.

3.3 Euskara batuko eta bizkaierako baliabideak

Proiektuaren analisi eta garapenean sartu aurretik, euskara batuarekin eta bizkaierarekin lan linguistikoak egiteko dauden baliabideen errepaso bat egingo da. Lanean hasi baino lehen, proiektua gauzatzeko erabil daitezkeen hizkuntzarekiko mendekoak diren baliabideak zein diren aztertuko da.

Alde batetik, *Euskararen Datu-Base Lexikala* dago, *EDBL* (Aduriz, I. et al. 1998), euskara inguruan egiten diren hainbat erremintetan erabiltzen den oinarrizko

informazioa gordeta dagoen datu-basea. Bertan gordetako jakintza etengabe eguneratzen da eta nagusiki, *Hiztegi batua*n oinarrituta egiten da lan. Beste hiztegi batzuk ere erabili ohi dira datu-basea aberasteko, hala nola, *Elhuyar Hiztegi Txikia*, *UZEI Sinonimo Hiztegia* edo *Euskal Hiztegia*.

EDBL euskararen prozesamendu automatikorako oinarri gisa erabiltzen da eta orain arte hainbat lanetarako erabili da, esaterako, *Morfeus* analizatzaile morfologikoa egiteko, *Euslem* lematizatzailea, *Xuxen zuzentzaile ortografikoa* eta orain *Eleka*n garapen prozesuan dagoen *Eletagger* etiketatzaile morfosintaktikoaren garapenean ere oinarrizko zatia da.

Bestalde, bizkaierarako ere bada oinarri lexikalik: *BiDBL*, *Bizkaieraren Datu-Base Lexikala*. Hau, euskara batuaren deskribapen morfologikoan eta bere datu-basearen egituran oinarrituz sortua izan zen, Labayru institutuak bizkaiera estandarrerako ezarritako gidalerroetan oinarrituz. Oraingoz, datu-base hau erabiliz egindako lan bakarra *XuxenB* izan da, bizkaierako zuzentzaile ortografikoa (Alegria, I. et al. 2010). Artikulu honetan, bizkaierako datu-basearen xehetasunak eta informazioa *Foma*rekin nola maneiatu azaltzen da. Esaterako, datu-baseak dituen sarrera mota ezberdinak zerrendatzen ditu: sarrera lexikalak, deklinatutako aditz formak eta bestelako morfemak. Datu-baseko informazio honen esportazioari esker, bizkaieraren deskribapen lexikoa lortzen da; hau izango da, ondoren, *Foma* erabiliz maneiatu ahal izango den informazioa.

(Alegria, I. et al. 2010)ekin jarraituz, esan, orokorrean *Xerox* erabiltzetik *Foma* erabiltzera egin behar izan den aldaketarik garrantzitsuena erregela morfofonologikoen berridazketa izan dela. Erabiltzen ziren erregela paraleloak alde batera utzi, eta erregela sekuentzialak sortu behar izan ziren. Erregela hauek sortzean, kontutan hartu behar da erregela non kokatzen den, hau da, erregelen ordena garrantzitsua da, idatzitako ordena berean exekutatzen baitira. Erregela paraleloak kodetzean, beste erregela guztiak kontutan hartu behar baziren ere, sekuentzialak kodetzea sinpleagoa da; aurretik dauden erregelak bakarrik hartu behar baitira kontutan.

3.4 FSTak eta morfologia

FSTak morfologian nola aplikatzen diren ikusteko, *Foma*ko wikian datu morfologikoak maneiatzeko dagoen eskuliburuko[7] ideia nagusiak azalduko dira.

Analisi edo sorkuntza morfologikorako erabiliko den FST bat sortzeko bi elementu garrantzitsu behar dira: lexikoia eta erregelak.

Lexikoiak, hizkuntzaren deskribapena zehazten du, hau da, sarrera gisa hizkuntza horretako lemez eta morfemez osatutako, eta gramatikalki zuzenak diren formak besterik ez ditu onartzen. Lexikoiarekin bakarrik sortutako transduktore batek, ordea, ez luke euskarak dituen forma guztiak egoki analizatzeko modurik izango; erregelak behar ditugu. Har ditzagun adibide moduan *baserritar* eta *plazer* lemak. Biei, atzetik artikulua (*a* morfema) eranstean forma hauek lor daitezke:

```
baserritar + a = baserritara
plazer + a = plazera
```

Euskararako, *baserritara* forma ez da egokia. Beraz, erregela fonologiko bat gehitu beharko genioke forma egokia lortzeko. Euskararako prestatuta daukagun lexikoian, ordea, bi-mailatako formak erabiltzen dira horrelako konbinaketak egiteko. Hau da, *baserritar* sarreraren kasuan, bere bi-mailatako forma *baserritaR* izango litzateke; *plazer* sarreraren bi-mailatako forma, aldiz, *plazer* izango litzateke. Honenbestez, forma egokiak sortzeko erregelak honen arabera definituko dira:

```
baserritaR + a = baserritarra
plazer + a = plazera
```

Adibidean ikusten den moduan, *R* letra larria *rr*gatik ordezkatuko da, euskaraz zuzena den forma bat lortzeko. Horrelako erregelak dagoeneko sortuta daude euskara baturako; baita bizkaierarako ere. Hona hemen, bizkaierako erregela baten adibide bat:

```
A -> e || \i _ MM 5 (E) OpenVowel ;
```

Bizkaierako analisi morfologikoak lortzeko, euskara batuko erregelak aplikatzeaz gain, bizkaierarako sortutako erregelak ere aplikatzen dira. Azaldutako

7 http://foma.sourceforge.net/dokuwiki/doku.php?id=wiki:morphtutorial

erregela horrek, adibidez, euskara batuko *alaba* forma onartu beharrean, bizkaierako *alabea* forma onartzen du (alabA+5a:alabea)[8].

Euskararen moduan morfologikoki konplexuak diren hizkuntzekin lan egiteko egokia da *Foma*; baita aurreko ataletan aipatutako *HFST* ere.

8 5 marka edo diakritikoa gehitu da bizkaieraz deklinabide berezia duten zenbait atzizkitan, horrela erregelen aplikazioa modu kontrolatuagoan egin ahal izateko

4 Analisia

Tresna garatzeko oinarri gisa *Apertium* erabiliko den arren, garapena egiteko bi aukera ezberdin aztertu dira. Hurrengo lerroetan bi aukera hauek zehatzago azalduko dira, eta hauek aztertu ondoren hartutako erabakia azalduko da.

4.1 Lehenengo aukera: *Apertium*en bikote berri bat sortu

Lehenengo aukera, *Apertium* orain arte *Eleka*n erabili izan dugun moduan erabiltzea da. Hizkuntza-bikote berri bat sortzeko behar diren baliabideak eskuz sortuz, adibidez hiztegiak, garapen arrunt bat egiten.

Horretarako komunitatean garatutako *Erdaratu* proiektua oinarritzat hartu eta bertako baliabide linguistikoak aprobetxatuz, euskara batua eta bizkaieraren arteko itzultzailea sortzea aztertu da. Erregeletan oinarritutako itzulpen automatikoko sistemetan, lehen esan moduan, erabiltzen diren datu linguistikoak berrerabil baitaitezke. *Erdaratu* proiektuan erabiltzen diren baliabide linguistikoak libreak dira, *Matxin* proiektuan eta *FreeLing* proiektuetan libre jarritako euskarazko eta espainierazko hiztegiak oinarritzat hartuz sortutako datuak dituelako.

Apertium erabiliz hizkuntza-bikote berri bat sortzeko beharrezkoak dira bi hiztegi elebakar (iturburu-/helburu-hizkuntzakoak) eta hiztegi elebidun bat (bi hizkuntzetako sarrerak erlazionatzen dituena) sortzea. *Erdaratu* proiektuko baliabide linguistikoak berrerabiliz gero, euskara batuko hiztegi elebakarra dagoeneko sortuta egongo litzateke. Sortu beharreko hiztegiak beraz, bizkaierako hiztegi elebakarra eta bien arteko lotura egiten duen hiztegi elebiduna izango lirateke.

Hiztegi hauek XML formatuko fitxategiak dira, *Apertium*en DIX luzapenarekin ezagutzen direnak. Hiztegi hauen egitura azaldu asmoz, fitxategi hauetako bakoitzean aurki ditzakegun atalak zerrendatuko dira:

- Erabiltzen den alfabetoaren zehaztapena

- Analisi morfologikorako erabiliko diren etiketen definizioa

- Paradigmen definizioak. Bertan paradigma ezberdinak definitzen dira, banaka, bakoitzaren flexio guztiak zerrendatuz. Flexioa zehazteaz gain, honen analisia ere jartzen da, adibide honetan moduan:

```
<pardef n="I_41">
   <e>
      <p>
         <l>aren</l>
         <r><j/>a<s n="det"/><s n="art"/><s
n="sg"/><j/>en<s n="post"/></r>
      </p>
   </e>
   <e>
      <p>
         <l>ari</l>
         <r><j/>a<s n="det"/><s n="art"/><s n="sg"/><j/>i<s
n="post"/></r>
      </p>
   </e>
...
```

- Sarrerak. Sarrera bakoitzak left (l) eta right (r) aldea izango du, aurrekarietan aipatu den bezala, *lttoolbox* erabiliz fitxategi hauek konpilatzerakoan dagokion norabideko transduktoreak sortzeko, eta sortutako norabidearen arabera erabiltzeko: analisirako edo sorkuntzarako.

```
<e>
   <p>
      <l>apaiz</l>
      <r>apaiz<s n="n"/></r>
   </p>
   <par n="I_41"/>
</e>
```

Hiztegi elebidunak, hiztegi elebakarren atal berdinak ditu baina sarrerak adierazteko modua, bestelakoa da. Iturburu-hizkuntzako lema bakoitza xede-hizkuntzako dagokion lemarekin elkartuta dago:

```
<e>
  <p>
    <l>apaiz</l>
    <r>apaiz<s n="n"/></r>
  </p>
  <par n="I_41"/>
</e>
```

Adibide hau zuzena izateko, bizkaierako hiztegi elebakarrean *abade* sarrera egongo litzateke.

Hiru DIX fitxategi hauek batera editatu ohi dira, sarrerak banaka landuz eta bertako datuak sortu edo moldatzeko *BiDBL* eta *EDBL*ko datu linguistikoetan[9] oinarrituz. Esan bezala, euskarazko hiztegia *Erdaratu* proiektutik hartuz, hiztegi elebiduna eta bizkaierako hiztegi elebakarra sortzea izango litzateke egin beharreko lana; lan astuna eta luzea, beraz. *Apertium*en euskara batua eta bizkaiera hizkuntza-bikote berria modu honetan sortzeko abantaila nagusia, hasieratik librea den euskarazko baliabide batean oinarritzea da. Hau da, sortuko diren sarrera berriak bizkaierakoak izango dira eta euskara batukoekin lotuta joango dira; beraz, dagoeneko libre dagoen informazioa osatuko da.

4.2 Bigarren aukera: Morfologikoki konplexuak diren datuentzat egokiagoak diren teknologiak

Bigarren aukera, *BiDBL* eta *EDBL* datu-baseetako informazioa egoera finituko transduktoreen bidez tratatzeko daukagun esperientzia kontutan hartuz, eta bereziki *Foma* erreminta erabiliz, *Apertium*en garapen prozesuan dauden inplementazio berriak erabiliz datu linguistikoak automatikoki tratatzeko modua aztertzea da.

Aukera hau garatzeko, lehenengo pausua, bi datu-baseetatik informazioa egoki erauztea da (esportatzea), horrela, bi datu-baseen bildura bat eginez, hitz bakoitzaren itzulpena eta analisi morfologikoa batera lortu ahal izango dira. Bi datu-baseek egitura bera daukate, beraz, bietan aurki dezakegu sarrera bat estandarra den ala ez adierazten duen eremua. Eremu honi esker, bi datu-baseen arteko lotura egin ahal da: sarrera berdinerako, batean estandar dena eta bestean ez lotuaz. Horrela, lexikoia sortzerako garaian, analisia xede-hizkuntzan ematea lortuko da; baita lexikoien eta jarraitze-klaseen informazioa ere.

9 XuxenB garatzerakoen erabili ziren irizpide linguistiko berberak erabiltzeko helburuarekin.

Adibide moduan, *berriro* sarrera hartuko dugu. *EDBL* datu-basean, beste hainbat informazioz gain, honen antzeko datuak topa ditzakegu:

Kode	Sarrera	Homografo-zenb	Hiztegi-sarrera	Estandarra	...
001	berriro	0	+	+	
002	barriro	0	+	-	

4.1. taula: EDBL-ren egituraren lagina

*BiDBL*n berriz, hauek:

Kode	Sarrera	Homografo-zenb	Hiztegi-sarrera	Estandarra	...
001	berriro	0	+	-	
002	barriro	0	+	+	

4.2. taula: BiDBLren egituraren lagina

Adibideari erreparatuz, ikus daiteke, sarrera bera datu-base batean estandarra izanik bestean ez dela, eta alderantziz. Hori izango da hain justu bi datu-baseen arteko lotura egingo duena. Behin lotura hori finkatuta, datu-baseko beste tauletan kontsulta eginez sarrera bakoitzaren morfotaktika lortzen da; datu-basean, sarrera bakoitzak zein jarraitze-klase daukan eta zein lexikoiri dagokion gordeta baitago.

Lortutako esportazioa *Foma*rekin konpilatuz, iturburu-hizkuntzako azaleko forma bat xede-hizkuntzako analisi batean bihurtzen duen transduktorea lortuko da. Euskara batuko forma bat analizatzean, bizkaierako analisi bat lortuko da.

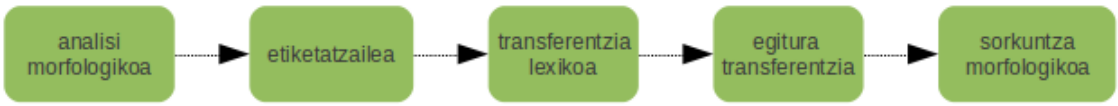
Datu hauek automatikoki lantzeak eragozpenak izan ditzakeela ere aurreikusi da. Orain arte, ez da inoiz bi datu-baseen arteko lotura egin bertako informazio morfologikoaz baliatzeko, ez eta itzulpen automatikoko arloan ezaugarri hauetako lanik egiteko ere. Honek espero gabeko emaitzak lortzea ekar lezake.

