Universidad del País Vasco
Euskal Herriko Unibertsitatea

Errore ortografikoen azterketa web-eko dokumentuetan

Egilea: Izaskun Etxeberria
Tutorea: Iñaki Alegria

hap

Hizkuntzaren Azterketa eta Prozesamendua Masterreko titulua lortzeko bukaerako proiektua

2010eko iraila

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

Laburpena

Lan honetan euskarazko errore ortografikoen azterketa sakona egin nahi da web-etik jasotako hainbat corpusetan, horrela corpus horien kalitatea estimatzeko. Analisi horren bitartez, kalitate minimoa ez duten dokumentuak detektatzeko filtro bat lortu nahi da eta, bide batez, errore ortografiko mota bakoitzaren presentzia estimatu. Metodologia finkatzeko, ingeleserako eta alemanerako egin den antzeko lanean oinarritu gara (Ringlstetter et al., 2006), baina ez dugu teknologia bera erabili erroreak identifikatzeko. Euskarak morfologia aberatsa duenez, erroreak identifikatzeko aurretik garatutako zuzentzaile ortografikoak berrerabili ditugu. Bide horretatik, detekzioaren estaldura handiagoa da eta, gainera, prozesuaren garapena azkarragoa da berrerabilpena dela eta.

Abstract

This work's aim is to analize the quality of corpora retrieved from the Basque Web. The main objective of the analysis is to detect documents containing errors over a given threshold. The second objective is to estimate how many errors are of each class. The metodology folowed is similar to that used for English and Germany by (Ringlstetter et al., 2006). The main difference lies in the fact we reuse spelling checkers for detecting errors. By this way, we obtain a higher error coverage and also, we can develop the work faster because of reusing previous tools.

Gaien aurkibidea

Taulen zerrenda

Irudien zerrenda

1 Proiektuaren motibazioa eta helburuak

Ohikoa da gaur egun testu-bildumak osatzea Internetetik jasotako hainbat dokumenturekin, Internetek aukera paregabea eskaintzen baitu corpus handiak eta egungoak osatzeko. Baina, nolako kalitatea dute hala osatutako corpusek? Lan honetan aztertu nahi dugu zein eta zenbat errore aurkitzen ditugun Internetetik eskuratutako euskarazko testuetan, eta posible bada, tresna automatiko bat eskaini nahi dugu zeinaren bitartez posible izango den corpusaren kalitate minimo bat bermatzea. Hori lortzeko, nahitaezkoa da azterketa sakon bat egitea eta horretarako hainbat corpus osatuko dugu irizpide ezberdinen arabera, azterketa ahalik eta osoena izateko. Azterketa hori aurrera eramateko metodologia *Computational Linguistics* aldizkarian argitaratuko lan azpimarragarri batean oinarrituta dago, (Ringlstetter et al., 2006) artikuluan, hain zuzen. Artikulu horretan egiten den analisia ingeleserako eta alemanerako da, eta bertan erabiltzen duten metodologian zein irizpidetan oinarrituko dugu gure azterketa, euskararako.

Aurkezten dugun lanaren helburu nagusia, beraz, web dokumentuak filtratzeko tresna lortzea da, tresna horren bitartez dokumentuen kalitate minimo bat bermatu ahal izateko. Bide batez, hainbat dokumentu analizatu beharko direnez, estimatu nahi da mota ezberdinetako erroreen presentzia dokumentuetan. Web-etik eskuratuko diren dokumentuen artean denetarik egongo denez, esperokoa da euskalkiek eragina izatea lan honetan, baina aurretik ez dakigu eragina hori zenbatekoa izango den. Asko erabiltzen dira euskalkiak idatzizko testuetan? Posible izango al da euskalkiak erabiliz idatziko testuak identifikatzea? Azterketak informazio berria emango digu eta horren arabera, agian, etorkizuneko lanak planteatu ahal izango ditugu.

Lanaren metodologiari buruz esan dugu erreferentziazko lan gisa (Ringlstetter et al., 2006) izango dugula. Lan horretan, azterketa sakona egiten dute ingeleserako eta alemanerako hainbat corpus analizatuz. Alde batetik, gaiaren arabera antolatzen dituzte corpusak: gai orokorreko corpusak eta gai espezifikotako corpusak. Beste aldetik, dokumentuen jatorrizko formatuaren arabera antolatzen dituzte: HTML formatuko dokumentuen corpusak eta PDF formatuko dokumentuen corpusak. Era horretan, posible da analizatzea ezaugarri horiek eragina duten erroreen distribuzioan. Ildo beretik, guk ere hainbat corpus analizatuko ditugu euskaraz, baina ez ditugu lan honetarako espreski osatu corpus horiek. Web-eko dokumentuen bilaketa euskaraz egitea ez da sinplea eta Elhuyarrekin dugun lankidetza baliatuz, *Web as corpus* gaiaren inguruan egin dituzten proiektuetan zehar lortutako emaitzen azpimultzo bat hartuko dugu gure corpusak osatzeko. Hala, Elhuyarren egindako lanaren zati bat berrerabili dugu lan honen esperimentuak aurrera eramateko. Horrela erreferentziazko lanarekin parekoa den analisia egin ahal izango dugu. Erroreen tipologiari dagokionez, ingeleserako eta alemanerako lanean, errore tipografikoak, ezagutzakoak, OCR erroreak zein kodeketa erroreak analizatzen dituzte. Guk berriz, lehenengo hirurak analizatuko ditugu soilik, euskaraz ez baitugu azentu-markarik eta ez dugu alderdi horretatik arazorik espero. Antzekotasunak erreferentziazko lanarekin, beraz, nabariak dira baina, halaber, desberdintasun nabarmen bat dago eta hori erroreak detektatzeko erabili dugun metodoan datza. Ingelesez eta alemanez, "errore-hiztegiak" sortzen dituzte era semiautomatikoan gero horiekin erroreak detektatzeko. Euskararen kasuan, plantea-

mendu hori ezinezkoa da euskararen ezugarriak direla eta. Hizkuntza eranskaria denez, errore-zerrendak prestatzea ez da metodo eraginkorra, eta horren ordez, gure analisirako hizkuntza horietan erabili ohi den teknologia erabiliko dugu: egoera finituko teknologiaren bitartez sortutako transduktoreak.

Transduktoreak oinarrizko tresna dira hizkuntza eranskarietan analisia zein sorkuntza egiteko. Euskararen kasuan, guztiok ezaguna dugun *Xuxen* zuzentzaile ortografikoak teknologia hori du oinarri eta lan honetan, aplikazio horretarako landu ziren euskara estandarraren transduktorea eta ohiko erroreak zuzentzeko transduktorea berrerabiliko ditugu. Horietaz gain, lan honetarako espreski sortutako transduktore berriak erabiliko ditugu, ezagutza-erroreak ez diren beste errore motak identifikatu ahal izateko, hots, errore tipografikoak eta OCR erroreak. Transduktore berri horiek eraikitzeko, oso baliagarria izango da beste lanetatik jasotako esperientzia. Izan ere, 2009an zehar euskara estandarraren transduktorea kode irekiko tresna berri batera migratu zen: `foma` tresnara (Alegria et al., 2009), eta lan horretan ikasitakoa funtsezkoa izan da `foma` tresnarekin transduktore berriak sortu ahal izateko orain. Horretaz gain, OCR erroreen kasuan, oso baliagarria izan da HAP masterraren 2008-2009 ikasturtean egindako lana "Morfologia eta Sintaxia" ikastaroaren barruan. Ikastaro horren azkeneko lan gisa, OCR erroreak detektatzeko erregelak landu nituen hainbat adibide analizatuz, eta erregelak probatzeko `foma` tresna erabili nuen lehendabiziko aldiz. Lan hartan sortutako erregelak berrerabili ditugu lan honetan eta beste iturri batzuetatik lortutakoekin osatzen saiatu gara. Hala, alde batetik erreferentziazko lanean erabiltzen dituzten batzuk hartu ditugu, eta bestetik, berriro ere lankidetza baliatuz, Eleka enpresan erabiltzen dituzten patroi edo erregelak kontsultatu ahal izan ditugu gureak osatzeko.

Azkenik, aipatu nahi dugu lan honen barruan egin dugun analisiaren emaitza gisa, artikulu bat publikatu dugula 2010eko SEPLN kongresuan. Lanaren izenburua "*Errores ortográficos y de competencia en textos de la web en euskara*" da eta autoreak Iñaki Alegria, Izaskun Etxeberria eta Igor Leturia gara.

Sarrerako atal hau bukatzeko, txostenaren hurrengo atalen azalpen laburra eman nahi dugu. Bigarren atalean, lan aurrekarien berri emango dugu. Atal horretan, gure lanaren metodologia finkatzeko erreferentea izan den (Ringlstetter et al., 2006) lanaren nondik norakoak emango ditugu, batez ere. Gero, hirugarren atalean, egin dugun analisia aurrera eramateko erabili ditugun baliabideen ezaugarriak azalduko ditugu: esperimentuak egiteko erabili ditugun corpusen eta laginen ezaugarriak zein jatorria, eta erroreak detektatzeko eta identifikatzeko erabili ditugun tresnen ezaugarriak, argi utziz zer den berrerabiltzen duguna eta zer den guk garatutakoa lan honetarako. Horren ondoren, laugarren atalean, esperimentuen berri emango dugu agerian utziz analisia egiteko jarraitu dugun prozesuaren urratsak, eta gero, bosgarren atalean, lortutako emaitzak azalduko ditugu. Bukatzeko, seigarren atalean, lanaren ondorioak eta geratzen diren lerro irekiak aipatuko ditugu.

2 Aurrekariak

Proiektua aurkeztean aipatu dugu gure lana planteatzeko garaian jadanik publikatutako antzeko lana hartu dugula irizpidetzat. Ikerketa-lan hori ingeleserako eta alemanerako egiten dute (Ringlstetter et al., 2006) artikuluan. Beste antzeko lanak aurkitzen saiatu gara, eta kontuan hartu dugu, alde batetik, lan horren bibliografian ageri diren lanak, eta beste aldetik, bilaketa bat egin dugu ikusteko zein beste lanetatik erreferentziatzen den (Ringlstetter et al., 2006) lan hori. Artikulu horien artean begiratu ditugu batzuk (Reynaert, Tilburg Unibertsitatekoa, 2008; Tavosanis, Pisako Unibertsitatekoa, 2007), eta interesgarriak diren arren, haien gai nagusia ez dago estuki lotuta guk planteatzen dugun helburuarekin.

Hala, bigarren atal honetan (Ringlstetter et al., 2006) lanaren nondik norakoak laburbilduko ditugu horrela agerian uzteko gure lanerako hartuko ditugun erabakiak eta irizpideak, antzeko azterketa errepikatu nahi baitugu guk euskararako. Atal honen azpiatalak, beraz, erreferentziazko artikulu horren antolaketarekin bat dator. Laburbilpena gehiegi ez luzatzearren, oro har ingeleserako ematen dituzten emaitzak besterik ez ditugu azalduko hemen, nahiz eta aipamen bat edo beste egingo dugun alemaneko emaitzei, egokia deritzogunean.

Azkenik, kontuan hartu behar dira Elhuyar enpresako Igor Leturia et al. egin dituzten lanak *Web as Corpus* gaiaren inguruan, euskaraz. Xehetasunak aurrerago emango ditugun arren, aiptu nahi dugu hemen gure esperimentuetarako erabili ditugun corpusak Elhuyarreko lan horietan sortuak direla, eta funtsezkoak izan direla gure azterketa aurrera eramateko.