Bestalde, datu-baseetako informazio osoarekin egingo da lan, automatikoki lortuko baitira datuak. Honek beste eragozpen bat dakar: transduktoreetako informazioa libre jartzeko momentuan hiztegien murrizketa egin beharko da, *Matxin*en libre dauden sarrerak zein diren kontutan hartuz.

4.3 Erabakia eta diseinu orokorra

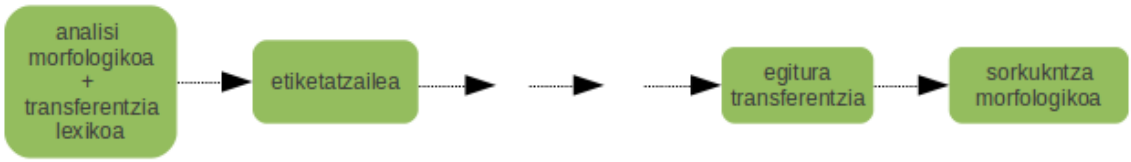
Proiektu honen garapenerako zeuden aukeren artetik bigarrena aukeratu da, hau da, *EDBL* eta *BiDBL* datu-baseetan dagoen informazioa automatikoki tratatuz transduktoreak sortzea. Horrela, *Apertium*en garapen prozesuan dauden beste softwareen garapenean parte hartu ahal izango dugu. Gainera, azken urteotan datu-base hauetako informazioa FST teknologiaz maneiatzen izan dugun esperientziarekin eta proiektu honetarako aurkezten den erronkarekin, hizkuntzalaritza konputazionalean antzeko datu-iturriak eta teknologiak erabili behar direnerako ikuspuntu zabalago bat izateko aukera ematen du.

*Apertium*en itzulpen-fluxuan, irudian agertzen den moduan, bost modulu nagusi definitzen dira. Bertan, bi transduktore erabiltzen dira norabide bakoitzerako: bata analisi morfologikoa gauzatzeko eta bestea sorkuntza morfologikorako.



4.1. irudia: Apertiumen ohiko itzulpen-fluxua

Proiektuaren garapenerako hartutako erabakiak, ordea, itzulpena egiteko erabiltzen den ohiko eskema aldatuko du. Transferentzia lexikoa analisi morfologikoa eta etiketatzearen ondoren egin beharrean, analisi morfologikoarekin batera egingo baita. Beraz, bost modulu exekutatu beharrean lau exekutatuko dira, modu honetan:



4.2. irudia: Apertiumen erabiliko den itzulpen-fluxua

Laburbilduz, hizkuntza-bikote berri bat modu arruntean sortzetik proiektu hau garatzeko hartutako garapen modura dauden ezberdintasunak hauek dira:

1.- *Apertium*en ohiko hiztegirik ez da erabiliko. Beraz, XML formatuko hiztegi elebakarrak eta elebiduna ez dira existituko.

2.- Analisi eta sorkuntza morfologikoa egiteko, *lttoolbox* erabili beharrean morfologikoki konplexuak diren hizkuntzetarako egokiagoak diren *Foma* eta *HFST* erremintak erabiliko dira. Beraz, *Apertium*eko katea exekutatzean *lt-proc* exekutagarria erabili ordez, *foma-proc* eta *hfst-proc* erabiliko dira.

Lan-ingurunearen instalazioan, *Apertium*en instalazio arrunta egiteaz gain, *Foma* eta *HFST* erremintak eskuratu eta instalatu dira. *Apertium*en repositorioko *branch*ean aurki daiteke *Foma*ko iturburua eta *Apertium*ekin elkarreragiteko behar duen *foma-proc*. *HFST* erremintaren kasuan, aldiz, proiektu horretako bertako repositoriotik eskura daitezke iturburu kodea eta *Apertium*ekin elkarreragiteko beharrezko *hfst-proc* exekutagarria.

5 Garapena

Datu linguistikoak, lehen azaldu bezala, aurrez sortuta dauden *EDBL* eta *BiZDBL*ko informazioa modu egokian bilduz sortuko da. Analisirako erabili diren transduktoreak sortzeko bi datu-baseen arteko alegiazko lotura bat sortu da sarrera bakoitza estandarra den ala ez esaten duen eremuaren bitartez, informazioa modu egokian erauziz; aldiz, sorkuntzarako erabili diren transduktoreetan, aurrerago azalduko den moduan, kasuan kasuko datu-basetik hartu da informazioa.

Analisi morfologikoa egiteko sortuko diren transduktoreak, beraz, bi datu-baseetako informazioa erabiliz sortuko dira. Iturburu-hizkuntza euskara batua izanik, hau analizatzeko transduktorea sortzeko *EDBL*n estandarrak eta *BiEDBL*n ez-estandarrak diren sarrera guztien artean lotura sortu da, horrela gramatikaren fitxategia sortzeko datu-baseetatik esportazioa egitean datu hauek hartu dira kontutan: bi-mailatako forma *BizEDBL*tik, azaleko forma *EDBL*tik eta jarraitze-klasea *BizEDBL*tik.

Adibide gisa euskara batuko *eman* forma hartuz, horrela sortu da beharrezko informazioa gramatikaren fitxategian txertatzeko: berdez dagoena, *BiDBL*ko informazioa da. Urdinez dagoena, aldiz, *EDBL*koa.

```
LEXICON et_izenak
      emon[ald_eman][[Sarrera_emon--3][KAT_IZE][AZP_ARR][BIZ_-]
[ERROR-KODE_DIAL]]:eman I;
```

*Apertium*en barnean erabiltzen diren etiketak, ordea, ez dira normalean egiten ditugun esportazioetan erabiltzen direnen modukoak. Horregatik, aldaketa hauek egin dira gramatiketan:

- Proiektu honetarako soberan dagoen informazioa kendu, hala nola, aldaeren informazioa, eta errore-kode dialektalen informazioa.
- Informazio morfologikoa ematen duten etiketak *Apertium*en normalean erabiltzen diren etiketekin *mapeatu* dira, adibidez, *KAT_IZE* guztien ordez *n* erabiliz.

- Etiketak '[' eta ']' karaktereekin mugatu beharrean '<' eta '>' erabili dira *Apertium*ekin integratu ahal izateko.
- Azkenik, homografo-zenbakia eta behar ez den beste informazioa kendu da.

Beraz, aurreko adibidea, hurrengo honetan bihurtuko litzateke:

```
emon<n>:eman      I;
```

Aurreko paragrafoetan azaldutako prozedura erabiliz, lau transduktore sortu dira itzultzaileak bi norabideetako itzulpenak egiteko:

Euskara batua -> bizkaiera norabiderako,

1.- Euskarazko formak bizkaieraz analizatzen dituen ***analizatzailea***

2.- Bizkaierako formen analisia egiten duen ***analizatzailea***

eta bizkaiera -> euskara batua norabiderako,

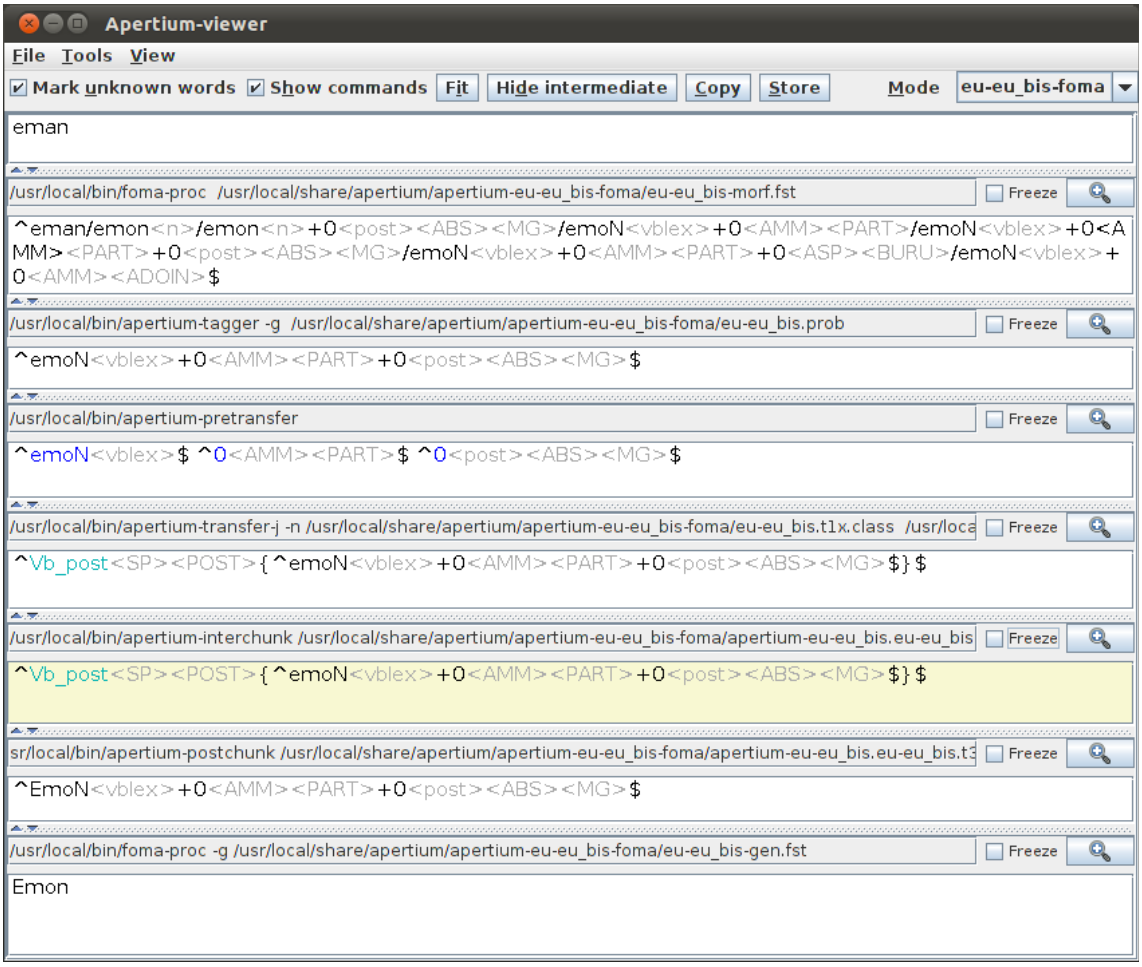
1.- Bizkaierazko formak euskara batuan analizatzen dituen ***analizatzailea***

2.- Euskara batuko formen analisia egiten duen ***analizatzailea***

Transduktoreen izaera dela eta, analizatzaile hauetako bakoitzak sarrera bakoitzeko gutxienez irteera bat emango du eta besterik ezean analizatzaileak izango dira. Transduktoreek bi norabideetan funtzionatzeko gaitasuna dute: *up* eta *down* egin dezakete; esperimentu honetarako sortutako transduktoreek *up* norabidean analizatzen dutenez, analizatzaile deitu zaie guztiei (ikus 3.3. irudia: Transduktorearen funtzionamendu orokorra).

Sortu diren lau transduktoreak eta hauek dituzten ezaugarriak kontutan hartuz, hau izango litzateke euskara batua -> bizkaiera norabidean itzulpenak egiteko sortuko

litzatekeen datu-fluxu osoa apertium-viewer[10] aplikazioarekin pausuz pausuko emaitzak ikusiz:



5.1. irudia: apertium-viewer erabiliz, euskara batutik bizkaierara itzulpenak egiteko erabili den fluxu osoa

Irudian, *BiscayTrad*ek itzulpen bat lortzeko exekutatzen dituen pausu guztiak ikus daitezke; 4.2. irudia: Apertiumen erabiliko den itzulpen-fluxuan agertzen diren pausu guztiak agertzen dira. Gainera, *apertium-viewer*i esker, egituraren transferentzia burutzeko behar diren pausu guztiak ere ikus daitezke: *pretransfer*, *transfer*, *interchunk* eta *postchunk*.

10 http://javabog.dk:8080/apertium-viewer/launch.jnlp

Transduktoreak sortzeko erabili diren datuekin jarraituz, sorkuntza morfologikoko moduluan euskara batuko formak sortzeko *EDBL* erabili da; *BiDBL*, aldiz, bizkaierako formak sortzeko. Transferentzia lexikoa lehenengo moduluan egiten denez, nahikoa da xede-hizkuntzako formak sortzeko dagokion datu-baseko informazioa erabiltzea.

Hasierako probak egin ostean, itzulpen batzuetan emaitza gisa analisi morfologiko bat jasotzen zen; sorkuntza morfologikoko modulua exekutatzean, transduktorean kontsulta egin ondoren, analisi horri ez zegokion formarik xede-hizkuntzan. Kasu hauetan, *Apertium*ek, analisia bera itzultzen du itzulpen moduan. Emaitzak aztertu ondoren, akats hau, *EDBL* eta *BiDBL* modu independente batean aberastu izanagatik izan daitekeela ikusi da, ez baitaude datuak batera erabiltzeko pentsatuta.

Hona hemen, itzulpen prozesuan arazoak eman ditzakeen hitz bat: *fedegabe*. *EDBL*n kontsulta eginez gero, *fedegabe* formak bi kategoria posible ditu, aditza eta adjektiboa. *BiDBL*n kontsultatuz gero, *fedebako* hitzak (esperotako itzulpenak), kategoria posible bakarra du, adjektiboa. Hau horrela, bizkaieratik euskara batura egingo diren itzulpenetan ez da arazorik egongo; bai, ordea, euskara batutik bizkaierara egiten diren itzulpenetan. Etiketatzaile morfologikoak aditza kategoria daukan analisia aukeratzen badu, ezin izango du xede-hizkuntzan formarik sortu. Gainera, kontutan hartzekoa da euskara batuko eta bizkaierako formak eta analisiak nahastuta egongo direla. Hau hobeto ulertzeko, hona analisiaren deia eta ematen duen emaitza:

```
echo fedegabe | apertium -d . eu-eu_bis-foma-anmor

^fedegabe/
      fedebako<adj>/
      fedebako<adj>+0<post><ABS><MG>/
      fedegabe<vblex>+0<AMM><ADOIN>$
```

Sortu den azken analisia euskara batuan dago, ez dauka transferentzia lexikorik, datu-baseen esportazio bateratua egitean ez baita sarrera horren baliokiderik topatu bizkaierako datu-basean. Kasu honen antzekoetarako eta sorkuntza morfologikorako

erabiltzen den transduktoreak beti emaitzaren bat izan dezan, *Foma* erremintak eskaintzen duen *priority union* funtzioa erabili da.