2.1 Sarrera

(Ringlstetter et al., 2006) lanaren helburua da errore ortografikoen distribuzioa mota ezberdineko web orrietan nolakoa den ikertzea. Ikerketa hori aurrera eramateko orri akastunak detektatzeko metodoak garatu behar dituzte eta baita errore ortografikoak markatzeko tresnak ere. Horrela, automatikoki web-etik hartutako testuekin corpusak edota datu-base linguistikoak osatzerako garaian, posible izango da tresna horien bitartez errore ortografikoak gutxitzea.

Gaur egun, corpus oso handiak eta heterogeneoak behar dituzten aplikazio ugari dago eta ohikoa da web-era jotzea corpus horiek osatzeko. Izan ere, web-a da hizkuntza gehienen testu-bilduma nagusiena eta, gainera, erraza da bertan bilaketak egitea Google APIaren edota antzeko produktuen bitartez. Bilaketa horien bitartez, ezaugarri edota eduki jakin batzuekin erlazionatutako orriak lortzen dira. Baina, nolakoak dira aurkitzen diren orri horiek? Denak balio al dute corpus bat osatzeko? Jakina da erroreak dituzten orriak topatuko direla askotan: errore tipografikoak, ezagutza-erroreak, OCR bitartez sortutako erroreak (testuen jatorria hori denean)... Esperimentuak oinarritzat hartuta, erreferentziazko lanean aztertu nahi dute lehendabizi:

- zein motakoak diren web orrietan topatzen diren erroreak

- nolakoa den erroreen maiztasuna, errore motaren arabera eta ehuneko zenbat orrik gainditzen duen τ errore-tasa[1] jakin bat
- ea aurreko balioak aldatzen diren testuen gaiaren arabera edo web orrien formatuaren arabera

Azterketa hori egin ostean, hasieran planteatutako problemari ekingo diote: ba al dago metodorik errore asko dituzten web orriak automatikoki detektatzeko? zein metodo izan daiteke egokia orrietan aurkitutako erroreak markatzeko? Beraz, azterketa egiteko eta aurreko hiru galderak erantzun ahal izateko, corpusak behar dituzte. Ingelesezko eta alemanezko corpusak osatzen dituzte web-ean aurkitutako hainbat dokumenturekin eta horiek analizatzen dituzte. Analisien emaitzak ematean, hots, errore-estatistikak ematean, honelako sailkapena egiten dute corpusetan:

- Gaiaren arabera:
  - corpus orokorrak, non orriak edo dokumentuak edozein gairen ingurukoak diren
  - corpus espezifikoak, non orriak gai zehatz bati buruzkoak diren
- Formatuaren arabera:
  - HTML lengoaian dauden orriekin osatutako corpusak
  - PDF formatuan dauden orriekin osatutako corpusak

Hiztegi berezia. Artikuluan aipatzen da hainbat orrik "hiztegi berezia" (*special vocabulary*) erabiltzen dutela eta multzo horren barruan kontsideratzen dituzte: izen "bereziak" (*special names*) erabiltzen dituzten orriak, argotak (*slang*) erabiltzen dituztenak, hizkuntza diakronikoa (*archaic language*) erabiltzen dutenak, atzerriko hizkuntzetako espresioak (*expressions from foreign languages*) erabiltzen dituztenak edota konputazio-zientziak eta programaziotik hartutako espresio bereziak erabiltzen dituzten orriak (*special expressions from computer science/programming*). Dena den, artikuluan ez da garbi geratzen norainoko filtroa egin duten arazo hau ebazteko. Izan ere, honela diote egileek artikuluaren sarreran, *Special Vocabulary* azpiatalean:

> "*Classification and detection of special vocabulary is outside the scope of the present article. Since sometimes a clear separation between special vocabulary and errors is difficult, we briefly come back to this problem in Section 5.4.*"

Gero, 5.4. atalean, berriz, honela dio:

> "[...] *One problem is caused by the fact that nonstandard vocabulary and errors do not represent disjoint categories.* [...] *demarcation issues are sometimes difficult to solve. The construction of special dictionaries for*

[1] Errore-tasa token akastunen portzentajea da.

> *slang, foreign language expressions, special names, and archaic word forms represent an important step for future work. Using these dictionaries in the filtering step of the construction of the error dictionaries, overproduction may probably be reduced in a significant way. Furthermore, these dictionaries should help to detect Web pages with nonstandard vocabulary of a particular type.*"

Beraz, badirudi ez dutela filtro hori kontuan hartu lan honetan eta etorkizunerako uzten dutela filtro hori garatzea "gainsorkuntza" (*overproduction*) jaisteko.

Erroreak. Hiztegi arruntari dagokionez, testuetan aurkitzen diren errore ortografikoak lau mota hauetakoak izan ohi dira:

- errore tipografikoak, hau da, testua tekleatzerakoan teklak nahasten direlako suertatzen diren erroreak;
- ezagutza-erroreak, hau da, hizkuntza guztiz menderatzen ez delako egiten diren erroreak;
- karaktereen kodeketa desegokiagatik sortutako erroreak;
- OCR erroreak;

Mota bakoitzeko erroreak estimatu nahi dituzte, eta horretarako, errore-hiztegiak eraikiko dituzte. Gero, hiztegi horiek erabiliko dituzte web orrietan edo dokumentuetan erroreak aurkitzeko.

Beraz, artikuluaren hurrengo ataletan azaltzen dute zein den errore-hiztegien motibazioa eta eraikuntza (artikuluaren 3. eta 4. atalak). Gero saiatzen dira estimatzen zenbatekoa den erroreen bilaketa horretan egindako huts kopurua: alde batetik, badira detektatu gabe geratzen diren hitz akastunak errore-hiztegietan ez daudelako, eta beste aldetik, badira hitz zuzenak direnak eta errore gisa tratatzen direnak errore-hiztegietan agertzen direlako (artikuluaren 5. atala). Hutsen estimazioa egin ondoren, analisiaren emaitzak aurkezten dituzte (6. eta 7. atalak). Emaitzak aurkeztu ondoren, saiatzen dira argitzen ea erlaziorik badagoen dokumentu baten errore tasaren eta generoaren artean (8. atala). Esperimentuetan agerian geratzen da errore-tasa oso aldakorra dela: badira dokumentu perfektuak, eta aldiz, badira gutxi batzuk onartezinak direnak. Beraz, horrek ematen die bidea filtro egokiak sortzeko horiekin detektatzeko eta ez onartzeko erabiltzaileak adierazitako errore-tasa jakin batetik gora egiten duten orriak (9. atala). Bukatzeko, bi aplikazio aurkezten dituzte garatutako filtroak erabilgarriak direla eta aplikazio horien eraginkortasunean lagun dezaketela erakusteko .

2.2 Corpusak

Arestian aipatu den moduan, erroreen analisia egiteko corpusak behar dira eta corpus horiek osatzeko web bilaketa-motorren bitartez topatutako orriak erabiltzen dituzte erreferentziazko lanean: Google eta AllTheWeb motorrak, hain zuzen. Erroreak aztertzeko

garaian, errore ortografikoen distribuzioa hizkuntzaren arabera aldatzen ote den ikusi nahi dute, eta horregatik, corpus guztiak ingeleserako eta alemanerako eraiki dituzte. Bi hizkuntza horien identifikazioaz, motorrak arduratzen dira. Gaiari dagokionez, aipatu da lehen alde batetik corpus orokor bat osatzen dutela, hau da, edozein gaiko web orriekin osatutakoa, eta bestalde, bost gai espezifiko dituzten beste hainbeste corpus: "Erdi Aroa", "Holokaustoa", "Arrainak", "Perretxikoak" eta "Neurologia". Ez da erraza izaten zehaztea noraino diren adierazgarriak web-etik lortutako corpusak, haien osaketa baldintza bat baino gehiagoren menpe baitago: egiten diren galderak; motorraren ordenatze-metodoa; aukeratze-estrategiaren detaileak... Alderdi horretatik begira, metodoa ahalik eta sinpleena izatea proposatzen dute: galdera eta aukeratze-estrategia sinpleak. Emaitzen orokortasuna edo adierazgarritasuna bermatzeko, corpusaren tamaina izan ohi dute kontuan eta, horregatik, orri asko jaso dituzte.

2.2.1 Web corpus orokorra

Lehendabizi alemanezko corpus orokorra osatu zuten. Orrien bilaketa egiteko, hasieran planteatu zuten metodoa esanahi gutxiko galderak egitea izan zen: adibidez, alemaneko hiru artikulu mugatuak dituen galdera egitea, hots, *der*, *die* eta *das*. Baina bai hori eta bai antzeko beste galderen bitartez lortutako emaitzak ez ziren gustukoak suertatu, izan ere, bide horretatik jasotako orri asko erakunde eta konpainia handietakoak ziren, testu gutxikoak (elementu grafiko asko) eta testu oso zainduak. Horregatik, lagin errealagoa lortzeko asmoz, beste metodo bat erabili zuten: alemaneko 10.000 hitz erabilienen bost hitzetako boskoteak sortu zituzten ausaz, eta horiekin bilaketak egin (ez dute adibiderik ematen, ordea). Gero, Googlen bitartez boskote bakoitzeko 10 orri jaso zituzten, horrela 1.000 orri lortu arte. Jasotako orriak ASCII kodera itzuli eta aurretiko errore-analisia egin zuten. Lehen analisi horren ondorioz, hainbat orri kendu behar izan zituzten hitz-klabe zerrenda handi bat gehituta zutela aurkitu zutelako, erroreak barne. Horrekin ziurtatzen omen da bilaketa-motorrak orri horiek aukeratuko dituela. Geratu zitzaizkien orriekin lehen corpusa osatu zuten (*German HTML (1)* corpusa). Bestalde, aukeratutako orriek emaitzetan eragina ote duten aztertu nahi dutenez, bigarren corpus bat osatu zuten prozesua berriro errepikatuz antzeko galderak eginez eta lortutako orri-multzoa aurrekoarekin disjuntua izanik (*German HTML (2)* corpusa).

Antzeko prozedurekin lortu zituzten, halaber, ingeleseko bi HTML corpus orokorrak (*English HTML (1)* eta *(2)*) eta PDF corpus orokorrak (*English PDF* eta *German PDF* corpusak). PDF formatua ASCIIra itzultzean, errore asko azaldu ziren kasu batzuetan, batez ere alemanaren kasuan, eta hala gertatzean orriak corpusetik kendu zituzten. Hori dela eta, corpusen tamaina ez da azkenean izan 1000 dokumentukoa, gutxiagokoa baizik. 2.1 taulan sei corpus horien ezaugarriak ageri dira, artean, web orri kopurua eta token arrunten[2] kopurua.

[2]Letra estandarrez soilik osatuak dira arruntatzat hartzen dituzten tokenak: digiturik gabe, beste karaktere arrarorik gabe.

General corpora	Web pages	Normal tokens	Size (MB)
English HTML (1)	829	7,900,337	157
English HTML (2)	929	7,152,783	188
German HTML (1)	618	9,525,484	189
German HTML (2)	857	11,539,035	284
English PDF	570	2,193,598	393
German PDF	603	1,528,914	240

Taula 2.1: Corpus "orokorren" ezaugarriak: web orri kopurua, token arrunten kopurua (letra estandarrez soilik osatutako tokenak) eta corpusaren tamaina megabytetan. (1) eta (2) zenbakiek lehen corpusa eta bigarren corpusa identifikatzen dituzte, hurrenez hurren.

2.2.2 Gai espezifikotako web corpusak

Lehen aipatu da web corpusen bost gai espezifikoak zeintzuk diren: "Erdi Aroa", "Holokaustoa", "Arrainak", "Perretxikoak" eta "Neurologia". Corpus horiek osatzeko bi metodo planteatzen dituzte: *simple crawl*[3] eta *refined crawl*, hurrenez hurren. 2.2 taulan ageri dira lortutako corpusen ezaugarriak, Ingeleserako.

Simple crawl metodoaren lan modua hauxe da: gaiko corpus txiki bat hartu eta mekanikoki aukeratu (ez dute gehiago zehazten) 25 gako-termino (*keyword*). Adibideak ematen dituzte ingelesezko bi corpusetarako: "Perretxikoak" eta "Neurologia". Gero, termino horiek eman AllTheWeb motorrari eman eta erantzuna jaso. Motor horrek orriak sailkatzen ditu, kontuan izanik 25 gako horietatik zenbat termino aurkitu diren orrian: termino horietako gehiago dituzten orriak hasieran daude. Metodo sinple horrekin lortzen diren orri guztiak ez dagozkio gaiari, baina ez dute ezer esaten egindakoari buruz: eskuz begiratu ote dituzte denak eta gaiari ez dagozkion orriak baztertu dituzte? Ez dute zehazten.

Refined crawl metodoarekin aztertu nahi dute ea metodo aurreratuago bat erabiliz gero, emaitzak aldatzen diren. Bigarren metodo hau ez diete gai guztiei aplikatzen. Honetan datza bilaketa: gaiko corpus etiketatu txiki bat hartu "hazi" gisa, eta hortik atera mekanikoki "agerkidetza terminologiko irekiak" (*terminological open compounds*), ingelesaren kasuan (Sornlertlamvanich eta Tanaka, 1996; Smadja eta McKeown, 1990), eta izen konposatuak (*compound nouns*), alemanaren kasuan. Termino horien adibide batzuk, neurologian, hauek lirateke: *amino group*, *action potential*, *defense mechanism* ... (beste gaietako adibideak ere ematen dituzte). Termino berezi horietako bakoitza Googleri eman eta erantzun bakoitzeko, gehienez, 30 orri hartzen dituzte. Ondoren, aukeratutako orri bakoitzak nolako antza duen hasierako corpus txikiarekin aztertzen dute (kosinuaren neurria aplikatzen dute). Orri gehienek pasatzen dute kontrol hori. Aurreko metodoa (Baroni eta Bernardini, 2004) metodoaren aldaera kontsidera daiteke eta helburua da corpusaren gai zehatzagoa lortzea.

[3] *Crawling* omen da nabigatzaileek web-ean egiten dutena: bilaketa metodikoa eta ordenatua.

Topic/Language	Web pages	Normal tokens	Size (MB)
Middle Ages E	710	5,069,796	172
Fish E (1)	510	10,090,682	266
Fish E (2)	940	547,407	22
Holocaust E	699	8,797,882	199
Mushrooms E (1)	676	7,876,067	197
Mushrooms E (2)	933	734,337	22
Neurology E (1)	624	8,765,899	197
Neurology E (2)	923	779,699	24

Taula 2.2: Ingeleseko (E) gai espezifikotako corpusen ezaugarriak. (1) eta (2) zenbakiek corpusak osatzeko erabili diren bi estrategiak identifikatzen dituzte: *"simple"* eta *"refined"*, hurrenez hurren.

2.3 Erroreen detekzioa

Bi metodo sinple erabili ohi dira, oro har testuetan, mota jakin bateko errore ortografikoak detektatzeko

1. Mota jakin horretako errore zerrenda adierazgarri bat sortu eta eskuz egiaztatu. Zerrenda horretan ageri den testuko token bakoitza, errore bat da. Metodo horri "beheko hurbilpena" deritzo (*lower approximation*).

2. Zuzentzaile bat edo hiztegi handi bat erabili hitz "anbiguoak" (erroreak izan daitezkeenak) detektatzeko. Horietako token bakoitza eskuz egiaztatu eta erabaki errorea den ala ez, eta baiezko kasuan, zein motakoa den zehaztu. Metodo horri "goiko hurbilpena" deritzo (*upper approximation*).

Baina aurreko metodo horiek erabiltzen badira corpus handiak analizatzeko, gabezi larriak antzematen dira. Lehendabiziko metodoarekin, erroreen portzentaje txikia detektatzen da eta zaila da estimatzea errore errealen kopurua. Bigarren metodoarekin, berriz, eskuz egiaztatu beharreko token kopurua handiegia da. Praktikan, gainera, token "anbiguo" asko token zuzenei dagozkie, izan ere, izen bereziak eta beste motako hiztegi ez estandarra ohikoa baita web orrietan. Horregatik, hirugarren aukera planteatzen dute (Ringlstetter et al., 2006) artikuluan. Aukera berri horretan aurreko bi metodoak sintetizatu nahi dituzte, hots, erdiko bidea hartu. Honela deskribatzen dute proposatutako metodo berria:

1. Hartu honako hiztegiak (ikus 2.3 Taula):

   (a) ingelesa, alemana, frantsesa eta gaztelania hiztegi estandarrak

   (b) hiztegi geografiko bat

   (c) izen nagusien hiztegia

   (d) laburduren eta akronimoen hiztegia

Dictionary	Number of entries
D(English)	315,300
D(German)	2,235,136
D(French)	85,895
D(Spanish)	69,634
D(Geos)	195,700
D(Names)	372,628
D(Abbreviations)	2,375

Taula 2.3: Hasierako hiztegien tamainak.

2. Hiztegi horietan oinarrituta, **errore-hiztegiak** sortu gero esperimentuetan erabiltzeko. Errore-hiztegiak lortzeko, lehendabizi eskuz sortzen dituzte "patroi" edo erregela batzuk errore mota bakoitzeko (*typing, spelling, OCR*) eta gero, patroi horiek aplikatzen dizkiete hiztegiko hitzei, horrela balizko errore-hiztegiak sortzeko.

3. Bukatzeko, lortutako balizko erroredun hitz horiek filtratu egin behar dira, posible baita erregelen bitartez sortzea hitz akastun gisa, hitz zuzenak direnak. Era honetan lortzen da errore-hiztegi bat errore mota bakoitzeko.

Hurrengo atalean, 2.4 atalean, errore-hiztegi bakoitzaren xehetasunak ematen dituzte, eta gero, esperimentuen bitartez ondorioztatzen dute errore-hiztegi horien bitartez posible dela ingelesez eta alemanez idatzitako web orrietako errore gehienak detektatzea. Dena den, prozedura ez da perfektua eta badira errore gisa tratatzen diren kasuak, benetako erroreak izan gabe. Portzentaje hori zenbatekoa den eta zergatik gertatzen den estimatzen saiatzen dira.

Ohar bat azaldu behar dugu puntu honetan. Artikulu osoa aztertu ondoren, ez zaigu guztiz garbi geratu zertarako erabiltzen dituzten beste hizkuntzetako hiztegiak, alegia gaztelania eta frantsesa. Gure ustez, hiztegi horiek filtro moduan erabiltzen dituzte, hots, behin patroiak aplikatu eta gero, sor daitezke erroredun gisa hitz zuzenak direnak, eta horiek detektatzeko hasierako hiztegi horietara jotzen dute, beraz, badirudi beste hizkuntzetako hitz zuzenak lortuz gero, horiek ere kentzen dituztela errore-hiztegietatik.

2.4 Errore-hiztegien sorkuntza

Errore-hiztegiak sortzeko, lehendabizi errore mota bakoitzaren patroiak edo erregelak zehaztu behar dira. Horiek lortzeko hainbat prozedura erabili dituzte: errore tipografikoen kasuan, analitikoki lortu dituzte, baina ezagutza-erroreen eta OCR erroreen kasuan, erregelak enpirikoki lortu dituzte. Dena den, badago erregela orokor bat kasu guztientzat: hitz motzak ez dira kontuan hartuko erroreak sortzeko. Muga 4ko luzera da, hots, 4 letra baino gehiagoko tokenak izango dira errore-hiztegietakoak. Autoreek diotenez, luzera hori edo txikiagoa duten tokenak akronimoak, izen bitxiak edo laburdurak izaten dira eta zaila da mekanikoki ezberdintzea zer den errore bat eta zer den hiztegi berezitu horretako hitza.

2.4.1 Errore tipografikoen errore-hiztegia

Errore tipografikoak (*typing errors*) hauek dira: transposizioak, ezabatzeak, ordezkapenak eta tartekatzeak.

- **Transposizioa**. Bi letren arteko transposizioa suertatzen da hitz batean, bi letren ordena aldatuta dagoenean: *lerta** adibidez, *letra* hitzaren ordez.
- **Ezabatzea**. Ezabatzea gertatzen da hitz batean, letraren bat falta denean (agian, tekla ez delako ongi sakatu): *leta** esaterako, *letra* hitzaren ordez.
- **Ordezkapena**. Ordezkapena gertatzen da letra zuzen baten ordez beste letraren bat ageri bada (gehienetan, teklatuaren aldamenekoren bat): *lertz** edo *lerts**, esaterako.
- **Tartekatzea**. Tartekatzea gertatzen da soberan dagoen letraren bat ageri bada (bi teklen artean, nahi gabe, besteren bat sakatu da): *letrra** adibidez.

Horiek dira literaturan jasotzen diren errore tipografiko gehienak (Kukich, 1992). Ez dituzte kontuan hartu errore "homologoak" deritzenak, hau da, suertatzen diren ordezteak ezker eta eskuin eskuak nahastu direlako. Eta beste xehetasun bat: arestian aipatutako patroiak edo "mutazio-erregelak" sortzean, letra estandarrak baino ez dituzte kontuan hartzen, hots, sinbolo bereziak eta digituak ez dira kontuan hartzen. Azkenik, hitzaren lehenengo letrari ez zaio errore-patroirik aplikatuko, ohikoa delako errore tipografikorik ez gertatzea lehenengo letran (Kukich, 1992).

Beraz, aurreko irizpideak zehaztu eta gero analisiari ekin zioten eta hauxe ikusi zuten: bai teklatu amerikarrean bai alemanean, l luzerako hitz batek, batez beste, $16l$ bariante lortzen ditu. Erregelak, beraz, oso produktiboak dira, eta ezin dira hitz guztiak erabili errore-hiztegia sortzeko. Hala, ingelesaren kasuan, maiztasun altueneko 100.000 hitzak soilik hartzen dituzte kontuan errore-hiztegi berria sortzeko. Horiei erregelak aplikatuta, 10,7 milioi *string* lortzen dituzte. Horietatik errepikatuak zein hitz zuzenak kenduz gero (hasierako hiztegietan dauden hitzak, alegia), 9,4 milioi *string* dituen errore-hiztegi bat lortzen dute ingeleserako: D_{err} (*English*, *typing*). Antzeko prozedura erabiliz lortzen dute alemanerako errore hiztegia (handiagoa, batez beste, hitzak luzeagoak direlako alemanez).

Beste errore motetan ez bezala, errore tipografikoen azalpena eta gero, artikuluan ez dituzte ematen errore horiek sortzeko erregelen adibide zehatzak.

2.4.2 Ezagutza-erroreen errore-hiztegia

Ingelesez idatzita dauden web orri asko bertakoak ez diren pertsonek idazten dituzte, hau da, hizkuntzaren konpetentzia edo ezagutza mugatua duten pertsonek. Hori dela eta, ezagutza-errore (*spelling errors*) asko suertatzen dira. Beraz, ezagutza-erroreak dira gaizki idatzita daudenak, hala idazten direla uste duelako idazleak.