5.1.1 Lehentasunaren araberako elkarketa (priority union)

Aipatu bezala, sorkuntza morfologikorako sortutako transduktoreek beti emaitza eman dezaten transduktoreen arteko konbinaketa jakin bat egin da. Horretarako, Fomak eskaitzen duen *priority union* eragiketa erabili da, sorkuntzarako prestatutako transduktorea analisikoarekin konbinatuz beti emaitzaren bat emango duen transduktore bat izateko.

Priority union eragigaiak *lower* edo *upper* moduan lan egin dezake, eta gure kasuan, *lower* moduan erabili da: .P. erabiliz egin da transduktoreen konbinaketa.

Demagun horrelako kasu hipotetiko bat:

- T1 transduktorean horrelako informazioa dagoela:
    - F1 : A1
    - F3 : A3
- T2 transduktorean aldiz, beste informazio hau dago:
    - F2 : A2
    - F4 : A3

Transduktoreen konbinaketarekin lortu nahi den transduktore berriak, A1, A2 eta A3 analisien sorkuntza egin ahal izango du. Konbinaketak egiteko *Foma*k *union* eta *priority union* (*upper* eta *lower*) eragigaiak ditu. Lehenengo eragigaiak, *union*ek (| eragigaia), bi transduktoreen emaitzen arteko konbinaketa arrunt bat egiten du, hau da, kasu guztietarako bien emaitzak biltzen ditu. *Priority union* konbinaketarako, aldiz, transduktore baten ala bestearen emaitza itzultzen da, ezarritako lehentasunaren arabera; transduktoreen konbinaketa egitean ordena kontutan hartzen da. T1 .P. T2

konbinaketaren emaitzak, A3 analisia sortzean F4 forma emango luke emaitza gisa; T2 .P. T1 konbinaketa eginez, aldiz, F3 forma emango luke emaitza gisa.

Modu hau erabiliz sortu dira esperimentu honetarako sortutako sorkuntza morfologikorako transduktoreak.

5.1.2 Transduktore haztatuak

*Eleka*n gehien ezagutzen eta erabiltzen dugun FST teknologian oinarritutako softwarea, lehen aipatu bezala, *Foma* da. *Apertium* komunitatean, ordea, *Foma*rekin lan egiteaz gain *HFST*rekin ere lan egiten da, eta bi software hauek elkarrekin erabiliz transduktore haztatuak sortzeko aukera aztertu da.

Transduktoreak *Foma*rekin daukagun esperientzia dela eta datu-baseetatik informazioa erauzi eta formatu *Foma* formatuko transduktoreak sortu dira. Hauetatik abiatuz, *HFST*k bere formatura pasatzeko aukera ematen du. Adibidez, *Foma*tik *HFST*rako formatu aldaketa arrunt bat modu honetan egingo litzateke:

```
gzip -d -c <sarrera> | hfst-fst2fst -O -o <irteera>
```

Modu honetan, *HFST* formatuko transduktore bat izango dugu.

Sorkuntza morfologikorako prestatutako transduktoreak erabiltzean, gainsorkuntzaren arazoa aurkitu da. Analisi bakarrerako, azaleko forma bat baino gehiago lortzen dira eta honek azken emaitzan ondorioak dauzka. *HFST*k formak eta analisiak ponderatzeko ematen duen aukera aztertuko da beraz, datu originalak moldatu gabe eta hizkuntzalari baten laguntzaz, gainsorkuntza hori ekiditeko. Helburua, ponderazio hori oinarrizko transduktoreetako kodetik kanpo egitea da, hizkuntzalari batek erraz ulertuko duen fitxategi batean.

*HFST*k, beraz, hurrengo adibidean agertzen den moduko fitxategi batetik abiatuz transduktore bat sortzea ahalbidetzen du, adibidez, beste batekin konbinatzeko. Fitxategi hau, corpus baten arabera sor liteke, bertako testu osoa lematizatuz eta lema bakoitzaren

maiztasuna kalkulatuz. Horrela, hizkuntzalariaren lana fitxategiaren edizioa besterik ez litzateke izango.

- priorities.txt fitxategia

```
ostiral 1
bariku  5
```

Testu-hutsa duen fitxategia transduktore bihurtuko da, komando hau exekutatuz:

```
hfst-strings2fst -j priorities.txt -o priorities.txt.hfst
```

- Orain demagun gramatikan lexikoi hau daukagula:

```
Multichar_Symbols
%<n%>

LEXICON Root
bariku%<n%>:ostiral J ;
bariku%<n%>:bariku J ;

LEXICON J
# ;
```

Lexikoi honetatik *Foma*ko transduktore bat sortuko da (analizatu.fst), *hfst-fst2fst* erabiliz *OpenFST*ra bihurtzeko.

```
$ gzip -d -c analizatu.fst | hfst-fst2fst -t -o analizatu.hfst
```

Sortutako azken transduktore hau, lehen sortutako *priorities.txt.hfst* transduktorearekin konbinatuko da, bilatzen ari ginena sortu arte.

```
$ hfst-combine -o analizatu_haztatua.hfst analizatu.hfst
priorities.txt.hfst
$ hfst-lookup analizatu_haztatua.hfst
bariku<n>
ostiral     1.000000
bariku      5.000000
```

HFST erabiliz egindako esperimentu txiki honetan, sorkuntza morfologikoa egiten denean sarritan geratu ohi den gainsorkuntza ekiditeko modu egoki bat aurkitu

da. Hala ere, *Apertium*ekin batera martxan jartzeko, *hfst-proc* exekutagarriak mota honetako emaitzak interpretatu beharko lituzke, eta oraingoz ez da horrela. Beraz, etorkizuneko lanetarako utziko da esperimentu hau benetako sistema batean aplikatzea.

5.2 Etiketatzaile morfologikoa

*Apertium*en kateko bigarren modulua, aipatu bezala, etiketatzaile morfologikoa da. Analisi morfologikoan lortutako emaitza guztietatik bakarra aukeratzea da modulu honen helburua, eta horretarako gainbegiratu gabeko ikasketa automatiko metodo bat proposatzen da. Kasu honetan *HMM*n oinarritzen da (*Hidden Markov Model*) eta emaitza gisa probabilitate fitxategi bat sortzen du. Fitxategi hauen sorkuntzarako beharrezkoak izango dira iturburu-hizkuntzako corpus handiak eta hauek analizatzeko transduktoreak. Euskarazko probabilitate fitxategia sortzeko Berriako corpus bat erabiliko da (16,5 milioi hitz inguru); Labayru Institutuko bat bizkaierarako (500.000 hitz inguru).

Pausu hauek jarraituko dira probabilitate fitxategi horiek sortzeko:

- Iturburu-hizkuntzako corpusa analizatzailetik pasako da eta egon daitezkeen anbiguotasun-klase guztiak definituko dira dagokion *tsx* fitxategian. Datu hauek *apertium-filter-ambiguity* komandoarekin batera erabiliz hiztegi bat sortuko da, adibidez, *eu.dic*.

- Ondoren, ikasketa automatiko honetarako pentsatuta dagoen *eu-eu_bis-unsupervised.make makefile*-a martxan jarriko da, dagokion probabilitate fitxategia sortuz, adibidez, *eu-eu_bis.prob.*

Hurrengo adibidean, etiketatzaile morfologikoak emaitzan izan dezakeen eragina ikus dezakegu. Euskara batuko *dio* forma hartuz, honen itzulpen posibleak bi izan daitezkeela badakigu: dino (esan aditzaren forma trinko bat) edo deutso (nor-nori-nork motako aditz laguntzailea). Hauek dira, beraz, *dio* formaren analisi morfologiko posibleak:

```
edun<vbsint><pri><NR_HU><NI_HU><NK_HU>
```

```
esan<vbsint><pri><NR_HU><NK_HU><PNT>
ukan<vbsint><pri><NR_HU><NI_HU><NK_HU><PNT>
```

Analisi morfologikoaren ondoren datorren pausua da etiketatze morfologikoa. Bertan, hiru analisi posible horietako bakarra aukeratuko da. Aukera horren arabera, *dio* formaren itzulpena *dino* edo *deutso* izango da. Analisi egokia aukeratzeko, esan bezala, estatistikan oinarritutako fitxategiak sortuko dira (hauek sortzeko emandako pausu guztiak gehigarrietan daude zehatz-mehatz azalduta).

5.3 Transferentzia sintaktikoa

*Apertium*en sorreran, gertuko hizkuntzen arteko itzulpenak egiteko helburuarekin, azaleko transferentzia sintaktikoa diseinatu zen. *Apertium*en ondorengo bertsioetan, aldiz, transferentzia sintaktiko sakonagoa erabiltzen du, antzekoak ez diren hizkuntzen arteko itzulpenak egin ahal izateko, hizkuntza hauen ezaugarri linguistikoak maneiatzeko aukera ematen duena (Armentano-Oller, C. et al.).

BiscayTrad sistemaren garapenerako ere, transferentziarako erregela sintaktikoak sortu dira. Erregela orokorrak berdinak izan dira euskara baturako eta bizkaierarako, hitzek jarraitzen duten egitura berdina baita. Bestalde, fenomeno jakin batzuk hizkuntzaren arabera definitu behar izan dira; hona hemen adibide bat:

```
<rule comment="REGLA: verbos + tu + Ez + gero (eskatuz gero -- eskatu
ezkero)">
  <pattern>
      <pattern-item n="verbos"/>
      <pattern-item n="amm_atz"/>
      <pattern-item n="post_gra"/>
      <pattern-item n="gero"/>
  </pattern>
  <action>
      <call-macro n="firstWord">
        <with-param pos="1"/>
      </call-macro>
      <out>
        <chunk name="vb_post" case="caseFirstWord">
            <tags>
              <tag><lit-tag v="SUBADV"/></tag>
              <tag><clip pos="1" side="tl" part="temps"/></tag>
```

```
            </tags>
            <mlu>
              <lu>
                  <clip pos="1" side="tl" part="whole"/>
              </lu>
              <lu>
                  <clip pos="2" side="tl" part="whole"/>
              </lu>

            </mlu>
        </chunk>
        <b pos="2"/>
        <chunk name="adv" case="caseFirstWord">
            <tags>
              <tag><lit-tag v="ADV"/></tag>
            </tags>

            <lu>
              <lit v="ezkero"/>
              <lit-tag v="adv"/>
            </lu>
        </chunk>
      </out>
  </action>
</rule>
```

Erregela bat sortzeko beharrezkoa da, lehendabizi, patroi bat definitzea. Adibide honetan, ikus dezakegu, lau elementuko patroi bat definitzen dela: *verbos*, *amm_atz*, *post_gra* eta *gero*. Elementu hauetako bakoitza ere definituta egongo da, erregela fitxategiaren hasieran; adibidez, *verbos* elementuak aditz guztiak barne hartzen ditu.

Behin patroiak definituta, sistemak horrelako patroi bat topatzen duen bakoitzean egin beharrekoa definituko da *action* etiketen artean. Adibide honen kasuan, *firstWord* izeneko *macro*ari deituko dio lehendabizi, hitzaren lehenengo hizkiak letra larria ala ez izan behar duen erabakitzen duen *macro*a. Ondoren, irteera gisa jaso nahi diren bi *chunk*ak definituko dira. Adibide honetan, adibidez, lehenengo chunkak sarrera moduan datozen lehen bi elementuak jasoko ditu (pos=1 eta pos=2) bakoitzari dagokien informazio morfologiko guztiarekin (part="whole").

Bigarren *chunk*a, aldiz, eskuz sortuko da; honi esker gero->ezkero itzulpena egin ahal izango da.

Erregela honek itzulpenean daukan eragina ikuste aldera, hona hemen erregela horrekin eta gabe izan ditzakegun emaitza ezberdinak.

Erregela hori kodetu gabe:

- Sarrera : eskatuz gero
- Transferentzia baino lehen:

```
^eska<vblex>$ ^tu<AMM><PART>$ ^Ez<post><INS><MG>$ ^gero<adv>$
```

- Transferentziaren ondoren:

```
^vb_post<SP><POST>{^eska<vblex>+tu<AMM><PART>+Ez<post><INS><MG>$}
$ ^gero<adv>{^gero<adv>$}$
```

Erregela horrekin:

- Sarrera : eskatuz gero
- Transferentzia baino lehen:

```
^eska<vblex>$ ^tu<AMM><PART>$ ^Ez<post><INS><MG>$ ^gero<adv>$
```

- Transferentziaren ondoren:

```
^vb_post<SUBADV>{^eska<vblex>+tu<AMM><PART>$}$
^adv<ADV>{^ezkero<adv>$}$
```

Erregela aplikatzean, beraz, deklinabide-kasu instrumentala desagertu egiten da; baita *gero* adberbioa ere. Azken honen ordez, *ezkero* adberbioa txertatzen da.

Euskarazko hitzek duten morfologia konplexua dela eta, erregela orokor asko idatzi behar izan dira, hitzek dituzten morfema konbinaketa guztiak barne hartzeko. Beraz, erregela horiek itzulpen bakoitzean exekutatzea lan nekeza izan da sistemarentzat. Hau konpondu asmoz, *bytecode for transfer* izeneko proiektua erabili da.

5.3.1 Bytecode for transfer

*Apertium*ek transferentzia sintaktikoak behar duen baliabideen kontsumoa jaisteko asmoarekin sortutako proiektu bat da hau. Orokorrean, *Apertium* oinarritzat hartuz sortutako itzulpen automatikoko sistemetan, transferentzia sintaktikoa lantzen duen

modulua da kate osoan baliabide gehien kontsumitzen duen zatia; beraz, hementxe galtzen da denbora gehien itzulpen-eskaera bat egiten denean.

Hainbeste baliabide behar izatearen arrazoia, transferentzia sintaktikoa egitean erabiltzen diren XML erraldoien interpretazioa da; exekuzioa gauzatzen den bakoitzean XML horietan definituta dauden erregela guztiak interpretatzen dira, ez baitaude makina lengoaian kodetuta. Abiadura arazo hau konpontzeko, *bytecode for transfer*[11] proiektuaren laguntzaz, XML horiek hartu eta *Java*ko *bytecode* kompilatzailea erabiliz *Java*ko iturburu-kode bihurtzen dira. Transferentzia sintaktikoa egiten den bitartean, *Java Virtual Machine*k erregela erabilienak makina-lengoaian bihurtzen ditu, abiadura nabarmen azkartuz.