Baina nola lortu mota horretako erroreak detektatzeko patroi esanguratsuenak? Nondik osatu errore-zerrenda bat? Kasu honetan, metodo enpirikoa jarraitzea erabaki zuten. Hasteko, ohiko 4 ezagutza-errore hartu zituzten kontuan: *verry**, *noticable**, *arguement**

Word	Google hits	Transformation	Misspell. variant	Google hits)
accommodate	5,800,000	mm → m	accomodate	559,000
category	109,000,000	teg → tag	catagory	525,000
definitely	10,800,000	itely → ately	definately	1,270,000
independent	25,700,000	dent → dant	independant	523,000
millennium	10,500,000	nn → n	millenium	2,540,000
occurrence	4,640,000	rr → r	occurence	279,000
receive	57,000,000	ie → ei	recieve	1,260,000
recommend	31,400,000	mm → m	recomend	707,000
separate	26,300,000	ara → era	seperate	1,340,000

Taula 2.4: Ezagutza-errore ohiko batzuk. Googlek emandako asmatze kopurua bai hitz zuzenarentzat bai bariantearentzat.

Deletion of doubled consonants	
cc→c	occasionally → ocasionally
nn→ n	drunkenness → drunkeness
Deletion of consecutive consonants	
mn→ m	column → colum
rh→ r	rhythm → rythm
Deletion of doubled vowels	
ee→ e	exceed → exced
uu→ u	vacuum → vacum
Deletion of silent vowels	
?ed →?d	maintained → maintaind

Taula 2.5: Ingeleseko ezagutza-erroreak sortzeko erregela batzuk eta horiekin lortzen diren adibideak. Taula honetan ez daude artikuluan ageri diren. Bakar batzuk kopiatu ditugu hemen.

eta *inteligence**. Horiekin bilaketak egin zituzten web-ean eta bakoitzarentzat 20 dokumentu hartu. Dokumentu horietan token arruntak hartu, maiztasunez ordenatu eta hiztegi estandarretan zeudenak kendu zituzten. Geratzen zirenetatik, balizko erroreak zirelakoan, eskuz aukeratu zituzten maiztasun altuena zutenak eta berriro errepikatu zuten prozedura bera errore-zerrenda berriarekin. Bilaketa eteten da maiztasun altuko hitz akastun berri gehiago aurkitzen ez direnean. Prozesu iteratibo horretan zehar, web orriak topatu zituzten, non ohiko erroreak zerrendatzen ziren (*common mispelled words*). Errore horiek ere errore-zerrendan sartu zituzten. 2.4 taulan adibide batzuk ageri dira. Bertan azaltzen dira hitz zuzenak eta haien bariante bat, bakoitzaren asmatze kopuruarekin Googlen.

Hitz akastunetan ageri diren errore gehienak erregelen bitartez sortu ahal dira eta hala, 95 erregela sortu zituzten. Erregela horien adibide batzuk 2.5 taulan ageri dira. 95 erregela horiek ingelesezko hiztegiari aplikatuz, 1.223.128 sarrerako errore-hiztegi berria sortu zuten. Filtroa pasa eta gero, azkenean 1.202.997 sarrerako hiztegia lortu zuten: D_{err} (*English, spell*).

Character substitutions	Character merges	Character splits
l → i	rn → m	m → rn
i → l	ri → n	n → ri
g → q	cl → d	ü → ii
o → p		
l → t		
v → y		
y → v		
o/e → c		
l → 1		

Taula 2.6: Ohiko OCR erroreak sortzeko erregelak.

Word	Transformation	Garbled result	Google hits
company	m → rn	cornpany	1220
from	m → rn	frorn	5310
government	rn → m	governrnent	705
more	m → rn	rnore	707
most	m → rn	rnost	1540
only	y → v	onlv	4080
said	d → cl	saicl	172
system	m → rn	systern	2060
time	m → rn	tirne	2090
will	ll → 11	wi11	3570

Taula 2.7: Maiztasun altueneko 1.000 hitzetako adibide batzuk. OCR errore tipiko bat aplikatu zaie, eta azkeneko zutabean, zenbat aldiz topatu duen Googlek erroredun bariantea ageri da.

2.4.3 OCR erroreen errore-hiztegia

Testu baten iruditik abiatuta testua sortzen duten aplikazioei OCR aplikazioak deritze: *Optical Character Recognition* programak. Prozesu horretan, ordea, programek erroreak egiten dituzte, eta horiei deritze OCR erroreak . Errore horiek sortzen dituzten erregelak bilatzeko hauxe egin zuten: 200 web orrietako OCR corpus txiki bat hartu eta bertan OCR aplikazioen eraginez sortutako erroreak aztertu. Analisi horretatik hainbat erregela atera zituzten (ikus 2.6 eta 2.7 taulak) eta, aurrekoetan bezala, hiztegiko sarrerei aplikatu zizkieten erregela horiek (sarrera guztiei, dirudienez). Bukatzeko, aurreko hiztegiekin bezala, emaitzak filtratu zituzten. Hala, beste errore-hiztegia sortu zuten, 1,5 milioi sarrera zituena: D_{err} (*English*, *ocr*).

OCR erroreen tratamendua alemanerako berezia da. Alemanez badira asko erabiltzen diren letrak eta askotan ezin direnak kodetu web orrietan: Ä, Ö . . . eta antzekoak. Tratamendu hori zehaztuta dago artikuluan, baina laburpen honetan ez dugu jaso.

2.5 Erroreen detekzioa: mugak

Corpusetako erroreak analizatu baino lehen, 2.4 atalean sortutako hiztegien bitartez, kontuan hartu behar dira analisi mota honek dituen mugak:

1. Errore-hiztegi bakoitzean ez daude mota horretako errore posible guztiak, beraz, ez dira detektatuko corpusetan dauden errore guztiak, gutxiago baizik. Horri azpisorkuntza (*underproduction*) deritzo. Azpisorkuntza suertatzen da, erabiltzen diren erregelak ez dituztelako errore posible guztiak sortzen (ezagutza- edo OCR erroreak) eta erregelak ez zaizkielako hiztegiko sarrera guztiei aplikatzen. Horrez gain, errore-dun hitzak sortzean, ez dira konbinazio eta aldaketa guztiak egiten, hau da, jatorrizko hitzak ez dira "gehiegi" aldatzen.

2. Gerta daiteke errore gisa sailkatzea token batzuk errore-hiztegian topatu direlako, eta benetan hitz zuzenak izatea: gainsorkuntza (*overproduction*). Hori gerta daiteke hiztegiak sortzean egiten den azkeneko filtroa ez delako guztiz zehatza.

Bi muga horiek zenbatekoak diren estimatu behar da, eta horretarako, corpus orokorra erabiliko dute. Lehendabizi, corpusa 4 sailetan banatzen dute dokumentu bakoitzaren errore-tasaren arabera, hots, kontuan izanik dokumentuaren 1000 tokenetik zenbat token dauden errore-hiztegietan. Hauek dira 4 sailak eta 2.8 taulan ageri da, beste datuen artean, sail bakoitzeko zenbat dokumentu dauden ingeleseko HTML corpus orokorrean:

- *Best*: 1.000 tokenetik, errore-hiztegietan aurkitzen direnen kopurua ≤ 1 da.
- *Good*: 1.000 tokenetik, errore-hiztegietan aurkitutakoen kopurua 1-5 tartean dago.
- *Bad*: 1.000 tokenetik, errore-hiztegietan aurkitutakoen kopurua 5-10 tartean dago.
- *Worst*: 1.000 tokenetik, errore-hiztegietan aurkitzen direnen kopurua 10 da.

2.5.1 Azpisorkuntzaren estimazioa

Lehen aipatu denez, errore guztiak ez daude errore-hiztegietan, baina zenbat falta dira? Azpisorkuntzaren balioa estimatzeko, 300 tokeneko fitxategitan banatzen dute corpusa eta fitxategi horiek ausaz hartuz, erroreen analisiari ekiten diote. Analisi horretan zuzentzaile ortografiko bat (*spell checker*) erabiltzen dute laguntza gisa eta baita haien errore-hiztegiak ere. Analisi horren helburua 1000 erroreko bilduma egitea da. Bilduma hori ahalik eta orekatuena izateko, eta errore guztiak sail beretik ez hartzeko (adibidez, *Worst* sailetik) hauxe egiten dute: fitxategi bakoitzetik gehienez 3 errore hartu. Bestalde, aipatzekoa da ez dituztela kontuan hartzen hiztegi bereziari lotutako erroreak, ez eta bi hitz elkartzen dituzten erroreak ere. Bildutako errore guztiak eskuz kontrolatzen dituzte, eta horrela 1.000 erroreko bilduma egiten dute. Gero, bildumako zenbat errore ageri diren errore-hiztegietan begiratzen dute. Emaitza 624 da, hots, %62.4 dira errore-hiztegian aurkitzen direnak. 2.8 taulan ageri da 1.000 errore horiek nola banatzen diren sailen artean, eta baita horietatik zenbat diren errore-hiztegietan topatutakoak ere. Aipagarria da dokumentu onenetan ageri diren erroreetatik erdia besterik ez dela ageri errore-hiztegietan.

Document class	Documents	Errors found	Entries of error dict.	%
Worst	24	248	166	66.93
Bad	39	194	131	67.53
Good	226	389	242	62.21
Best	540	169	85	50.29

Taula 2.8: Errore-hiztegien azpisorkuntza estimatzeko datuak. Ingeleseko HTML corpus orokorrari dagozkio datuak.

Document class	Best	Good	Bad	Worst
Hits	1000	1000	1000	1000
Percentage proper errors	72	86	89	95
Proper errors	722	856	894	952
Standard words	206	31	21	5
Personal names and geographic entities	23	35	24	27
Foreign language expressions	32	42	36	12
Archaic and literary word forms	9	28	1	1
Abbreviations	8	6	24	2

Taula 2.9: Errore-hiztegien gainsorkuntza estimatzeko datuak. Ingeleseko HTML corpus orokorrari dagozkio datuak.

2.5.2 Gainsorkuntzaren estimazioa

Lehenengo esperimentuetan antzeman zuten errore gisa sailkatutako hitz asko izen bereziei zegozkiela eta askotan, gainera, "asmatutako" izenak zirela. Horregatik, letra xehez hasten diren tokenak soilik analizatzea erabaki zuten.

Beraz, aurreko sail bakoitzeko 1.000 errore hartzen dituzte ausaz, errore-hiztegietan topatutakoak, eta gero, eskuz begiratzen dute 1.000 horietatik zenbat diren gainsorkuntzari dagozkionak, hau da, zenbat diren hitz zuzenak. Hori jakiteko kontuan hartzen dute testuingurua eta, zalantzazko kasuetan, behar diren kontsultak egiten dituzte (*Merriam-Webster Online* hiztegia erabiltzen dute kontsultetarako). Analisi hori egitearekin batera, erroretzat gaizki sailkatu den tokena zein motakoa den begiratzen dute: batzuetan, hasierako hiztegietan egon beharko luketen hitz zuzenak baina ez daudenak dira; beste batzuetan, toki izenak eta adierazpen geografikoak dira; badira atzerriko hizkuntzetako adierazpenak ere, eta abar luze bat. 2.9 taulan datu zehatzak ageri dira. Bertan ikus daiteke sail bakoitzeko zenbat errore dagoen kasu ezberdin horietan. Emaitzen arabera, ondorioa da benetako erroreen portzentajea handiagoa dela errore asko dituzten dokumentuetan. Bestalde, harrigarria da dokumentu onen sailean (*Best*) hitz estandarrek sortutako errore kopurua, 1.000tik 206 baitira horrelakoak. Autoreen justifikazioa da erabili duten ingeleseko hiztegiak ez dituela ingeles britainiarraren eta amerikarraren aldaera guztiak.

2.6 Erroreen distribuzioa HTML corpus orokorretan

Analisia egin eta gero, atal honetan HTML corpus orokorren emaitzak aurkezten dituzte. Alde batetik, errore guztiak batera hartuta lortutako emaitzak, eta beste aldetik, errore mota bakoitzari dagozkion emaitzak.

2.6.1 Errore guztiak batera

Errore guztiak batera hartuta sail bakoitzeko (*Best, Good, Bad, Worst*) batez besteko errore-tasa[4] ateratzen dute, eta baita corpus osoaren batez bestekoa ere. Datu zehatzak 2.10 taulan ageri dira. Emaitza horietan ikusten denez, ez dago diferentzia nabarmenik bi corpusen artean. Bestalde, interesgarria den beste balio bat ere ematen dute, analizatzen baitute batez besteko errore-tasaren balioa nola aldatzen den dokumentu "txarrenak" kentzen badira corpusetik: %10 txarrenak kenduta, eta %20 txarrenak kenduta. Hala, posible da ikustea nolako influentzia duten dokumentu horiek azkeneko batez bestekoetan. Oro har, ikusten da errore-tasa askoz baxuagoa dela (ia 4 aldiz baxuagoa), dokumentu txar horiek kontuan hartzen ez badira (ikus 2.10 taula). Zenbaki horiek, gero beste esperimentu batzuk baieztatuko duten efektu bat adierazten dute: corpusean dauden dokumentu gehienak kalitate onekoak dira erroreei dagokienez, hots, errore-tasa baxukoak dira, baina espektroaren "bukaeran" badira onartezinak diren dokumentu gutxi batzuk, oso errore-tasa altukoak. Horien eraginez, batez besteko errore-tasa asko igotzen da. Antzeko informazioa ikus dezakegu 2.1 irudian. Bertan azaltzen da errore-tasaren balioa dokumentuen zehar eta agerikoa da dokumentu gehienek errore-tasa baxua dutela.

Document class	Best	Good	Bad	Worst	Best 80%	Best 90%	All
E(1)	0,30	2,31	8,83	23,23	0,67	1,06	2,47
E(2)	0,27	2,19	6,77	21,61	0,68	1,03	2,24

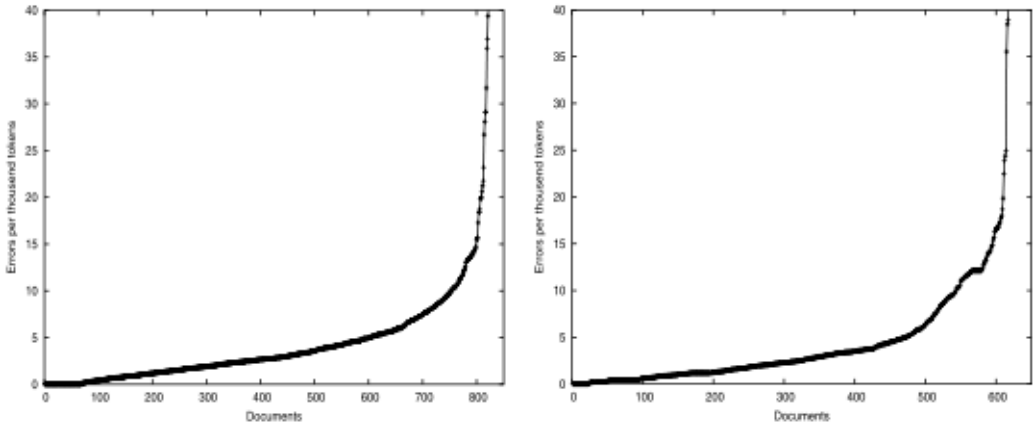
Taula 2.10: Dokumentu sail bakoitzari dagokion errore-tasa kontuan izanik errore ortografiko guztiak. Emaitzak ingelesezko bi HTML corpus orokorrekoak dira.

2.6.2 Errore mota bakoitza

Atal honetan, errore mota bakoitzaren ezaugarriak aztertzen dituzte, hau da, errore tipografikoen, ezagutza-erroreen eta OCR erroreen ezaugarriak: zenbat errore dagoen, zein den maiztasun altueneko mota, eta abar.

Errore tipografikoak. Gehien suertatzen diren erroreak dira: ingelesaren kasuan, 2,47 erroretik, 2,31 dira mota honetakoak (%93,5). Alemanez, berriz, portzentajea baxuagoa da. Horren arrazoia da alemanez badirela beste errore mota bereziak eta garrantzitsuak (adibide horiek ez ditugu laburpenean jaso).

[4]Gogoratu: errore-tasak adierazten du 10.00 tokenetik zenbat aurkitzen diren errore-hiztegietan.



Irudia 2.1: Errore-tasaren balioa HTML bi corpus orokorretan: ezkerrean ingelesekoa, eskuinean, alemanekoa.

Ezagutza-erroreak. Mota hauetako errore-tasa baxua da: 0,39 ingelesaren kasuan. Badakite ez dituztela detektatzen benetan dauden errore guztiak (ondorio hori agerian geratu da 2.5.1 atalean) eta horregatik diote, euren ustez, benetako ezagutza-erroreak gehiago direla baina ez dituztela detektatzen.

OCR erroreak. Emaitzek erakusten dute corpus orokorretan ia ez dagoela OCR errorerik: ingeleseko corpusetan OCR errore-tasa 0,06 da. Autoreen ustez, arrazoi nagusia da HTML corpus orokorretan oso dokumentu gutxi direla eskaneatutako testutik datozenak. Esperokoa litzateke OCR errore kopurua altuagoa izatea analizatutako corpusak eskaneatutako dokumentuekin osatuko balira.

2.7 Erroreen distribuzioa corpus berezietan

Atal honetan laburbiltzen dute zenbatekoak diren errore-tasak bai PDF corpusetan, bai gai espezifikoko corpusetan, eta emaitza horiek HTML corpus orokorretan lortutakoekin konparatzen dituzte.

2.7.1 Erroreak PDF corpus orokorrean

Errore guztiak batera hartuta, analisi honetan PDF corpusaren errore-tasa eta HTML corpusetakoak konparatzen dituzte. Datuen arabera, ondorioa da kalitate hobea dutela PDF dokumentuek HTML dokumentuek baino: errore-tasa 2,47 da HTML corpusean, eta 1,38 PDF corpusean. Tendentzia bera ikusten da alemaneko corpusetan.

2.7.2 Erroreak corpus espezifikoetan

Corpus espezifiko guztien errore-tasa kalkulatzen dute eta lortutako balioak konparatzen dituzte. Emaitzen arabera ondorioztatzen da errore-tasa, ia beti, altuagoa dela corpus

espezifikoetan corpus orokorretan baino. Salbuespen bakarra "Neurologiako" corpusean gertatzen da: honetan errore-tasa 2,05 da, hau da, HTML corpus orokorraren 2,47 baino baxuagoa. Gainontzeko kasuetan tasa beti altuagoa da: 2,97 ("Holokaustoa"), 4,10 ("Perretxikoak"), 5,32 ("Erdi Aroa") eta 7,08 ("Arrainak"). Autoreen iritziz, corpusa gai zehatz bat jarraitu gabe osatzen denean, jasotzen diren orriak "profesionalak" eta kontuz editatuak dira. Aldiz, gai espezifikoko orriek ez dute ia publizitaterik eta ez dira hain "profesionalak". Horri gaineratu behar zaio, halaber, gaia *hobby* motakoa baldin badela (esaterako, "Arrainak" gaia) jasotako orriak gai zientifikoagoko orriak baino errore gehiago dituztela.

2.7.3 Errore-tasak, bilaketa estrategiaren arabera (*crawling*)

Gogoratu behar da hiru gai espezifikotan ("Arrainak", "Perretxikoak" eta "Neurologia") bi corpus osatu dituztela: bata, bilaketa-estrategia sinplea erabiliz osatutakoa, eta, bestea, estrategia landuagoa erabiliz osatutakoa (ikus 2.2.2 atala). Bilaketa-estrategia landuagoa erabiliz osatutako corpusetan errore-tasa txikiagoa da beti. Kasu batzuetan gainera, oso nabaria da diferentzia hori, ia erdira jaisten baita errore-tasa. Aipamen berezia egiten dute atal honetan dokumentu txarrenen eraginari batez bestekoan: estrategia landuaz osatutako corpusetan %80 dokumentu onenak soilik hartuz gero, errore-tasak benetan txikiak dira: 0,35, 0,32 eta 0,25 hiru corpusetan, hurrenez hurren.

2.7.4 Laburbilpena

Aurreko atalen laburpen gisa, gai orokorreko corpusetan PDF corpusek errore-tasa txikiagoa dute HTML corpusek baino. Bestalde, gaiari dagokionez, corpus espezifikoen errore-tasa altuagoa da corpus orokorren errore-tasa baino, baldin eta corpusaren gaia zientifikoa ez bada. Azkenik, bilaketa estrategiak zerikusia du lortzen den corpusaren errore-tasarekin: ingelesaren kasuan, corpus hobeak lortzen dira bilaketarako estrategia landuak erabiliz gero.

2.8 Errore-tasa eta dokumentuen generoa

Zer korrelazio dagoen errore-tasa eta dokumentuaren "generoaren" (*genre*) artean aztertu nahi dute atal honetan. Horretarako, lehendabizi genero edo klase horiek definitzen dituzte:

1. *Prof* klasea: testu profesionalak dituzten orriak sartzen dira bertan, hala nola, erakundeak, enpresak eta administrazioa, testu zientifikoak eta literatura profesionala.

2. *Priv* klasea: orri pribatuak eta ikuspuntu pertsonal batetik idatzitakoak (*I* izenordaina omen da klabea orri horietan). Gerta daiteke mota horretako testuak erakunde baten orrian egotea, baina kasu horietan ere, *Priv* gisa sailkatzen dituzte.

3. *Chat* klasea: bertan daude *chat*-ak eta iritzien bildumak jasotzen dituzten tokiak: *guest books*, *mailing lists* eta abar.

4. *Junk* klasea: klase honetako orrietan hizkuntza ez da "garbia", adibidez, bertan sartzen dira hitz akastunen zerrendak dituzten orriak, atzerriko hizkuntzetako espresioen zerrendak dituztenak, programa-kode zatiak dituztenak, aspaldiko hizkuntza erabiltzen dutenak, eta abar.

5. *Other* klasea: bertan sartzen dituzte aurreko lau klaseetan sartzen ez diren dokumentuak. Dena den, beti saiatzen dira aurreko klaseetako batean sailkatzen. Horregatik, azken klase honetan ia ez dago dokumenturik.

Generoaren analisiarekin egiten dituzten esperimentuak ingeleserako bakarrik egiten dituzte, eta ez corpus guztiekin. Analisia zazpi corpusetan egiten dute: HTML corpus orokor batean eta bi corpus dituzten gai espezifikoko corpusetan, hots, "Arrainak", "Perretxikoak" eta "Neurologia".

2.8.1 Generoen distribuzioa eta corpusaren kalitatea

Corpus bakoitzean eskuz sailkatzen dituzte orriak generoetan, baina ez guztiak. *Worst* eta *Bad* ataletako orri guztiak sailkatzen dituzte (gogoratu dokumentu gutxien dituzten sailak direla horiek), eta *Good* eta *Best* ataletako 100 orri, ausaz aukeratuak. 2.11 taulan HTML corpusaren emaitzak ageri dira. Bertan garbi ikusten da nola orri "txarretako" sailetako zati handi bat, hau da, *Worst* eta *Bad* sailetako zati handi bat, *Chat* eta *Junk* generoei dagokien. Tendentzia hori corpus guztietan errepikatzen da.

Bestalde, gai espezifikoko corpusetan *Prof* generoa altuagoa da nahiz eta orrien kalitatea altua ez izan. Hori oso ongi ikusten da 2.12 taulan jasotzen diren datuetan.