Apertiumeko transferentzia sintaktikoa burutzeko kate zatia horrelakoa izango litzateke:

```
apertium-transfer apertium-eu-eu_bis.eu-eu_bis.t1x eu-
eu_bis.t1x.bin
```

Bytecode for transfer erabiliz, ordea, horrelakoa:

```
apertium-transfer-j -n eu-eu_bis.t1x.class eu-eu_bis.t1x.bin
```

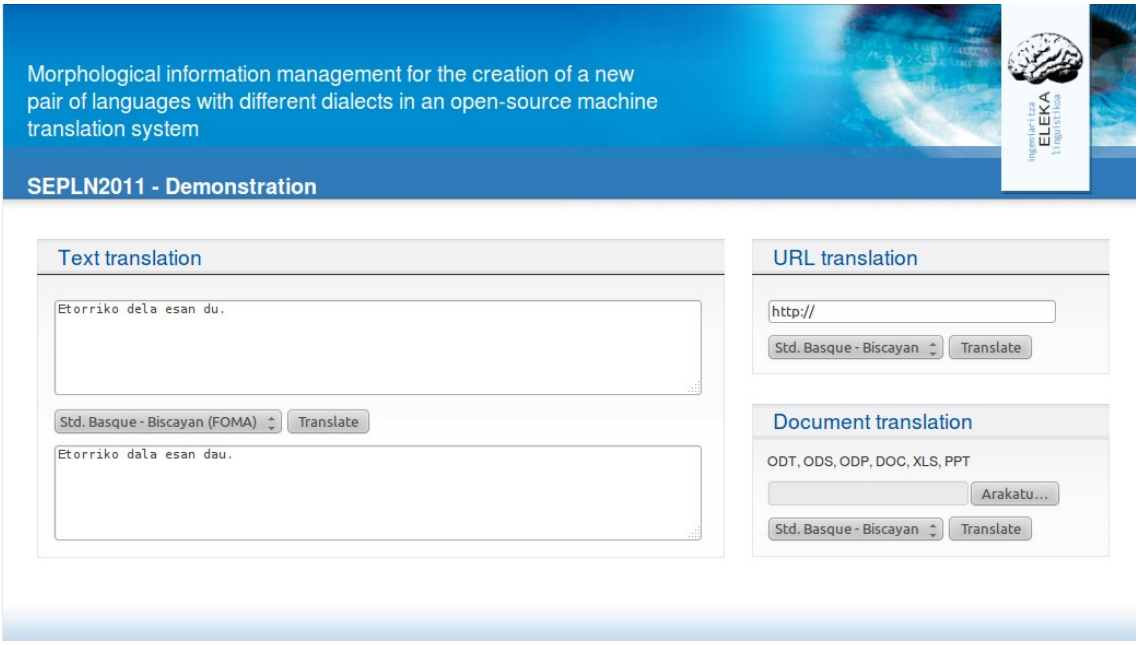
Abiadurari dagokionez, euskara batua eta bizkaieraren artean egindako itzulpenetan *bytecode for transfer* erabiliz lortu den abiadura hasierako abiadura baino bost aldiz txikiagoa izan da.

11 http://wiki.apertium.org/wiki/Bytecode_for_transfer

6 Aplikazioa

Egindako lana erakusteko modu egokiena web bidezko interfaze bat garatzea da, eta *Apertium*ek eskaintzen dituen aukerak aprobetxatuz, itzulpen-zerbitzu bat izan litekeen webgune bat sortzea.

Sarreran aipatu moduan, master-tesi honetan egindako ikerketatik lortu den prototipoa, irailaren 6an *SEPLN2011* kongresuan egongo den *demo*en saioan aurkeztuko da. Demo horrek, horrelako itxura izango du eta http://sepln2011.eleka.net webgunean egongo da atzigarri:



6.1. irudia: Demo-aren pantaila-argazkia

Irudian ikus daitekeen bezala, interfazea hiru zatitan banatuta egongo da: testu hutsa itzultzeko formularioa, URLak itzultzeko formularioa eta dokumentuak itzultzeko formularioa.

6.1.1 Testu-hutsa itzuli

Testu hutsa itzultzeko interfazeak, testu-kutxa batetik testua hartuko du eta *conboboxean* itzulpenaren norabidea eta transduktoreen teknologia aukeratu ondoren itzultzeko botoiari eman ahalko zaio. Emaitza ere testu-kutxa batean emango da. Emaitza hau lortzeko erabiltzen fluxua horrelakoa izan da:

1.- Testu-kutxako testua txt fitxategi batean sartu (sarrera.txt)

2.- apertium eu-eu_bis-foma < sarrera.txt > irteera.txt

3.- irteera.txt fitxategian dagoen testua web-interfazean kargatu

Hona hemen, testu-hutsa itzultzeko erabiltzen den interfazearen zatia:

6.2. irudia: Testu-hutsaren itzulpena

6.1.2 Dokumentuen itzulpena

*Apertium*ek integratuta dauzka ODF (*Open Document Format*)formatuko dokumentuak itzultzeko behar diren testua erauzteko eta formateatzeko moduluak. Beraz, ODF formatuko dokumentu bat itzultzeko fluxua horrelakoa izango da:

1.- ODF fitxategia interfazetik zerbitzarira igo (sarrera.odt)

2.- apertium -f odt eu-eu_bis-foma < sarrera.odt > irteera.odt

3.- irteera.odt dokumentua emaitza gisa itzuli

MS Office formatuko dokumentuak, ordea, ODF formatukoak baino arruntagoak dira erabiltzaile gehienentzat. *Apertium*ek MS Office formatuko dokumentuen itzulpenak egiteko modulurik ez duenez, dokumentuen bihurketa bat egitea erabaki da. MS Office formatuko dokumentua behin zerbitzarira igota, ODF formatuko dokumentu batean bihurtuko da. Honen itzulpena egin ondoren, alderantzizko bihurketa egingo da, erabiltzaileari igo duen formatuko fitxategi bat emaitza gisa itzuliz.

6.1.3 URLen itzulpena

*Apertium*ek URLen itzulpena egiteko modulua garatuta dauka, beraz, interfazetik sartzen den URL helbide hori hartu eta bertako informazioa *Apertium*i pasa behar zaio. Horrelako exekuzio-fluxua erabiltzen da URLak itzultzeko:

1.- Interfazetik URLa eta itzulpenaren norabidea jaso

2.- URL hori pythoneko urllib2 paketearekin tratatuz, html iturburu-kodea fitxategi batera exportatu

3.- apertium -f html eu-eu_bis-foma < sarrera.txt > irteera.txt

4.- irteera.txt fitxategian dagoen html iturburu-kodea itzuli

Hurrengo irudian, Berria egunkariaren zati bat, automatikoki bizkaierara itzulita.

6.3. irudia: Berria egunkaria BiscayTrad sistemarekin automatikoki bizkaierara itzulita

6.2 Zerbitzariaren arkitektura

Apertium komando lerrotik exekutagarriak diren moduluez osatutako erreminta bat da, ez da liburutegi bat, beraz, hau zerbitzari gisara modu eraginkor batean jartzeko hainbat proba egin dira.

Era berean, webgunea martxan jartzeko, *Django* web garapenerako *framework*a erabili da. Hau *Python* programazio lengoaian garatuta dago, eta garapen berriak ere lengoaia hau erabiliz egiten dira. Beraz, *Apertium*ek eskaintzen dituen baliabideen eta *Django*ren arteko elkarrekintza beharrezkoa da *demo*ko webgune hau martxan jarri ahal izateko.

*Apertium*eko wikian[12] dagoen informazioan oinarrituz, sistemaren eraginkortasuna hobetzeko apertium-service erabiltzea pentsatu da (Minervini, P., 2009). Hau erabiliz *Apertium* zerbitzu moduan instalatu, eta itzulpen-eskaerak modu eraginkorrean erantzuten duen sistema bat lortuko da.

Atal honetan, zerbitzariaren arkitekturari dagozkien xehetasunak emateaz gain, erantzun-denbora egokiago bat lortzeko egindako probak ere azalduko dira.

6.2.1 Apertium-service

Pakete hau *(*Minervini, P., 2009) *Apertium*eko gordailutik deskarga daiteke. *Apertium*, zerbitzuetara zuzendutako arkitektura (*SOA, Service Oriented Architecture*) batean bihurtzea du helburu; horrela, zerbitzu moduan jartzeko aukera bat eskaintzen du. Arkitektura hau egokia da, adibidez, itzulpen automatikoko zerbitzu egoki bat emateko. Hau kontutan hartuta, pakete hau deskargatu eta instalatu da; probak egin eta proiektu honen amaierako demo egoki bat prestatzeko.

Pakete hau ordea, zaharkitua dago; azkenaldiak ez du eguneraketarik izan. Pakete honen sortzailearekin kontaktuan jarri ondoren, eskuartean dauzkagun *Foma* eta *HFST* erremintekin bateragarria ez zela jakin da. Hurrengo pausua beraz, *ScaleMT*rekin

12 http://wiki.apertium.org

proba egitea izango litzateke; alde batetik, gordailuan kode eguneratuago bat daukalako, eta bestetik, *apertium-service* paketearen egileak ere hauxe gomendatu zuelako.

*ScaleMT*ren instalaziorik, ez da egin ordea. Honen ordez, sistemaren demo soil bat egiteko *Python*ek berak eskaintzen dituen aukerak aprobetxatzea pentsatu da.

6.2.2 Pythoneko threading modulua

Aurretik aipatutako *apertiun-service*rekin probak egiten aritu ondoren, eta *demo*aren garapena *Python* lengoaiarekin egin dela kontutan hartuz, irtenbiderik egokiena *Python*ek eskaintzen duen *threading* modulua erabiltzea da; horrela, *demo* hau beste makinaren batera pasatzeak ere ez du eragozpen nabaririk izango.

Threading modulu honek, 'semaforo' bat erabiltzeko aukera ematen du programaren exekuzioan erabiltzen diren baliabideen kudeaketa egoki bat egiteko. HTTP protokolotik, hau da, interfazetik eskaera bat egiten den bakoitzean, *Python*eko hari edo *thread* bat jarriko da martxan, non 'sekzio kritiko' bat izango duen: itzulpena gauzatzen den funtzioari dei egitean. Aipatu bezala, itzulpen-eskaerak hiru motakoak izan daitezke: testu hutsa, URLen itzulpena eta dokumentuen itzulpena. Funtsean, ordea, itzulpen mota hauetarako guztietarako *Apertium*i egiten zaio dei. *Threading* modulu honi esker, modu erraz batean kudeatuko dira baliabideak, nahiz eta baliabideen kudeaketa hau *Apertim*en moduluak hobeto aztertuz, agian, emaitza hobeak lortzeko moduan egin ahalko den.

6.3 Datuen murrizketa

Itzulpen automatikoko sistema libre bat sortzea izan da proiektu honetan hasieratik jarritako helburua. Horregatik, maila teknologikoan erabili diren baliabideak kode irekikoak izan dira. Datu linguistikoei dagokienez ere, hauek libre jartzeko prozedura bat jarraitu da, *Matxin* proiektuan bertan dagoen espainiera-euskara hiztegi librea oinarritzat hartuz, euskara batuko eta bizkaierako lexikoa libre jartzea. Horretarako,

oinarritzat hartutako gramatiketan moldaketak egin behar izan dira, *Matxin*eko hiztegiarekin konparatuz, bertako sarreren informazioa bakarrik utziaz. Gramatika hauek konpilatuz, sistema martxan jartzeko beharrezkoak diren *Foma* eta *HFST* formatuko transduktore guztiak sortu ahal izango dira.

Datuen estaldura kalkulatu da, *EDBL* eta *BiDBL*ko datu guztiak erabiliz eta datu murriztuak erabiliz lortzen den emaitza kalkulatu ahal izateko. Estaldura kalkulatzeko, iturburu-hizkuntzako testua analizatzean lortzen den emaitza aztertu da, hau da, testuko zenbat hitz analizatzen diren. Euskara batuko estaldura kalkulatzeko, lehen aipatutako Berriako corpusa eta Labayruko corpusa erabili dira.

Lortutako emaitzen laburpena, ondorengo taulan azaltzen da:

Iturburu-hizkuntza	Estaldura (datu osoak)	Estaldura (datu murriztuak)
EU	%89.75	%67.04
EU_BIS	%87.86	%65.09

6.1.taula: Baliabide linguistikoen murrizketa. Estalduraren kalkulua.

7 Ebaluazioa

Itzulpen sistema honen ebaluazioa egiteko erabili den metodoa nolakoa izan den azalduko da atal honetan. Ebaluazioa, euskara batua – bizkaiera norabidean egin da, hau baita gehien landu den norabidea.

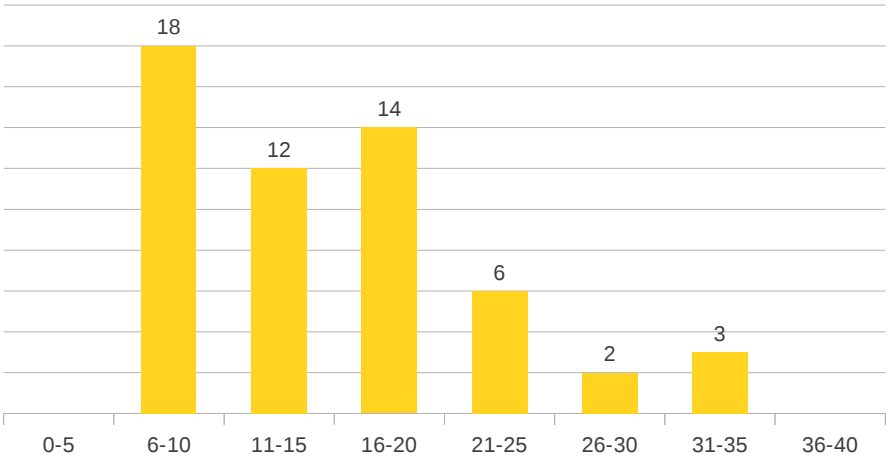
Itzulpen automatikoko sistema baten ebaluazioa egitea ez da lan erraza, testu baten itzulpen automatikoa eta benetako itzulpenaren arteko konparaketa neurtzea oso zaila baita. Lan hau automatikoki egin ahal izateko hainbat neurri ohi badira ere, esperimentu honen ebaluaziorako BLEU neurria erabiliko da (Papineni et al., 2002). Bestalde, asmatze-tasa ere kalkulatuko da, hau da, automatikoki itzulitako esaldietatik zenbatek eman duten esperotako itzulpena.