English HTML (1)	Worst (%)	Bad (%)	Good (%)	Best (%)
Chat	42.31	56.41	24.00	1.00
Junk	38.46	5.13	1.00	0.00
Priv	3.85	10.26	14.00	9.00
Prof	15.38	28.20	61.00	90.0
Other	0.00	0.00	0.00	0.00

Taula 2.11: Generoen distribuzioa lehendabiziko HTML corpus orokorrean. Distribuzioa ageri da kalitatearen araberako egin diren lau sailetan.

2.8.2 Generoen distribuzioa bilaketa-estrategiaren arabera

Analisi honetatik ondorioztatzen dute generoari dagokionez diferentzia handia dagoela corpusa osatzeko erabili den metodoaren arabera. Bilaketa-estrategia landua erabiltzen bada, *Chat* eta *Junk* generotako orriak gutxiago dira eta, aldiz, gehiago dira *Prof* generokoak. 2.12 taulan datu zehatzak ageri dira eta bertan ikus daiteke diferentzia zenbatekoa den.

Crawls	Fish(1)	Fish(2)	Mushr.(1)	Mushr.(2)	Neur.(1)	Neur.(2)
Chat	13.86	2.69	8.63	3.52	3.87	2.87
Junk	9.10	0.88	5.40	3.15	2.97	0.11
Priv	8.79	16.13	12.70	11.96	7.49	2.44
Prof	66.03	80.30	73.27	80.01	82.83	94.58
Other	2.22	0.00	0.00	1.36	2.84	0.00

Taula 2.12: Generoaren distribuzioa bi bilaketa-estrategien arabera (sinplea (1) eta landua (2)) corpusetan. Estrategia landua lagungarria da *Chat* eta *Junk* motako dokumentuak ez hartzeko eta, bide batez, *Prof* motakoak erakartzeko.

2.8.3 Errore-tasa generoaren arabera

Azkenik, genero bakoitzaren errore-tasa estimatzen dute ikusteko erlaziorik badagoen bien artean, alegia, generoaren eta errore-tasaren artean. Estimazio bat baino ez da, ez baitira kalitate oneko dokumentu guztiak (*Best* eta *Good*) generotan sailkatu, 100 dokumentu baizik (ikus 2.8.1). Beraz, sailkatutako 100 horiekin egin behar da kalkulua. Horregatik emaitza estimazio bat baino ez da. Emaitzak adierazten du errore-tasa generoaren adierazgarri izan daitekeela. Izan ere, errore-tasa altua duten dokumentuak *Junk* eta *Chat* generokoak izan ohi dira, eta *Prof* generokoek, berriz, oso errore-tasa baxua izaten dute. 2.13 taulan datu zehatzak ageri dira.

Crawls	Eng. HTML	Fish (1)	Fish (2)	Mushr.(1)	Mushr.(2)	Neur.(1)	Neur.(2)
Chat	6.90	13.05	14.29	10.71	6.27	4.94	11.22
Junk	27.31	23.61	59.05	12.37	16.00	4.59	3.15
Priv	2.82	7.85	3.16	3.34	3.37	3.79	5.89
Prof	1.26	3.68	2.04	2.94	1.20	1.67	1.31

Taula 2.13: Errore-tasaren estimazioa genero bakoitzeko zazpi corpusetan.

2.9 Filtratze-metodoak

Atal honetan zera aztertu nahi dute: nola erabil ditzaketen sortutako errore-hiztegiak kalitatezko corpus bat osatzeko web-etik hartutako dokumentuekin. Helburua da detektatzea eta corpusetik ezabatzea errore-tasa altua duten dokumentuak. Errore-tasa "altua" noiz den definitu egin beharko litzateke corpusa osatzeko unean, eta hori, oro har, corpusaren erabileraren mende egongo da.

Definizioa. Filtro batek $F = \langle D, \rho \rangle$ bi osagai ditu: errore hiztegia, D, eta filtroaren ataria, ρ. Filtroak ez du onartuko T dokumentu bat (web orri bat) baldin eta dokumentu horren errore-tasa D hiztegirekiko, ρ baino handiagoa bada.

Egiten dituzten esperimentutan oinarrituta, hauxe ondorioztatzen dute: ρ ataria zorrotza bada, antzeko emaitzak lortzen dira errore-hiztegi "txikiak" zein osoak erabiliz. Beraz,

errore-hiztegiak, nahiz eta osoak ez izan (ikus 2.5.1 atala) baliagarriak dira filtro gisa erabiltzeko. Ikus dezagun, bada, nolako esperimentuak egiten dituzten emaitza hori ondorioztatzeko.

2.9.1 Erroreen maiztasuna

Lehendabiziko galdera da ea filtroek funtzionatuko duten, hots, ea nahikoa izango diren sortutako errore-hiztegiak osoak ez direla jakinik (ikus 2.5.1 atala). Erantzuna baiezkoa da: web-eko errore ortografikoen maiztasunak Zipf-en antzeko distribuzioa jarraitzen du, eta horren ondorioz, erroredun sarrera gutxi batzuk errore guztien zati handi bat hartzen dute barne. Hori horrela dela erakusteko, grafiko bat ematen dute non ikusten den erroreen maiztasuna (gutxienez 2koa) 1,4 terabyteko corpus batean[5]. Bertan antzeman daiteke da errore batzuk askotan gertatzen direla (100 - 100.000 aldiz), baina errore guztien erdia baino gehiago 10 aldiz baino gutxiago ageri direla.

2.9.2 Filtroaren aplikazioa

Demagun web orri multzo handi bat, C, filtratu nahi dela erabiltzaileak ezarri duen μ atari jakin batekin. Atari hori gainditzen ez duten dokumentuak "egokiak" dira eta gainontzekoak, berriz, "desegokiak". Erroreak detektatzeko errore-hiztegiak erabiltzen dira.

Filtro ezberdinak definitu eta ebaluatu nahi dituzte, ez baitute uste beti beharrezkoa denik errore-hiztegi osoak erabiltzea. Hala, filtro baten egokitasuna neurtzeko haren doitasuna eta estaldura neurtzen dituzte. Doitasunak adierazten du filtroak egokitzat sailkatzen dituen dokumentuen artean zenbat diren benetan egokiak; estaldurak, berriz, adierazten du benetan egokiak direnen dokumentuetatik zenbat sailkatu dituen egoki gisa filtroak. μ atariaren balioa ere kontuan hartu behar da eta hainbat baliorekin egingo dira probak: 10, 5 eta 1 balioekin, hain zuzen.

2.9.3 Filtroen eraikuntza automatikoa

Filtro-hierarkia bat sortzen dute:

$$F_1 = \langle D_1, \rho_1 \rangle, F_2 = \langle D_2, \rho_2 \rangle, F_3 = \langle D_3, \rho_3 \rangle, \ldots$$

Hierarkiaren arabera, F_k filtroak emaitza hobeak lortzen ditu k zenbat eta altuagoa izan, baina D_k hiztegia ere handitzen joango da hierarkian.

Filtroak definitzeko, beharrezkoa dute errore-zerrenda bat maiztasunaren arabera ordenatuta eta zerrenda hori lortzeko 2.9.1 atalean aipatutako 1,4 TByteko corpusa erabiltzen dute. Errore-zerrenda osatzeko, corpusean gutxienez bi aldiz azaltzen diren erroreak soilik hartzen dira: 1.175.894 errore, guztira.

Bestalde, filtroen egokitasuna neurtzeko, beharrezkoa da filtratu nahi den corpusa bi zatitan banatzea: zati bat filtroa bera definitzeko erabiltzen da eta bestea, berriz, filtroa testeatzeko, hots, haren egokitasuna neurtzeko. Corpusaren banaketa hori ausaz egiten

[5]Corpus hori 1999an osatu zen web dokumentuekin

dute, eta 384 dokumentu uzten dituzte filtroak definitzeko eta 407 testeatzeko. 384 dokumentu horietan guztietan, errore-zerrendako 5 errore ezberdin ageri dira gutxienez (hasieran 427 dokumentu ziren, baina baldintza hau bete araztean, 43 dokumentu ezabatu zituzten).

Filtroen definizioa. Filtro bakoitzaren hiztegia, D_k, errore-zerrendako hasierako S segmentua da eta ezaugarri hau dauka: entrenatzeko azpicorpusaren dokumentu desegoki guztiek dituzte k sarrera gutxienez hiztegi horretan. Filtro bakoitzaren ρ_k atariak, berriz, adierazten du, batez beste, entrenatzeko azpicorpusaren dokumentu desegokietan, 1000 tokenetik zenbat ageri diren gutxienez D_k hiztegian (sarrera horiek ez dute zertan ezberdinak izan). Horrela eraikitako filtroekin lortzen den doitasuna %100 da entrenatzeko azpicorpusean, hau da, filtroak adierazten badu dokumentu bat egokia dela, hala da benetan. Ikusi behar da zenbateko doitasuna lortzen den gero testerako corpusarekin.

2.9.4 Filtroen aplikazioa ingeleseko HTML corpus orokorrean

Atal honetan filtro ezberdin batzuk ebaluatu nahi dituzte HTML corpusa filtratzeko garaian. Filtroak eraikitzeko, μ ataria ezarri behar da, eta hiru balio ezberdinekin egin dituzte probak: 10, 5 eta 1. 2.14 taulan lortutako emaitzak ageri dira. Ondorio nagusiak hauek dira:

- F_3 filtroarekin lortzen diren balioak doitasunerako eta estaldurarako nahiko onak dira, beraz, filtroaren kalitatearen eta tamainaren arteko oreka ona lortzen du filtro horrek.

- Hiru kasuetan, F_3 filtroak gaizki sailkatzen dituen dokumentuen errore-tasa, μ atari baliotik oso gertu dago. Gaizki sailkatutakoak dira egokiak bailira sailkatu direnak hala izan gabe (ataria gainditzen dute).

- Filtroaren ataria, ρ, eta hiztegiaren tamaina konpentsatu egiten dira: hiztegia zenbat eta txikiagoa izan, orduan eta filtroaren ataria baxuagoa da. Balioei dagokienez, berriz, ikusten da ez dela ia doitasunarik galtzen, eta estaldura oso gutxi galtzen dela.

2.10 Aplikazio-adibideak

Atal honetan, bi adibideren bidez erakutsi nahi dute, aurreko ataletan azaldutako filtratze metodoa eta eraikitako errore-hiztegiak erabilgarriak direla datu linguistikoak web orrietatik lortu ahal izateko direnean. Filtroak lagungarriak dira onartezinak diren dokumentuak kentzeko. Bestalde, geratzen diren dokumentuetan posible da markatzea errore gisa errekonozitzen diren tokenak, hau da, errore-hiztegian ageri direnak. Aplikazioaren arabera erabaki beharko da zer egin gero markatutako tokenekin.

	$\|D_k\|$	ρ_k	P^{Train}	R^{Train}	P^{Test}	R^{Test}
$\mu = 10$						
k=1	12,217	0.91	100.00	85.42	99.67	80.00
k=2	21,037	1.83	100.00	89.79	99.69	84.73
k=3	46,111	2.19	100.00	91.83	99.40	87.63
k=4	110,201	4.63	100.00	93.87	99.71	91.31
k=5	291,309	5.62	100.00	93.00	99.70	89.21
$\mu = 5$						
k=1	34,322	1.23	100.00	87.42	99.34	86.00
k=2	47,747	2.19	100.00	95.70	98.50	94.00
k=3	90,160	3.53	100.00	98.77	97.47	97.42
k=4	110,201	3.71	100.00	98.77	97.70	97.42
k=5	291,309	4.83	100.00	100.00	96.15	100.00
$\mu = 1$						
k=1	37,994	0.13	100.00	51.15	93.43	55.89
k=2	169,507	0.49	100.00	78.35	96.75	78.16
k=3	279,543	0.63	100.00	86.14	97.02	85.58
k=4	299,397	0.67	100.00	90.90	97.10	87.77
k=5	580,330	0.89	100.00	97.40	96.06	96.91

Taula 2.14: Hierarkiako lehen 5 filtroen ebaluaketa ingeleseko HTML corpus orokorrarekin. Probak egin dira erabiltzaileak ezarritako μ atariaren hiru balio ezberdinetarako: 10, 5 eta 1.

2.10.1 Testuen zuzenketa *crawled* hiztegiekin

Eskuz eraikitako hiztegiek estaldura txikia omen dute testu eta corpus berriekin lan egiteko erabiltzen direnean. Hala ondorioztatzen da (Kukich, 1992) lanean deskribatzen duen esperimentu batean:

> "*Nearly two thirds (61%) of the words in the Merriam-Webster Seventh Collegiate Dictionary did not appear in an eight million word corpus of New York Times news wire test, and, conversely, almost two-thirds (64%) of the words in the text were not in the dictionary.*"

Hori dela eta, plantea daiteke hiztegi berri bat eraikitzea web-etik hartutako orriek dituzten hitzekin: hori da *crawled* hiztegia. Horretarako, web orri egokiak eta zuzenak behar dira. Zoritxarrez, baina, beti izango dira errore ortografikoak hartzen diren orrietan. Aurreko ataletan landu diren filtroen bitartez, posible da hiztegi berriak izango dituen erroreen kopurua gutxitzea. Bi aukera daude horretarako: zuzenean hitz horiek kentzea edo hitzak markatzea gero eskuz erabakitzeko zer egin errore horiekin. Jarraian esperimentu bat egiten dute testu-zuzenketaren arloan, garatutako teknikak bi puntu hauetan izan dezakeen garrantzia erakusteko:

- *Crawled* hiztegi berriaren kalitatea hobetzen du sarrera akastunak ekiditen direlako.

- *Crawled* hiztegi berriaren bitartez egiten den testu-zuzenketaren zehaztasuna hobea da baldin eta maila altuko zuzenketa-sistema (Strohmaier et al., 2003a,b) erabiltzen bada

Zuzenketa-estrategia. Sarrera-testuko token bat zuzentzeko, hiztegian bilatzen dira harekin antza handia duten sarrerak (hiztegian token bera topatzen bada antzekotasun optimoa daukagu), horiek izango baitira zuzenketarako hautagaiak. Hautagai bakoitzari puntuazio bat ematen zaio kontuan izanik: 1) antzekotasuna sarrerako tokenarekiko eta 2) hautagaien maiztasuna corpusean. Puntuazioak τ atari jakin bat gainditzen badu, sarrerako tokena ordezten da, eta bestela bere horretan uzten da. Puntuazioa ematean, oreka behar da antzekotasunaren eta maiztasunaren artean. Hori lortzeko, entrenamendua behar da. Atari balioaren optimizazioa ere entrenamenduaren bitartez lortzen da, eta balio horrek ziurtatzen du zuzenketarako erabiltzen den hautagaia egokiena izatea. Esperimentuan "Perretxikoen" corpusa erabili zuten entrenatzeko eta "Arrainen" corpusa ebaluaziorako.

Zuzenduko den sarrerako testu erroreduna. "Arrainen" domeinuan bilaketa egin eta 10 testu aukeratu ziren web-etik. Testu horiek erroreak zituzten tartean, Google-kin bilaketa token erroredunekin egin baitzen. Eskuz analizatu ostean, 10 testu horiek bakar batean elkartu eta zuzenketa sistemari eman zioten haren funtzionamendua ebaluatu ahal izateko. Ebaluazioa egiteko, noski, testuaren eskuzko zuzenketa eta sistemak egindakoa konparatzen dira. Testuan, 17.697 token daude eta 418 dira akastunak (% 2,36).

Errore-hiztegien erabilera zuzenketan. Abiapuntu gisa, "Arrainen" corpusean ageri diren tokenekin hiztegi bat osatzen dute: D_{crawl} da eta 505.562 sarrera ditu. Bigarren hiztegi bat sortzen dute gero, corpusa filtratu ondoren ($\mu = 2$ atariarekin), horretarako ingeleseko errore-hiztegia erabiliz. Lortutako hiztegia D^{+F}_{crawl} da eta 291.065 sarrera ditu. Azkenik, hirugarren hiztegia sortzen dute, D^{+F}_{crawl} hiztegitik errore-hiztegian dauden sarrerak ezabatuz. Hau D^{+F+ED}_{crawl} hiztegia da eta 269.079 sarrera ditu.

Aurreko hiztegiak zabaltzeko aukera dago filtratutako web orri gehiago erabiliz, baina ez dute horrelakorik egin. Dena den, aurreikusten dute aukera horrek emaitzak hobetuko lituzkeela.

Emaitzak ebaluatzen. Atal honetan, hiru hiztegietan konparatzen dituzte: a) lexikoaren estaldura (zuzendutako testuko zenbat token ageri diren hiztegian) eta b) zuzenketaren zehaztasuna (token zuzenen kopurua, zuzenketa automatikoa eta gero). Emaitzak 2.15 taulan ageri dira eta hurrengo puntuetan komentatzen ditugu.

Dictionary	Entries	Coverage (%)	Accuracy (%)	±(%)	False friends
D_{crawl}	505,652	99.08	98.45	0.81	262
D^{+F}_{crawl}	291,065	98.77	98.61	0.97	92
D^{+F+ED}_{crawl}	269,079	98.75	98.74	1.10	49

Taula 2.15: Kalitatearen neurketa osatutako hiztegietan. D_{crawl} lortu da web orriak filtratu gabe, D^{+F}_{crawl}, berriz, web orriak filtratuekin. Azkeneko hiztegian, D^{+F+ED}_{crawl}, orriak filtratzeaz gain, hiztegitik ezabatu dira oraindik ageri ziren errore-hiztegiko sarrerak.

Errore-hiztegiaren azpisorkuntza eta gainsorkuntza. Lehen esan den moduan, eskuz egin den zuzenketan 418 errore detektatu dira. Horietatik 254 (%60,77) ingeleseko errore hiztegian ageri dira. Azpisorkuntzaren balio hau eta 2.5.1 atalean lortutakoa nahiko antzekoak dira. Gainsorkuntzari dagokionez, 7 dira sarrerako testuaren token zuzenak, errore-hiztegian ageri direnak.

Errore-hiztegien eta filtroen erabileraren analisia. Hiztegiak osatzen direnean web orrietako hitzekin, badira hainbat errore detektatzen ez direnak eta hiztegian geratzen direnak. Horiexek dira gero zuzenketa-prozesuan akatsak sortzen dituztenak. 2.15 taulan ageri diren datuek agerian uzten dute filtroek eta errore-hiztegiak oso lagungarriak direla hiztegi berriren errore kopuru hori minimizatzeko: 262 errore (*false friends*) daude D_{crawl} hiztegian, eta 49 soilik (D_{crawl}^{+F+ED}) hiztegian, hots, 5 aldiz gutxiago. Bestalde, filtratze prozesua bi urratsetan egiteak, hots, lehendabizi corpuseko orriak filtratzea, eta gero orri horietatik lortutako hiztegia filtratzea, emaitza hobetzen du. Ikusten denez, D_{crawl} hasierako hiztegia errore-hiztegiaren bitartez filtratzen bada soilik (urrats bat), hasierako 262 erroretatik 105 geratzen dira. Aldiz, filtratze prozesu hori bi urratsetan eginez gero, 49 errore soilik geratzen dira. Beraz, emaitza hobea lortzen da bi urratsak eginez.

2.10.2 Itzulpen-datuak lortu Corpus paraleloetarako

Corpus paraleloak aukera ona dira hiztegi elebidunak automatikoki sortzeko, baina gutxi dira corpus paralelo handiak eta, gainera, gai baten inguruan espezializatuak izaten dira. Hori dela eta, web-ak aukera ona eskaintzen du testu paraleloak lortzeko (Resnik eta Smith, 2003). Bai hiztegi elebidunak sortzeko, bai esaldi paraleloen erabilera sustatzeko, zuzentasuna bermatzea nahitaezkoa da. Esperimentu honetan erakutsi nahi dute errore-hiztegien erabilera lagungarria dela edozein corpus paralelotan erroreen kopurua gutxitzeko.

Esperimentuan *Europarl* corpusa erabiltzen dute: Europako parlamentuaren 1996-2001eko aktak jasotzen ditu corpus horrek, 11 hizkuntza ofizialetan. Esperimentuan bertsio ingelesa eta alemana analizatzen dituzte erroreen bila. Bilaketa egiteko, errore-hiztegiak erabiltzen dituzte eta horrela dokumentu bakoitzaren errore-tasa lortzen dute. Gogoratu beharra dago, errore-hiztegien gainsorkuntza handiagoa dela testu espezializatuetan, beraz, emaitza poliki aztertu beharra dago. Horretarako, errore-tasa altueneko 20 dokumentuetan aurkitutako erroreak aztertzen dituzte eskuz, zenbat diren benetako erroreak ikusteko. Analisi horren arabera, ikusten da kasu batzuetan aurkitutako erroreak benetakoak direla, baina beste batzuetan, berriz, ez. Horregatik diote, testu espezializatuetan errore-tasa beti ez dela benetako erroreen adierazgarri. Dena den, 40 dokumentu aztertu ostean, emaitza da esperimentuak benetako 246 errore detektatzen dituela. Hori dela eta, itzulpen adibideak bilatzean, ideia sinple hau aplikatzea proposatzen dute: ez hartu adibide gisa errore-hiztegian sarreraren bat duten esaldiak. Horrela, itzulpen erroredunak hartzea ekiditen da eta dokumentuak luzeak direnez, ez da ia estaldurarik galduko.

3 Baliabideak: corpusak eta erroreen detekzioa

Aurreko atalean gure lanaren aurrekariak eta bereziki (Ringlstetter et al., 2006) ikerketa-lana zertan den laburbiltzen saiatu gara. Orain, xehetasunetan sartu baino lehen, gure azterketa-lanaren baliabideak eta tresnak azalduko ditugu, ezinezkoa baita prozedura bera jarraitzea horrelako azterketa euskaraz egiteko, honek dituen ezaugarriengatik. Dena den, gure lanaren helburua antzekoa da oso. Ohikoa da web-a erabiltzea egungo corpus handiak lortzeko hizkuntzaren inguruan lan egiten duten hainbat aplikazio garatzeko unean, eta euskara ez da salbuespena. Lan honetan, azterketa sakona egin nahi dugu hala osaturiko corpusetan (azterketa ortografikoa soilik) ikusteko ea posible den detektatzea eta kentzea corpusetik arazoak eman ditzaketen testuak. Azterketa hori egiteko, IXA taldean landutako hainbat tresna berrerabili ditugu. Erroreak aztertu nahi ditugunez, lehendabizi prozesadore morfologiko bat behar dugu eta hori landuta dauka IXA taldeak. Euskara bezalako hizkuntza eranskarietan, morfologia ezin da deskribatu hitz zerrenda baten bitartez eta hitz horien analisiaren bitartez. Hizkuntza horietan, egoera finituko teknologia (FST, *Finite State Tecnology*) erabili ohi da prozesadore morfologikoak garatzeko eta IXA taldean lan asko dago egina alor horretan. Lehendabizi Xerox etxeko tresnen bitartez (Alegria et al., 2002) eta azken urtetan *foma* tresnaren bitartez (Alegria et al., 2009), IXA taldean euskara estandarraren prozesadorea eta zuzentzaileak garatu dituzte. Lan honetan, aplikazio horiek erabiltzen dituzten transduktoreak[6] berrerabili ditugu, eta antzeko beste transduktoreak definitu behar izan ditugu *foma* tresnaren bitartez, errore mota jakin bat zein besteko erroreak identifikatzeko. Beraz, erreferentziazko lanaren planteamendu bera egingo dugu baina euskararen ezaugarrietara egokituta eta tresnak berrerabiliz.