*Labayru Institutua*k *XuxenB* proiektuaren harira pasatako bizkaiera estandarrean idatzitako corpus batean oinarrituz egin da ebaluazioa, hau da, esperimentu hau aurrera eramateko erabili den bizkaierako corpusean oinarrituz. Bizkaieraz idatzitako testua hartu, itzulpen automatikoko sistema hau prestatzeko erabili diren gidalerroen arabera moldatu eta euskara batura itzuli ditu giza-itzultzaile batek. Horrela, euskara batuko esaldiak iturburutzat hartuz, *BiscayTrad* aplikazioarekin euskara batua – bizkaiera norabidean itzulpen automatikoa gauzatu da. Ondoren, bizkaierako esaldiekin konparatu dira lortutako emaitzak.

55 esaldietatik 20 espero moduan itzuli ditu, beraz proba honetan %36eko asmatze-tasa izan du itzultzaile automatikoko sistema honek. BLEU neurria kalkulatuz, aldiz, 0.7833 balioa lortu da. Kasu honetan, BLEU neurri altua lortu da beste hizkuntza-bikoteetarako ohituta gauden neurriekin alderatuz (ikusi bestela 3.1. taula: Itzulpen automatikoko sistema ezberdinen arteko ebaluazioa. Giza-ebaluazioa eta ebaluazio automatikoa). Emaitza hau, iturburu- eta xede-hizkuntzaren arteko antzekotasunagatik lortu da; euskara batua eta bizkaieraren arteko itzulpena egitean hitz asko iturburu-hizkuntzan bezala idazten dira.

Hartu diren ausazko adibideak, luzera ezberdinetako esaldiak izan dira. Testua narrazio motakoa eta esaldi oso luzekoa izanik, esaldi batzuk puntu-eta-komez bereizi

dira testu txikiagoak sortzeko. Hona hemen erabilitako ebaluazio corpuseko esaldien luzeraren araberako sailkapena:



7.1. irudia: Esaldi luzeerak

Esaldirik motzena 6 hitzekoa izan da; luzeena, aldiz, 34koa. Beraz, orokorrean, ebaluazioa egiteko erabili diren esaldiak luzeak izan direla esan dezakegu.

Automatikoki itzuli eta eskuz errepasatu diren 55 esaldi horietan, okerreko itzulpen automatikoa izan duten esaldiak errepasatu dira, eta aurkitu diren akatsak multzo desberdinetan sailkatu dira:

1. ***Analisien desanbiguazioan okerren bat egon da.***

Euskara batua	**Bizkaiera**
Ikasleen beharrizanak nolakoak ziren, horren arabera atonduko **zituen** lorategiak: ...	Ikasleen beharrizanak nolakoak ziran, horren arabera atonduko **zituen** lorategiak: ...

Iturburu-hizkuntzako esaldia morfologikoki analizatzean, *zituen* aditzak hainbat analisi posible ditu, haien artean adibidea egiteko erabiliko diren bi hauek:

```
1. edun<vbsint><pii><NR_HK><NK_HU>
2. edun<vbsint><pii><NR_HK><NK_HK-K>
```

Aditz hau NOR-NORK motakoa da. Euskara batuan hark-haiek pertsonak ditu, bizkaieraz aldiz, haiek-haiek. Hori dela eta, bi analisi posible horiek sortzen dira, baina etiketatze morfologikoan aukeratzen den analisiaren menpe egongo da emaitza:

1. `edun<vbsint><pii><NR_HK><NK_HU>  --> zituan`
2. `edun<vbsint><pii><NR_HK><NK_HK-K> --> zituen`

2. Datu-basean dagoen informazioa ez da nahikoa izan itzulpen hau egin ahal izateko

Euskara batua	**Bizkaiera**
... **beraien** aurrean erdaldunen bat zegoenean izango zen, ...	... **beraien** aurrean erdeldunen bat egoanean izango zan, ...

Beraien euskara batuko formaren itzulpena *euren* izango litzateke, baina kasu honetan itzulpenik ez da egin. *EDBL* eta *BiDBL*n kontsulta eginez, ikusten da, bi sarrera horiek landu gabeak direla, beraz, ez dira esportazioan sartu.

3. Transferentzia erregelak doitzea falta da.

Euskara batua	**Bizkaiera**
... ekarriko ditugu **ardurarik gabe** ...	... ekarriko ditugu **ardurarik barik** ...

Ebaluaziorako sortutako adibideetan lortutako emaitzak aztertuz, esaldi zati hau topatu da, non, horrelako erregela bat sortu beharko litzatekeen:

```
ardura + partitiboa + gabe  -->  ardura + barik
```

Ebaluazio honek, itzultzailea hobetzeko landu beharreko puntuak zein diren ikusten lagundu digu. Orokorrean, esan daiteke, hiztegi guztiak eskuz sortzea baina lan arinagoa izan dela modu erdi-automatikoan datu-baseetatik informazioa ateratzea. Hala

ere, *Apertium*ek behar dituen baliabide linguistiko batzuk derrigorrez eskuz egin behar dira, adibideetan oinarrituz.

8 Ondorioak eta etorkizuneko lanak

Proiektu honen garapenak hainbat teknologien konbinaketak egiteko eta orain arte eskuartean izan ditugun datuen ustiaketa berri bat egiteko balio izan du. Alde batetik, ezagutzen genituen teknologiak elkarrekin erabiltzeko modua izan dugu, hala nola, *Foma* formatuko transduktoreak eta *Apertium*. Beste alde batetik, ezezagunak genituen teknologien berri izan dugu, *HFST* proiektua kasu, eta egindako esperimentuek itzulpen automatikoarekin lotutako etorkizuneko proiektuetarako ikuspuntu zabalago bat izatea ahalbidetuko du.

Proiektu honetan erabili diren datuak zuzenean euskararako ditugun bi datu-base lexikaletatik erauzteak alde positiboak eta negatiboak izan ditu. Datuak sortzea ia modu automatikoan egin da, nahiz eta erregelen eta datuen konbinaketak egiteak lan dezente sortu duen. Datu-base hauen ezaugarriengatik, ordea, xede-hizkuntzako formak sortzeko orduan arazoak suertatu dira. Arazo hauek bi multzotan sailka daitezke: gainsorkuntza eragiten duten arazoak eta orokorrean helburu-hizkuntzan esperotako emaitzarik ez lortzea.

Gainsorkuntzaren arazoa, datu-basean gordeta dagoen aldaeren informazioarekin zuzenki lotuta dago. Hitz baten hainbat aldaerek analisi bera dute, beraz, analisi batetik abiatuz sorkuntza egiterako garaian aldaera guzti horiek lortzen dira emaitza moduan. Arazo hau ekiditeko, sorkuntzarako erabiltzen den transduktorean aldaeren informazioa kenduta dagoen arren, aldaera bezala markatuta ez dauden analisi berdina duten sarrerak ere badaude datuen artean, eta horrek ere gainsorkuntza sor dezake.

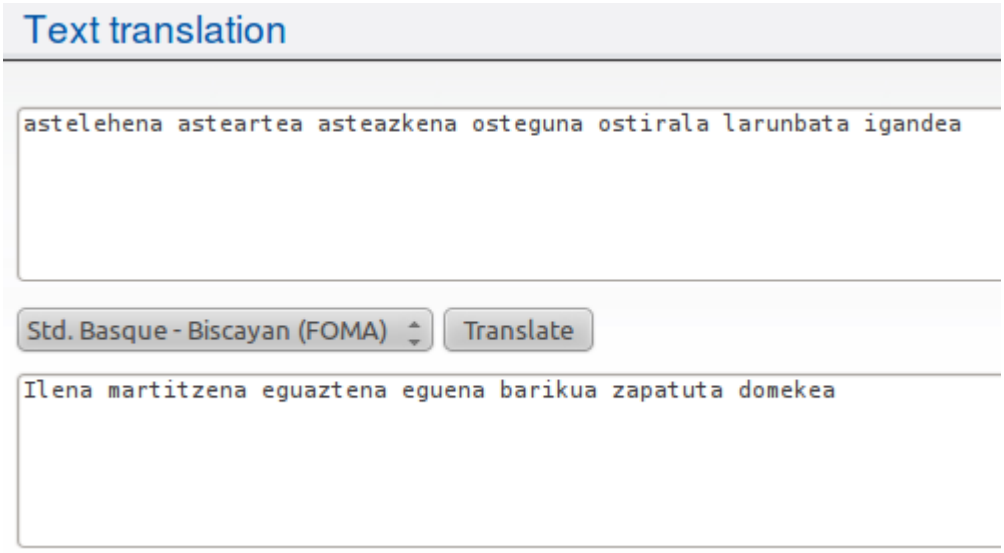
Gainsorkuntza hau ekiditeko, transduktore haztatuak sortzea probatu den arren, esperimenturako sortutako itzultzailean ezin izan da jakintza hau aplikatu. Jakin badakigu, datu-base lexikal hauek analisirako sortuak direla eta horrek eragozpenak sortzen dituela kontrako lana egiterako orduan. Etorkizunerako, datu-baseetan sorkuntzako informazioa beste moduren batean gordetzea plantea daiteke; baita *Apertium*eko transferentzian analisi morfologiko sakonagoa erabiltzea ere, anbiguotasun-maila txikitzeko. Bestalde, esperimentu honetarako eta honi konponbide

azkar bat eman asmoz, sorkuntza egiten den iturburu-kodea moldatu eta beti forma bakarra itzultzera behartu da. Forma hau, ordea, transduktoreak itzultzen dituenetatik lehenengoa izango da eta ez da emaitza optimoa izango.

Aipatutako bigarren arazoa, espero gabeko emaitzak lortzearena, euskaraz sarrera hobetsia duten sarrerekin gertatzen da. Har dezagun adibide moduan 'bariku' hitza. Hitz honen sarrera hobetsia ostiral da, eta *Hiztegi batua*n kontsulta eginez gero, 'bariku' sarrerak bizkaiera marka duela ikus dezakegu. Bigarren adibide gisa 'bortz' hartuta, hobetsitako forma 'bost' dela ikus dezakegu, eta hiztegi batuan berriz ere kontsulta eginez, Iparraldeko eta Nafarroako markak dituela ikusten da. Beraz, hobetsien kasuan, datu-base lexikaletan beste markarik ez dutenez, ezin jakin benetan bizkaierarekin lotutako informazioa den ala ez.

Bariku/ostiral adibidearekin jarraituz, euskara batuan nahiz bizkaieran bi formak onartzen dira; bariku hitzak, ordea, bizkaiera kutsua du eta hiztegi batuan bizkaiera marka. Demoa prestatzeko hartu den erabakia hau izan da: datuak eskuz editatu eta bizkaiera kutsua duten itzulpenak ematea. Prozedura hau aplikatu da hilabete izen eta aste egunekin.

Aurretik aipatu den moduan, morfologikoki konplexuak diren hizkuntzak *Apertium* bidez tratatzeak, sistemaren mantsotzea dakar. Transferentzia sintaktikoan, chunk-ak egiteko sortu behar izan diren erregela kopurua handia da, eta hauek prozesatzeko behar den denbora uste baino handiagoa da. Beraz, hasiera batean espero zena baino mantsoagoa da itzulpenak egiteko orduan.



8.1. irudia: Asteko egunen itzulpena

Irudian ikus daiteke, asteko egunak nola itzultzen dituen. Itzulpen hauek eskuz sartu dira, *Labayru Hiztegia* erreferentziatzat hartuz. Hala ere, oraindik xehetasunak fintzeke daude, larunbata formaren itzulpen gisa zapatuta itzultzen baitu zapatua itzuli ordez.

Etorkizuneko lanei dagokienez, egindako ebaluazio txiki horretan identifikatu diren arazoak konpontzeaz gain, ebaluazio sakonago bat egitea ere proposatzen da. Ebaluazio berri honekin, sistemak dituen gabezia linguistikoak detektatu ahal izango dira, hala nola, euskara batutik bizkaierara eta alderantziz gertatzen diren fenomenoak detektatzea. Fenomeno hauek kontrolatzeko, erregela berriak sortuko lirateke eta honekin batera, itzultzailearen kalitatea hobetuko litzateke.

9 Bibliografia

Aduriz, I. et al. “EDBL: A Multi-Purposed Lexical Support for the Treatment of Basque.” *Proceedings of the First International Conference on Language Resources and Evaluation*. 1998. 821–826. Print.

Aduriz, I. et al. “EUSLEM: A Lemmatiser/tagger for Basque.” *Proc. Of EURALEX’96* (1996) : 17–26. Print.

Alegria, I. et al. “A Morphological Processor Based on Foma for Biscayan (a Basque Dialect).” n. pag. Print.

Alegria, I. et al. “Automatic Morphological Analysis of Basque.” *Literary and Linguistic Computing* 11.4 (1996) : 193. Print.

Armentano-Oller, C. et al. “Apertium, Una Plataforma De Código Abierto Para El Desarrollo De Sistemas De Traducción Automática.” *Este Libro Se Distribuye Bajo Licencia Creative Commons Reconocimiento CompartirIgual 2.5 Espa* : 5. Print.

Beesley, Kenneth R., and Lauri Karttunen. *Finite State Morphology*. Center for the Study of Language and Inf, 2003. Print.

Beesley, K. R, and L. Karttunen. “Finite-state Morphology: Xerox Tools and Techniques.” *Studies in Natural Language Processing. Cambridge University Press* (2002) : n. pag. Print.

Bojar, O. “English-to-Czech Factored Machine Translation.” *Proceedings of the Second Workshop on Statistical Machine Translation*. 2007. 232–239. Print.

Bond, F. et al. “Open Source Machine Translation with DELPH-IN.” *Open-Source Machine Translation: Workshop at MT Summit X*. 2011. Print.

Brown, R. D. “Automated Generalization of Translation Examples.” *Proceedings of the 18th Conference on Computational linguistics-Volume 1*. 2000. 125–131. Print.

Brown, R. D. “Example-based Machine Translation in the Pangloss System.” *Proceedings of the 16th Conference on Computational linguistics-Volume 1*. 1996. 169–174. Print.

Canals-Marote, R. et al. “The Spanish-Catalan Machine Translation System interNOSTRUM.” *Proceedings of MT Summit VIII: Machine Translation in the Information Age*. 2001. 76. Print.

Corbí-Bellot, A. M et al. “An Open-source Shallow-transfer Machine Translation Engine for the Romance Languages of Spain.” *10th EAMT Conference” Practical Applications of Machine Translation”, Budapest, Hungary*. 2005. Print.

Garrido-Alenda, A. et al. “Shallow Parsing for Portuguese–Spanish Machine Translation.” *Tagging and Shallow Processing of Portuguese: Workshop Notes of TASHA’2003*. 2003. 21. Print.

Holmqvist, M., S. Stymne, and L. Ahrenberg. “Getting to Know Moses: Initial Experiments on German-English Factored Translation.” *Proceedings of the Second Workshop on Statistical Machine Translation*. 2007. 181–184. Print.

Hulden, M. “Foma: A Finite-state Compiler and Library.” *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics: Demonstrations Session*. 2009. 29–32. Print.

Hutchins, W. John, and Harold L. Somers. *An Introduction to Machine Translation*. Academic Press, 1992. Print.

Hutchins J., “Future prospects in Machine Translation usage and research.” *Presentation in February 2006 at the University of Leeds*, UK, 2006.

Labaka, G. (2010). “EUSMT: Incorporating Linguistic Information into SMT for a Morphologically Rich Language. Its use in SMT-RBMT-EBMT hybridation”. Lengoaia eta Sistema Informatikoak Saila (UPV-EHU).

Lindén, K., M. Silfverberg, and T. Pirinen. “HFST Tools for Morphology–An Efficient Open-Source Package for Construction of Morphological Analyzers.” *State of the Art in Computational Morphology* (2009) : 28–47. Print.

Mayor, A., and F. M Tyers. “Matxin: Moving Towards Language Independence.” (2009) : n. pag. Print.

Mayor, A. (2007). “Matxin: erregeletan oinarritutako itzulpen automatikoko sistema”. PhD thesis, Euskal Herriko Unibertsitatea.

Minervini, P. “Apertium Goes SOA: An Efficient and Scalable Service Based on the Apertium Rule-based Machine Translation Platform.” (2009) : n. pag. Print.

Nagao, M. “A Framework of a Mechanical Translation Between Japanese and English by Analogy Principle.” *Readings in Machine Translation* (2003) : 351. Print.

Nirenburg, S., S. Beale, and C. Domashnev. “A Full-text Experiment in Example-based Machine Translation.” *International Conference on New Methods in Language Processing (NeMLaP)*. 1994. 78–87. Print.

Papineni, K. et al. “BLEU: A Method for Automatic Evaluation of Machine Translation.” *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. 2002. 311–318. Print.

Sato, S., and M. Nagao. “Toward Memory-based Translation.” *Proceedings of the 13th Conference on Computational linguistics-Volume 3*. 1990. 247–252. Print.

10 Gehigarriak

10.1 Probabilitate fitxategiak sortzen

Atal honetan, etiketatze morfologikoa gauzatzeko egin diren pausu guztiak xehetasunez azalduko dira. Baita honetarako erabiltzen den *makefile* fitxategiaren iturburu-kodea ere.

1.- Iturburu-hizkuntzako corpusa hartu eta create_expanded.py scriptaren laguntzaz, corpuseko hitz bakoitza lerro batean jarri da ']' eta '.[' karaktereen artean.

```
python create_expanded.py eu-tagger-data/eu.crp.txt | sort |
uniq | egrep '^\]' > eu-tagger-data/eu.dic.expanded
```

eu.dic.expanded fitxategiak horrelako itxura izango du:

```
]
]%07.[
]%48.[
]%4raino.[
]%50.[
]abendura.[
]adierazi.[
]agertu.[
]ahal.[
]aitortza.[
]aitortzen.[
]apur.[
]ardatz.[
```

2.- eu.dic.expanded fitxategiko sarrera bakoitzari analisi bat esleitzen zaio modu honetan

```
foma-proc -a eu-eu_bis-morf.fst < eu-tagger-data/eu.dic.expanded
| iconv -c -t utf-8 > eu-tagger-data/eu.dic.expanded.analized
```

eu.dic.expanded.analized fitxategiaren itxura honakoa izango da:

```
^]/]$
^]%07/]%07$.[
]^%48/%48$.[
]^%4raino/%4raino$.[
]^%50/%50$.[
]^abendura/gabonil<n>+0<post><sg><M>+ErA<post><ALA>$.[
]
```

```
^adierazi/adierazo<vblex>+0<AMM><PART>/adierazo<vblex>+0<AMM><PART>+0
<post><ABS><MG>/adierazo<vblex>+0<AMM><PART>+0<ASP><BURU>/adierazo<n>
/adierazo<n>+0<post><ABS><MG>$.[
]
^agertu/ageR<vblex>+tu<AMM><PART>/ageR<vblex>+tu<AMM><PART>+0<post><A
BS><MG>/ageR<vblex>+tu<AMM><PART>+0<ASP><BURU>$.[
]^ahal/ahal<n>/ahal<n>+0<post><ABS><MG>/al<adv>$.[
]
^aitortza/autortzA<n>/autortzA<n>+0<post><ABS><MG>/autortzA<n>+5a<pos
t><ABS><sg><M>$.[
]
^aitortzen/autortzA<n>+eM<post><GEN><pl><M>/autortzA<n>+eM<post><GEN>
<pl><M>+0<post><ABS><MG>/autor<vblex>+tze<AMM><ADIZE>+n<post><INE>/au
tor<vblex>+0<AMM><ADOIN>+tzen<ASP><EZBU>$.[
]
^apur/apuR<n>/apuR<n>+0<post><ABS><MG>/apuR<vblex>+0<AMM><ADOIN>/apuR
<adj>/apuR<adj>+0<post><ABS><MG>$.[
]
^ardatz/ardatz<n>/ardatz<n>+0<post><ABS><MG>/ardatz<vblex>+0<AMM><ADO
IN>$.[
```

3.- *Tsx* fitxategian dauden anbiguotasun-klaseak kontutan hartuz, hiztegi morfologikoa sortu:

tsx fitxategiko definizio baten adibidea:

```
<def-label name="NOM">
      <tags-item tags="n"/>
      <tags-item tags="n.*"/>
</def-label>
```

```
apertium-filter-ambiguity apertium-eu-eu_bis.eu.tsx < eu-tagger-
data/eu.dic.expanded.analized > eu-tagger-data/eu.dic
```

4.- *eu.dic* hiztegiaren arabera etiketatu sarrerako corpusa eta eu.crp sortu.

```
make -f eu-eu_bis-unsupervised.make eu-tagger-data/eu.crp
```

```
5.- Probabilitate fitxategia sortu.
make -f eu-eu_bis-unsupervised.make eu-eu_bis.prob
```

eu-eu_bis-unsupervised.make fitxategiaren itxura, honakoa izango da:

```
TAGGER_UNSUPERVISED_ITERATIONS=8

BASENAME=apertium-eu-eu_bis

LANG1=eu
```

```
LANG2=eu_bis
TAGGER=$(LANG1)-tagger-data
PREFIX=$(LANG1)-$(LANG2)

all: $(PREFIX).prob

$(PREFIX).prob:   $(BASENAME).$(LANG1).tsx   $(TAGGER)/$(LANG1).dic   $
(TAGGER)/$(LANG1).crp
        apertium-validate-tagger $(BASENAME).$(LANG1).tsx
        apertium-tagger -t $(TAGGER_UNSUPERVISED_ITERATIONS) \
                           $(TAGGER)/$(LANG1).dic \
                           $(TAGGER)/$(LANG1).crp \
                           $(BASENAME).$(LANG1).tsx \
                           $(PREFIX).prob;

$(TAGGER)/$(LANG1).crp: $(PREFIX).fst $(TAGGER)/$(LANG1).crp.txt
          apertium-destxt < $(TAGGER)/$(LANG1).crp.txt | foma-proc $
(PREFIX).fst > $(TAGGER)/$(LANG1).crp

clean:
        rm -f $(PREFIX).prob
```

Prozedura hau erabili da bi norabideetarako beharrezko probabilitate fitxategiak sortzeko.

10.2 Instalazio gida

Atal honetan, *BiscayTrad* sistema instalatzeko jarraitu beharreko pausuak zehazten dira, pausuz pausu:

Dependentziak instalatu

```
sudo apt-get install libxml2-dev flex libpcre3-dev libxml2-utils
xsltproc
```

Apertiumen iturburu-kodea jaitsi, konpilatu eta instalatu

```
svn co
https://apertium.svn.sourceforge.net/svnroot/apertium/trunk/apert
ium
./autogen.sh
make
sudo make install
```

Lttoolboxen iturburu-kodea jaitsi, konpilatu eta instalatu

```
svn co
https://apertium.svn.sourceforge.net/svnroot/apertium/trunk/lttoo
lbox
cd lttoolbox
./autogen.sh
make
sudo make install
```

Bytecode for transfer proiektuko iturburu-kodea jaitsi, konpilatu eta instalatu

```
svn co
https://apertium.svn.sourceforge.net/svnroot/apertium/trunk/lttoo
lbox-java
cd lttoolbox-java
sudo apt-get install maven2
sudo apt-get install default-jdk
mvn install
make
sudo make install
```

Foma erremintaren iturburu-kodea jaitsi, konpilatu eta instalatu

```
svn co https://foma.svn.sourceforge.net/svnroot/foma/trunk/foma
foma_svn
cd foma_svn
sudo apt-get install bison libreadline5-dev libreadline5 flex
libedit-dev
make CFLAGS=-fPIC
sudo make install
```

HFST proiektuaren iturburu-kodea jaitsi, konpilatu eta instalatu

```
wget openfst.cs.nyu.edu/twiki/pub/FST/FstDownload/openfst-
1.2.7.tar.gz
tar xzvf openfst-1.2.7.tar.gz
./configure
make
sudo make install

svn co https://hfst.svn.sourceforge.net/svnroot/hfst/trunk hfst
sudo apt-get install libglib2.0-dev
```

```
cd hfst/hfst3
./autogen.sh
./configure --without-sfst
make
sudo make install
```

Apertiumetik, foma-proc martxan jartzeko iturburu-kodea jaitsi, konpilatu eta instalatu

```
svn co
https://apertium.svn.sourceforge.net/svnroot/apertium/branches/fo
ma/src/ foma-proc
cd foma-proc
make CFLAGS=-fPIC
sudo make install
sudo cp foma-proc /usr/local/bin
```

Apertiumen oinarritutako euskara batua eta bizkaieraren arteko itzultzaile automatikoko baliabide linguistikoak jaitsi, konpilatu eta instalatu

```
svn co
https://apertium.svn.sourceforge.net/svnroot/apertium/branches/ap
ertium-eu-eu_bis-foma
./autogen.sh
make
sudo make install
```

10.3 SEPLN2011

Gehigarri moduan, SEPLN2011rako prestatutako dokumentazioa erantsi da. Jarraian datozen dokumentuak, aipatutako kongresuan erabilitako materialak dira.

Gestión de la información morfológica para la creación de un nuevo par de lenguas con distintos dialectos en un sistema de traducción automática de código abierto

Morphological information management for the creation of a new pair of languages with different dialects in an open-source machine translation system

Garbiñe Aranbarri Ariztondo
Eleka Ingeniaritza Linguistikoa S.L.
Zelai Haundi kalea 3, Osinalde industrialdea,
20170 Usurbil, Gipuzkoa
garbine@eleka.net

Itziar Cortés Etxabe
Eleka Ingeniaritza Linguistikoa S.L.
Zelai Haundi kalea 3, Osinalde industrialdea,
20170 Usurbil, Gipuzkoa
itziar@eleka.net

Resumen: Se presenta una demostración del trabajo realizado para crear un sistema de traducción automática entre el euskera *batua* estándar y el vizcaíno estándar (un dialecto del euskera). La estandarización y la generación de los datos lingüísticos de ambas variantes, y los progresos tecnológicos realizados en *Apertium* han facilitado el manejo de datos de forma más eficiente para crear este traductor de código abierto.
Palabras clave: morfología, bases de datos lexicales, dialecto, *Apertium*, código abierto, *Foma*.

Abstract: This is a demonstration of the work procedure to create a machine translation system between standard Basque and Biscayan (a Basque dialect). The standardization and creation of linguistic data of both variants, and the technological progress made on *Apertium*, have helped to manage data on a more efficient way to create this open-source software translator.
Keywords: morphology, lexical databases, dialect, *Apertium*, open-source, *Foma*.

1 Introducción

El vizcaíno es un dialecto del euskera con un estándar definido (*Instituto Labayru Ikastegia*), con el cual ya se han realizado trabajos relacionados con PNL, como es el caso del corrector ortográfico para el vizcaíno *XuxenB* (Alegria, I. et al., 2010). Gracias a este último trabajo, existen recursos morfológicos tanto para el vizcaíno como para el euskera estándar. Además, contamos con una herramienta adecuada y en código abierto para crear traductores entre lenguas cercanas llamada *Apertium* (Ramírez-Sánchez, G. et al., 2006), cuya comunidad está trabajando en estos momentos en la integración de la tecnología *Finite State Transducers*, y para lo cual ha utilizando, entre otras tecnologías, *Foma* (Hulden, M., 2009). A continuación se presentan las fuentes que se han utilizado para crear un traductor entre el euskera estándar (*euskara batua*) y el vizcaíno (*bizkaiera*).

2 Objetivo

El objetivo es el siguiente: crear un traductor de código abierto entre el par de lenguas euskera batua (*eu*) y vizcaíno (*eu_bis*) utilizando la tecnología *Apertium* de la forma más eficiente posible.

Se estudiaron dos posibilidades para el manejo de datos: una consistiría en la modificación de los diccionarios morfológicos existentes para el euskera estándar en *Apertium,* y en la creación de nuevos diccionarios para el vizcaíno; la otra opción consistiría en la utilización de *Apertium* y *Foma* conjuntamente. Hemos desarrollado la segunda opción.

Aprovechando la posibilidad que ofrece Apertium para trabajar con *Foma,* y la experiencia que tenemos a la hora de tratar la información morfológica en estos formatos, se ha considerado que ésta podría ser una buena oportunidad para explotar las bases de datos lexicales del vizcaíno (*BIDBL*) (usado

anteriormente para el proyecto *XuxenB*) y del euskera *batua* (*EDBLO*) (usado para las diversas herramientas PLN que se han implementado para el euskera *batua*). La ventaja de este método es que ofrece la posibilidad de comenzar a trabajar con un léxico completo y de reducirlo a la hora de liberar los datos para publicarlos en el repositorio de *Apertium*, ya que dichos datos, por ahora, son privados.

3 Desarrollo

Con la información de las bases de datos lexicales se ha creado la información necesaria para que la transferencia léxica esté integrada en el análisis morfológico. Así, se han creado relaciones entre las dos bases de datos (vizcaíno y euskera *batua*) de forma que se consigue unir la entrada de una base de datos con el análisis de la otra, obteniendo así un análisis en la lengua de destino. Es decir, para traducir en la dirección *eu* → *eu_bis* (*batua* → vizcaíno) el análisis morfológico de la forma en *eu* dará como resultado su correspondiente análisis en *eu_bis*. A través de ese procedimiento se integra la transferencia léxica con el análisis morfológico.

Tomemos como ejemplo la forma en *eu* 'eraman', cuyo análisis después de haber sido desambiguado es el siguiente: *^eroaN<vblex>+0<AMM><PART>+0<post><ABS><MG>$*. Este análisis se corresponde con el del vizcaíno, y teniendo en cuenta la información de la base de datos de esta variante, se ha generado la forma léxica correcta: 'eroan'.

Puede ocurrir que en *BiDBL* tengamos una referencia a una forma del euskera *batua* que no aparece en *EDBL*, ya que el enriquecimiento lexical de las bases de datos se ha realizado de manera independiente. Es por ello que para generar las formas en la lengua de destino se ha utilizado una combinación de transductores creados con el operador *priority union* de *Foma*. Se ha combinado el transductor de generación (lengua de destino) con el de la información morfológica para el análisis (lengua de origen). Así, aseguraremos que siempre habrá una forma en la lengua de destino.

Otros problemas que se han encontrado en la implementación de este proyecto son los relacionados con las características propias del euskera y su integración en un sistema como *Apertium*, además de los derivados de los datos que hemos obtenido de nuestras bases de datos. Si nos centramos en estos últimos, cabe mencionar que las bases de datos utilizadas fueron creadas, en un principio, para el análisis morfológico del euskera *batua* y del vizcaíno. Es decir, en el momento de su creación, no se había previsto utilizar las dos bases de datos cruzadas para un único proyecto.

4 Mejoras y trabajo futuro

La revisión de los datos creados es uno de los trabajos 'no automáticos' que ha de hacerse después de terminar con el tratamiento automático de los mismos. Las bases de datos *BiDBL* y *EDBLO* carecen de cohesión en una serie de aspectos, y es por ello que resulta imprescindible que sean revisadas. Por consiguiente, se prevé un replanteamiento de la estructura y de los datos de las bases de datos.

Tras la revisión, se procederá a reducir el léxico para poder liberar parte de los datos generados.

En cuanto a la implementación del sistema, se cambiarán los autómatas en formato *Foma* para convertirlos al formato *HFST* (Lindén, K. et al., 2009), gracias al cual se logrará una mejor integración de las bases de datos en *Apertium*.

5 Referencias

Alegria, I. et al. “A morphological processor based on foma for Biscayan (a Basque dialect).”. 2010.

Hulden, M. “Foma: a finite-state compiler and library.” *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics: Demonstrations Session*. 2009. 29–32.

Lindén, K., M. Silfverberg, and T. Pirinen. “HFST Tools for Morphology–An Efficient Open-Source Package for Construction of Morphological Analyzers.” *State of the Art in Computational Morphology* (2009)

Ramírez-Sánchez, G. et al. “Opentrad Apertium open-source machine translation system: an opportunity for business and research.” *Proceedings of the 28th International Conference on Translating and the Computer*. 2006.

ingeniaritza ELEKA linguistikoa

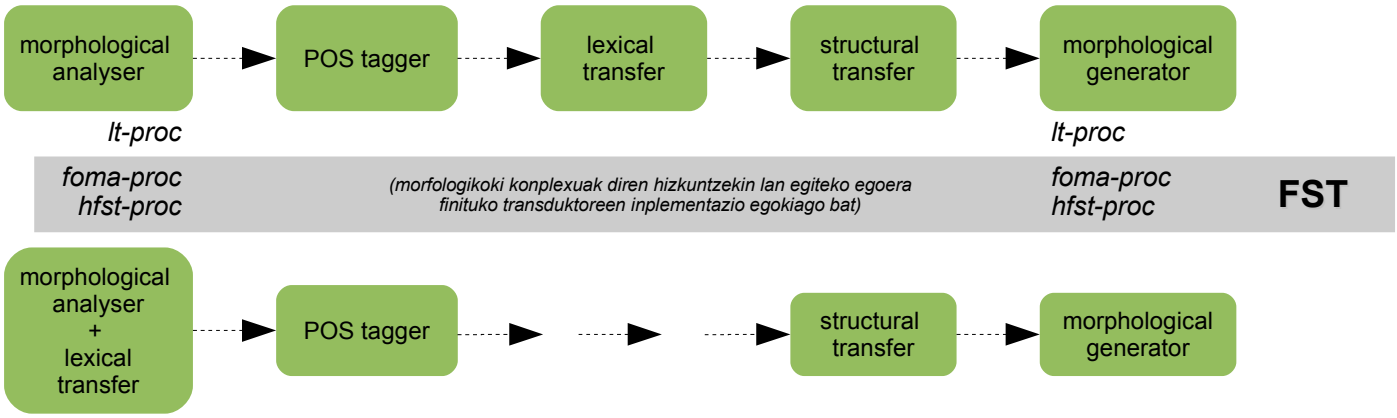
Morphological information management for the creation of a new pair of languages with different dialects in an open-source machine translation system

Abiapuntua

- ***Apertium*** kode irekiko itzulpen-automatikoko sistema.
- Euskara batuaren eta bizkaieraren gramatikaren deskribapena gordetzen dituzten datu-baseetan oinarrituz beharrezko ***datuak*** esportatu.
- ***Finite State Transducers (FST)*** teknologian esperientzia morfologikoki konplexuak diren hizkuntzen datuak maneiatzeko orduan.

Apertium

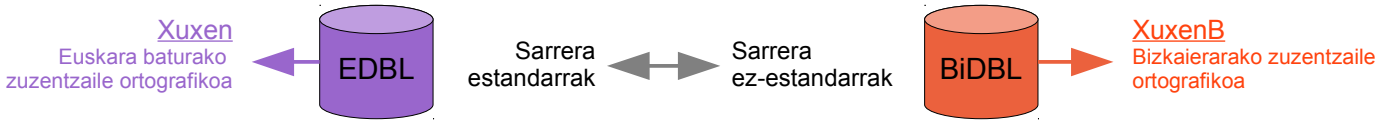
Itzulpen-prozesuaren fluxuaren aldaketa



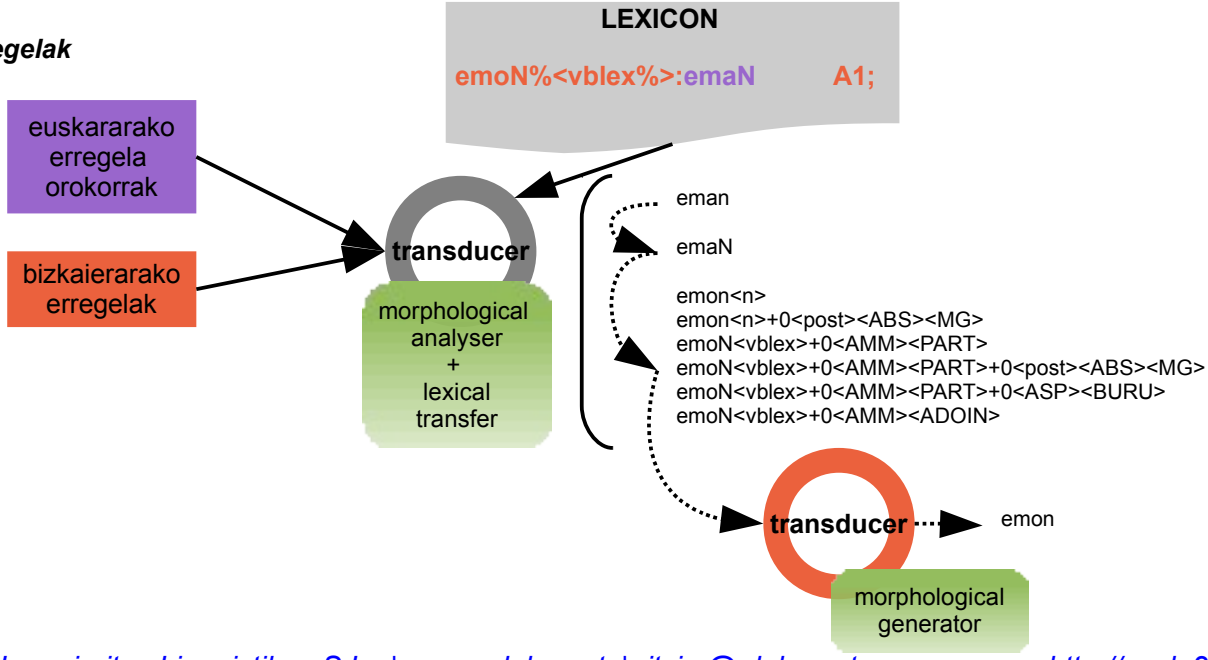
Datuak

Euskararen Datu-Base Lexikala (EDBL)
Euskara estandarraren gramatikaren deskribapena gordetzen duen datu-basea da. Hiztegi batuko informazioan oinarrituta dago eta FST teknologia erabili ahal izateko prestatuta dago. Euskara osoaren deskribapena biltzen du, beraz, aldaera dialektalen informazioa ere gordeta dago.

Bizkaieraren Datu-Base Lexikala (BiDBL)
EDBLk duen egitura bera du; informazio-mota berdina gordetzen da. Labayru Institutuak definitutako estandarrean oinarrituta dago.



Erregelak



Gestión de la información morfológica para la creación de un nuevo par de lenguas con distintos dialectos en un sistema de traducción automática de código abierto

Morphological information management for the creation of a new pair of languages with different dialects in an open-source machine translation system

Este informe presenta una demostración del trabajo realizado para crear un sistema de traducción automática entre el euskera estándar y el vizcaíno, un dialecto del euskera. La estandarización y la generación de los datos lingüísticos de ambas variantes, y los progresos tecnológicos de Apertium, han sido fundamentales para lograr manejar los datos de forma eficiente, y poder crear este traductor de código abierto.

This is a demonstration of the work procedure to create a machine translation system between standard Basque and Biscayan (a Basque dialect). The standardization and creation of linguistic data of both variants, and the technological progress made on Apertium, have been very important in order to manage the data on a more efficient way, and to create this open-source software translator.

Antecedentes

Al comienzo de este proyecto se estudiaron las distintas posibilidades a considerar para poder crear un sistema de traducción automática de código abierto, para lo cual se analizaron las tecnologías y recursos utilizados en proyectos de este tipo. Por lo tanto, se han analizado sistemas de traducción automática basados tanto en estadística como en reglas.

Entre los sistemas estadísticos libres con los que tenemos experiencia cabe destacar Moses[1], un sistema de traducción automática estadística bajo licencia LGPL. Se puede utilizar para entrenar modelos de traducción para cualquier par de lenguas, partiendo de un corpus paralelo bilingüe, lo más grande y de mejor calidad posible.

También se han analizado sistemas basados en reglas, como es el caso de Matxin[2] y Apertium[3], proyectos que forman parte de la plataforma de traducción automática de código abierto llamado Opentrad[4].

Matxin se asienta en el clásico modelo basado en la transferencia (Arnold et al., 1994), cuyo sistema tiene una arquitectura independiente a la lengua de origen y de destino, y a la distancia entre ellas. Fue

1 http://www.statmt.org/moses/
2 http://matxin.sourceforge.net/
3 http://www.apertium.org/
4 http://www.opentrad.com

creado para traducir en la dirección EU>ES. Matxin es un software libre y tiene la licencia GNU GPL. Por lo tanto, su código y sus recursos lingüísticos pueden ser utilizados para cualquier otra aplicación perteneciente a esa misma licencia.

Apertium es un sistema de traducción automática de código abierto, y fue creado para traducir entre lenguas cercanas.

Tras analizar esos tres sistemas (Moses, Matxin y Apertium), Apertium ha sido el sistema elegido para desarrollar este experimento.

Se ha descartado utilizar el sistema estadístico, ya que para ello se necesita un corpus paralelo que no se dispone, y resulta imposible conseguir un corpus en esos dos dialectos lo suficientemente grande como para obtener buenos resultados. Es más, incluso si se lograra obtener un corpus del tamaño adecuado, habría que repasarlo entero para cerciorarse de que las pautas que sigue el corpus son las mismas que se siguieron cuando se crearon los correctores ortográficos para el euskera batua y el vizcaíno.

Una vez descartado el sistema estadístico, sólo quedaba decidir entre las otras dos opciones analizadas, es decir, entre Matxin o Apertium. Se consideró que la utilización de la tecnología Matxin sería muy costosa, puesto que se necesitarían analizadores lingüísticos para los dos dialectos, además de otra serie de recursos lingüísticos. Apertium, sin embargo, se utiliza de forma operativa y con muy buenos resultados para los pares de lenguas cercanas. Teniendo en cuenta que el objeto de estudio trata de la creación de un sistema de traducción entre dos dialectos de la misma lengua, Apertium resultó ser, sin duda, la mejor de las tres opciones posibles.

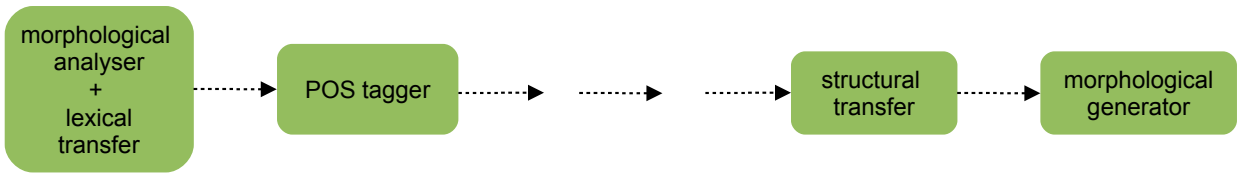
Objetivo

El objetivo de este experimento es crear un traductor automático libre entre el par de lenguas euskera batua (euskera estándar) y vizcaíno (un dialecto del euskera), en base a la plataforma Apertium. Se estudian dos formas posibles de desarrollo del nuevo par de lenguas en Apertium:

La primera opción consiste en crear un diccionario monolingüe para cada lengua, y un diccionario bilingüe donde se recogen las equivalencias necesarias para la traducción. En este caso, se puede aprovechar el diccionario del euskera batua del proyecto Matxin. Así, sólo quedan por crear el diccionario para el vizcaíno y el bilingüe.

Apertium se ejecutaría de este modo:

La segunda opción consiste en utilizar combinaciones entre las dos bases de datos con las que trabajamos: EDBLO (Base de datos lexical del euskera, o *Euskararen Datu-Base Lexikala*) y BiEDBL (Base de datos lexical del vizcaíno, o *Bizkaieraren Datu-Base Lexikala*). Gracias a la combinación de esos datos, se podría desarrollar un analizador/traductor morfológico, para lo cual se crearían analizadores que analizaran la forma de la lengua de origen en la lengua de destino. Dicho de otra manera, el análisis morfológico y la transferencia léxica se ejecutarían al mismo tiempo, tal y como se muestra en la siguiente imagen:



Desarrollo del proyecto

Entre las dos opciones posibles se ha elegido la segunda, cuyo desarrollo se ha dividido en dos partes: por una parte se realiza la manipulación de los datos lingüísticos, y por otra se integra la tecnología FST en Apertium, que además de soportar lttoolbox ahora trabaja también con Foma (software con el cual ya tenemos experiencia) y con HFST; estos dos últimos (Foma y HFST) son más adecuados para la manipulación de datos lingüísticos pertenecientes a idiomas de morfologías complejas. Este documento explica en qué consiste la manipulación de los datos lingüísticos, tema central de la demostración presentada en este congreso.

Los datos lingüísticos que se han utilizado en este proyecto son exportaciones de las bases de datos EDBL y BiDBL. Estas dos fuentes de información han sido ya utilizadas en otros proyectos anteriores, como es el caso de Xuxen[5] y XuxenB[6], correctores ortográficos para el euskera batua y para el vizcaíno, respectivamente, así como en el desarrollo de un analizador morfológico para el euskera[7] (utilizando los datos de EDBL). Estas dos bases de datos tienen la información suficiente como para poder ser exportadas a gramáticas que se puedan compilar con software basado en máquinas de estados finitos. En este caso, se utilizan transductores de estados finitos (FST, finite-state transducers), es decir, máquinas de estados finitos con dos cintas: una de entrada y otra de salida. Esto quiere decir que cada entrada (input) obtendrá como resultado una salida (output).

[5] Agirre, E. et al. "XUXEN: A Spelling Checker/corrector for Basque Based on Two-Level Morphology." *Proceedings of the Third Conference on Applied Natural Language Processing*. 1992. 119–125. Print.

[6] Alegria, I. et al. "*A Morphological Processor Based on Foma for Biscayan (a Basque Dialect).*"

[7] Aduriz, I. et al. "EUSLEM: A Lemmatiser/tagger for Basque." *Proc. Of EURALEX'96* (1996) : 17–26. Print.

Para crear los cuatro transductores necesarios para el flujo de la traducción, se han creado cuatro exportaciones diferentes de las bases de datos:

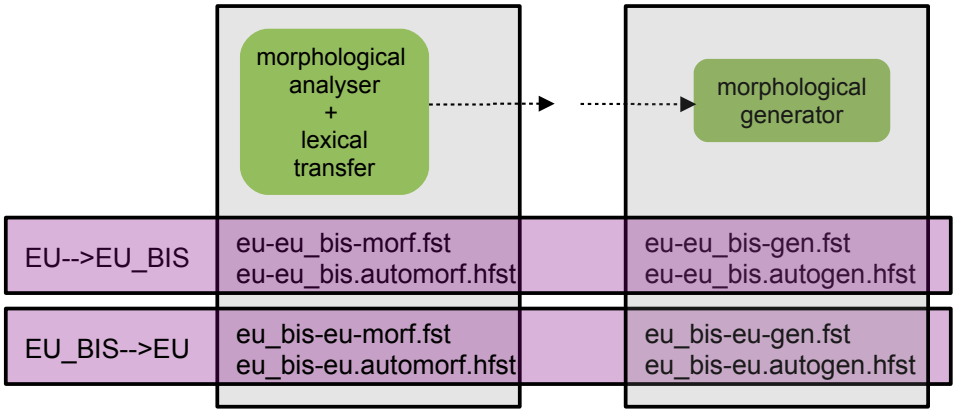
a) Para la dirección euskera batua → vizcaíno:

- Analizador morfológico[8] que analiza las formas superficiales del euskera batúa en vizcaíno: *eu-eu_bis-morf.fst* y *eu-eu_bis.automorf.hfst*.
- Analizador morfológico que analiza las formas en vizcaíno: *eu-eu_bis-gen.fst* y *eu-eu_bis.autogen.hfst*.

b) Para la dirección vizcaíno → euskera batua:

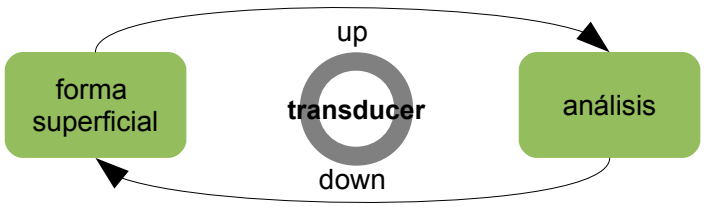
- Analizador morfológico que analiza las formas superficiales del vizcaíno en euskera batúa: *eu_bis-eu-morf.fst* y e*u_bis-eu.automorf.hfst*.
- Analizador morfológico que analiza las formas en euskera batua: *eu_bis-eu-gen.fst* y *eu_bis-eu.autogen.hfst*.
-

Gráficamente, podríamos explicarlo de este modo:

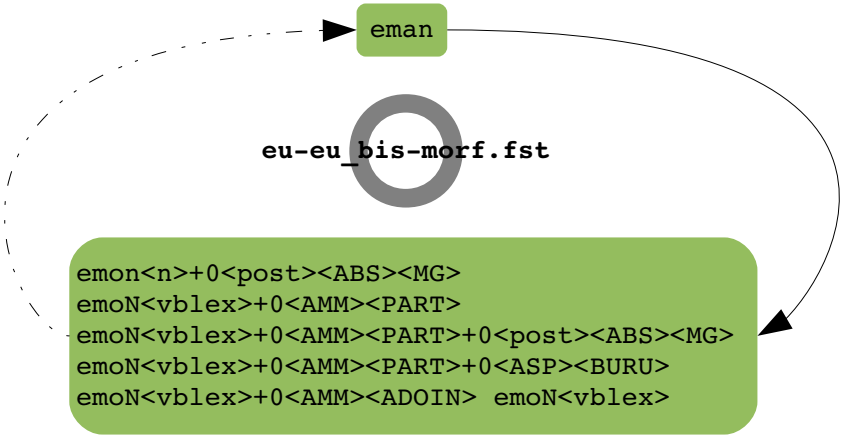


[8] Llamamos analizador morfológico a todos los transductores creados, ya que aplicando la dirección *up*, obtenemos el análisis de la forma de la lengua de origen.

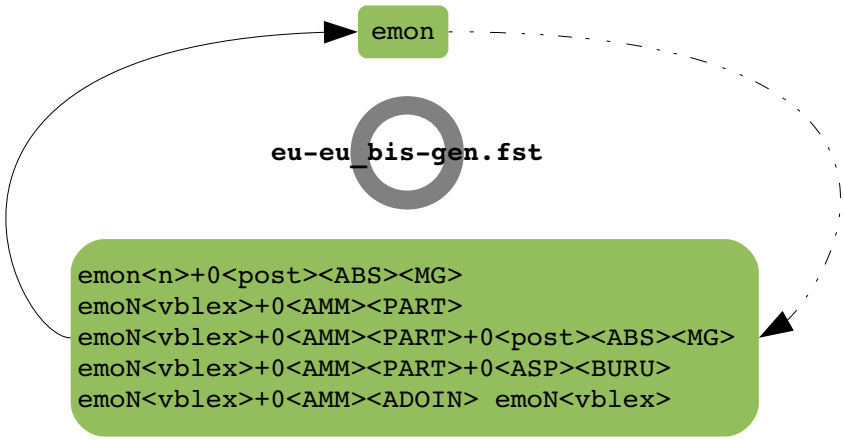
La utilización de transductores permite analizar y generar utilizando el mismo analizador. En cuanto a los analizadores creados para cada dirección de este traductor, el primero creará un análisis partiendo de una forma (up), mientras que el segundo analizador será utilizado para generar una forma en la lengua de destino utilizando el analizador para generar (down).



Tomamos como ejemplo el verbo *eman* en euskera batua, que se analiza de esta manera:



Si partimos de cualquiera de esos análisis sería posible crear la forma correcta en vizcaíno: *emon*.



Este caso sería óptimo, ya que con cualquiera de los análisis obtenidos se puede conseguir una forma en la lengua de destino utilizando el analizador que se ha creado para generar en la lengua de destino.

En otros casos, la cohesión de los datos obtenidos no es completa. Si se toma como ejemplo la forma *fedegabe* del euskera batua, y se analiza con el analizador creado, el resultado obtenido no es el esperado. Tal y como se puede ver en el ejemplo, el tercer análisis carece de la transferencia léxica. Esto supone un problema a la hora de generar, ya que en el generador que utilizaremos no existe ninguna forma en vizcaíno que tenga este análisis.

```
echo fedegabe | apertium -d . eu-eu_bis-foma-anmor
^fedegabe/
      fedebako<adj>/
      fedebako<adj>+0<post><ABS><MG>/
      fedegabe<vblex>+0<AMM><ADOIN>$
```

Para evitar que al final de la cadena de Apertium existan análisis sin generar, se ha utilizado el comando *priority union* que proporciona Foma, para que si en algún momento la información morfológica del vizcaíno no fuera suficiente, se consulte la información del analizador del euskera batua.

```
define LEXBISGEN @"eu_gen1.fst" ;  (información morfológica del vizcaíno)
define LEXEUMORF @"eu_morf.fst";   (información morfológica del euskara batua)

regex LEXBISGEN .P. LEXEUMORF ;
```

Así, el transductor final creará la forma *fedegabe.*

Otro de los aspectos que se ha investigado para la creación de esta demostración ha sido la sobregeneración de formas, ya que como se ha explicado anteriormente, la información lingüística utilizada tiene también análisis relacionados con otros dialectos del euskera.

Teniendo en cuenta que es probable que se produzca sobregeneración en la generación de formas en la lengua de destino, se ha hecho un experimento utilizando la herramienta HFST, teniendo en cuenta que ofrece la posibilidad de crear ficheros de probabilidades y combinar dichos ficheros para obtener un único transductor con formas ponderadas.

Este es un ejemplo de los experimentos desarrollados:

- *priorities.txt*: fichero en el cual anotamos las formas con sus prioridades.

```
ostiral 1
bariku  1000
```

- *prueba.lex*: fichero lexc con la información morfológica necesaria.

```
Multichar_Symbols
%<n%>

LEXICON Root
bariku%<n%>:ostiral J ;
bariku%<n%>:bariku J ;

LEXICON J
# ;
```

Partiendo de este último fichero, creamos un transductor en Foma (herramienta con la que estamos más familiarizados) llamado *prueba.lex.fst*. Tomando como base este transductor creamos otro en formato OpenFST, para lo cual se utiliza el siguiente comando:

```
gzip -d -c prueba.lex.fst | hfst-fst2fst -t -o prueba.lex.hfst
```

Intentamos generar el análisis *bariku<n>* obteniendo como resultado la forma *bariku* y *ostiral*.

```
hfst-lookup prueba.lex.hfst
      bariku<n>
      bariku<n>       bariku  0.000000
      bariku<n>       ostiral 0.000000
```

Una vez creado este transductor, se combina con el fichero de prioridades creado anteriormente de esta manera:

```
      hfst-strings2fst -j priorities.txt | hfst-compose -1 prueba.lex.hfst -o
out.hfst
```

Ahora, el resultado obtenido está ponderado de manera que la prioridad de *bariku* sería mayor que la de *ostiral*.

```
hfst-lookup out.hfst
      bariku<n>
      bariku<n>    ostiral    1.000000
      bariku<n>    bariku    1000.000000
```

Estos tipos de transductores podrían solucionar el problema, pero por ahora no se ha podido combinar con el programa hfst-proc.

Evaluación

Para la evaluación del sistema se ha partido de un corpus en vizcaíno utilizado anteriormente en la evaluación del corrector ortográfico XuxenB. Se han extraído 55 frases de dicho corpus de forma aleatoria, se han traducido dichas frases al euskara batua, y finalmente, se han traducido automáticamente desde el euskera batua al vizcaíno utilizando el sistema recién creado.

De entre las 55 frases traducidas automáticamente, 20 han sido traducidas correctamente. Las causas posibles de las incorrecciones han podido ser las siguientes:

- Problemas de desambiguación en el POS Tagger.
- Información incompleta en las bases de datos.
- Reglas de transferencia erróneas.

Conclusiones y trabajo futuro

Tras terminar con el experimento de enlazar las dos bases de datos (EDBL y BiDBL), aprovechando así la información creada anteriormente para el análisis y la creación de correctores ortográficos, se ha llegado a la conclusión de que sería casi imprescindible hacer cambios en la estructura de las bases de datos para relacionarlos entre sí de forma óptima. Sería necesario extender los campos relacionados con la información sobre dialectos, ya que en EDBL esa información no es del todo precisa. Esto ayudaría a la hora de enlazar las dos bases de datos.

Por otro lado, sería interesante añadir campos con los que se pudiera diferenciar entre información para el análisis e información para la generación.

Si se examinan esas cuestiones, se podría mejorar el sistema de traducción automática sólo con cambiar los transductores.

En cuanto a las reglas de transferencia y otras partes de Apertium, como trabajo futuro se propone un repaso general desde un punto de vista lingüístico. Sería de gran ayuda realizar un trabajo de comparación entre un texto en euskera batua y otro en vizcaíno, es decir, crear un corpus comparable, desde donde se pudieran extraer patrones de transformación , y otra serie de fenómenos, temas que precisan de un estudio más profundo que el realizado en esta demostración, ya que se trata del primer desarrollo del sistema de traducción automática entre el euskera batua y el vizcaíno.

Otro modo de mejorar las traducciones en general y el proceso de generación en particular sería profundizar más en el tema mencionado anteriormente, es decir, en la utilización de transductores ponderados partiendo de un fichero de prioridades combinado con cualquiera de los transductores que se han utilizado para esta demostración. Dicho fichero de prioridades podría ser extraído automáticamente de un corpus en la lengua de destino.

Agradecimientos

Este experimento ha sido posible gracias a la inestimable colaboración de los especialistas en sistemas de traducción automática en código libre Francis Tyers y Mireia Ginestí.