Aipatzekoa da *"real word error"* deritzon arazoa tratatzea, gure lanaren helburuetatik kanpo geratzen dela. Arazo hori suertatzen da errore ortografiko baten ondorioz lortzen den hitza hizkuntzaren barruan dagoenean, ez baita detektatuko errore gisa. Adibidez: *naiz* aditz laguntzailearen ordez, *nahiz* juntagailua ageri bada, testuinguruaren arabera oker dena, ez dugu detektatuko errore gisa, hitz hori zuzena baita euskaraz, nahiz eta testuinguru horretan oker egon.

3.1 Corpusak

Azterketari ekiteko eta esperimentuak egiteko corpusak behar ditugu eta saiatu gara erreferentziazko lanaren antzeko tamainako corpusekin lan egiten. Dena den, corpusen osaketa ez da izan lan honen barruan egindako zerbait. Lankidetza baliatuz, gure lanerako corpusak lortzeko Elhuyarrek egindako lanetan oinarritu gara. Aurretik ezagutzen genuen Elhuyarrek eginiko lana "Web as corpus" gaiaren inguruan euskaraz eta bere laguntzari esker eskuratu ditugu gure analisirako corpusak.

[6] Txosten honetan zehar, *transduktore* eta *automata* terminoak sinonimotzat erabiliko ditugu nahiz eta transduktore terminoa zehatzagoa izan: automata bat izateaz gain, itzulpena egiten dela adierazten du *transduktoreak*.

3.1.1 Elhuyar: *Web as corpus* euskaraz

Gure lanaren esperimentuak egiteko, Elhuyarren eginiko ikerketak *Web as corpus* alorrean euskaraz sortu dituen corpusen azpimultzo bat berrerabili dugu. Nahiz eta lan honen helburua ez izan corpusen osaketa, beharrezkoa iruditzen zaigu gainetik azaltzea da nola sortu dituzten Elhuyarren corpus horiek, haien adierazgarritasuna bermatzeko. Euskarazko corpusak osatzea ez da bilaketa sinple bat egitea, bilaketa-motorrek ez baitute ongi funtzionatzen euskararekin (oro har, ez dute ongi funtzionatzen ingelesa ez diren hizkuntzekin). Bilaketaren emaitzak onak ez izatearen arrazoien artean daude euskarak morfologia aberatsa duela eta motorrek ez dituztela bilatzen euskaraz soilik dauden dokumentuak. Hori dela eta, teknika bereziak behar dira bilaketa arrakastatsua lortzeko: galdera morfologikoki zabaltzea, hizkuntza filtratzeko hitzak erabiltzea... Teknika horiek asko handitzen dute bilatzaileen errendimendua eta hori agerian geratzen da Elhuyarren landu diren bi proiektuetan *Corpeus* (Leturia et al., 2007b) eta *Eusbila* (Leturia et al., 2007a) .

Ez ditugu corpusen osaketari buruzko xehetasunak emango hemen, luze joko lukeelako horrek, baina osatze-prozesu horren ezaugarri nabarmenenak emango ditugu (Ringlstetter et al., 2006) lanean azaldu diren corpusen osaketarekin konparatu ahal izateko.

Corpus orokorrak. Corpus orokorrak sortzeko, euskarazko 500 hitz ohikoenak hartu, eta horien ausazko hirukoteak eman zizkieten bilatzaileei. Bilatzaileen erantzunei, (Sharoff, 2006) proposatzen duenaren antzera, ez diete inolako filtrorik pasatzen gaiari buruz, eta horrela osatzen den corpusa orokorra da. Metodo horren bitartez, 71.500 dokumentuko corpusa (81 milioi hitz) osatu zuten.

Corpus espezifikoak. (Leturia et al., 2008) artikuluan gai espezifiko bati buruzko dokumentuak eskuratzeko metodologia azaltzen dute. Metodologia horrek antza handia dauka 2.2.2 atalean azaldu denarekin. Ideia nagusia mini-corpus bat osatzea da, eskuz, gai jakin baten dokumentu batzuekin, gero horietatik lortzeko gaiaren termino zehatzak bilatzaileei emateko. Bilatzaileen erantzuna filtratu egin behar da gaiari buruzko dokumentuak direla ziurtatzeko. Bilaketa arrakastatsua izan dadin, garrantzitsua da mini-corpusa heterogeneoa izatea eta ahalik eta azpigai gehien jasotzea. Elhuyarren egin dituzten esperimentuen arabera, gaia oso espezifikoa bada 10 dokumentu inguru nahikoa dira mini-corpusa osatzeko, baina dokumentu gehiago erabili behar dira "hazi"corpus hori sortzeko gaia orokorragoa den heinean. Mini-corpusetik gaiko terminoak erauzi behar direla esan dugu, eta prozesu hori automatikoki egiten da gako-terminoak erauzteko landu duten metodo baten bitartez *DokuSare* proiektuaren barruan (Saralegi eta Alegria, 2007). Automatikoki lortzen den zerrenda hori eskuz errebisatzen dute bilaketaren arrakasta bermatzeko asmoz: termino espezifikoegiak, izen bereziak, termino orokorregiak, polisemikoak eta abar kentzen dituzte. Prozesu hori guztia luzea dirudien arren, beharrezkoa da euskara bezalako hizkuntzetan, oraindik ez baitaude eskura hainbat gairen inguruko hiztegi berezituak edo glosategiak.

Metodologia horren bitartez hiru corpus osatu dituzte Elhuyarren, "Informatika",

"Turismoa" eta "Bioteknologia" gaien inguruan. Osatu dituzten corpusen ezaugarriak 3.1 taulan ageri dira.

Gaia	Dokumentuak	Hitzak
Informatika	1672	2,5 M
Turismoa	1238	1,5 M
Bioteknologia	302	0,6 M

Taula 3.1: Elhuyarreko corpus berezien ezaugarriak.

3.1.2 Gure esperimentuetarako corpusen ezaugarriak

Aurreko atalean aipatu diren corpusen azpimultzo bat erabili dugu gure analisian. (Ringlstetter et al., 2006) ikerlaneko corpusen antzekoekin lan egiten saiatu gara eta hauek izan dira gure laneko corpusak:

1. Corpus orokorrak. Bi corpus ditugu, bata HTML formatuko dokumentuekin eta bestea PDF formatukoekin. Bi corpus horiek osatzeko, Elhuyarrek zoriz aukeratu ditu 71.500 dokumentuko corpus orokorrean 2.000 HTML dokumentu corpus baterako eta beste 1.000 PDF dokumentu beste corpuserako.

2. Corpus espezifikoak. Elhuyarrek jadanik osatuak zituen hiru gairen ingurukoak, noski. Berriro zoriz aukeratu ditu 500 dokumentu "Informatika" eta "Turismoa" gaietan eta "Bioteknologia" corpusa osorik eman digu. Corpus horietan ez dugu kontuan izan dokumentuen formatua, gaia baizik.

3.2 taulan corpusen ezaugarriak jasotzen dira. Ikusten denez, HTML corpus bakarrarekin egin dugu lan, eta ez biekin erreferentziazko lanean bezala (ikus 2.1 Taula) baina gure HTML corpusak dokumentu kopuru bikoitza du: 2.000 dokumentu eta ez 1.000. Bioteknologiako corpusaren tamaina besteena baino txikiagoa da, baina hori zen Elhuyarrek osatuta zuena. Kontuan hartu behar da gai teknikoagoa dela eta normala dela ez topatzea dokumentu gehiegi euskaraz gai horren inguruan.

Gaia	Formatua	Dokumentuak	Hitzak
Orokorra	HTML	2000	3,2 M
Orokorra	PDF	1000	2,5 M
Informatika	–	500	0,7 M
Turismo	–	500	0,6 M
Bioteknologia	–	302	0,4 M

Taula 3.2: Euskarazko corpusen ezaugarri nagusiak.

3.2 Erroreen detekzioa: osagaiak

Zalantzarik gabe atal honetan dator ezberdintasunik handiena gure lanaren eta erreferentziazko lanaren artean. Lehen esan dugunez, euskaraz ezin da morfologia deskribatu zerrenden bitartez beste hizkuntzetan egiten den antzera, eta erroreen detekzioa ere, ezin da zerrenden bitartez planteatu, erreferentziazko lanean egiten duten moduan. Ondorioz, guk ez ditugu errore-hiztegiak sortu. Erroreen identifikazioa egiteko transduktoreak (FST, *Finite State Transductor*) erabili ditugu eta hurrengo azpiataletan horien xehetasunak azalduko ditugu, baina lehendabizi, transduktoreen oinarrizko teknologiaren ezaugarri nagusiak emango ditugu, eta transduktoreak sortzeko erabili dugun *foma* tresna nola funtzionatzen duen azalduko dugu.

3.2.1 Egoera finituko morfologia

Lanaren baliabideak aurkeztu ditugunean esan dugu lan honetan IXA taldeak garatutako hainbat tresna berrerabili ditugula, horien artean, euskara estandarraren prozesadoreak erabiltzen duen transduktorea. Transduktorea egoera finituko teknologiaren bitartez lortzen den tresna da. Teknologia horrek arrakasta izan du analizatzaile eta sortzaile morfologikoak garatzean euskara bezalako hizkuntza eranskarietan (turkiarra, finlandiarra, zulua ...). Hizkuntzaren morfologia bi fitxategien bitartez definitzen da: (1) lexikoaren fitxategia, non morfemak eta morfemen ostean egon daitezkeen morfema multzoak (morfotaktika) deskribatzen diren; eta (2) erregela fonologikoen fitxategia, non morfemen kateatzearen ondorioz gerta daitezkeen aldaketak deskribatzen diren. Fitxategien konpilazioak transduktore bakar batean elkar daitezke zeinaren bitartez posible den hitzen analisia zein sorkuntza oso azkar egitea. Erregela fonologikoak paraleloak zein sekuentzialak izan daitezke (Alegria et al., 2009). Erregela paraleloak erabiltzen badira, haien ordena ez da garrantzitsua eta gainera, hitzen eta lexikoaren adierazpenaren artean ez dira tarteko lengoaiak definitu behar. Erregela paralelo horiek definitzeko unean, ordea, kontuan hartu behar da beste erregelek izan dezaketen efektua, eta hori, maiz, errore-iturri bihurtzen da. Erregela sekuentzialak erabiltzen badira, berriz, ordenak garrantzia du baina, oro har, erregela horien definizioa erosoagoa suertatzen da luzera begira.

Azken urtean, euskara estandarraren transduktorea kode irekiko tresna berri baten bitartez landu da, horrek ekartzen dituen onurekin. Tresnak *foma*[7] izena du eta Mans Hulden-ek garatua da (Hulden, 2009b). Lan honetan, erroreen azterketa egiteko, euskara estandarraren transduktorea berrerabili dugu eta antzeko beste batzuk definitu ditugu *fomaren* bitartez, mota bat zein besteko erroreak identifikatu ahal izateko, hots, errore tipografikoak, ezagutzakoak[8] eta OCRkoak identifikatzeko.

[7] *http://foma.sourceforge.net/*

[8] Ezagutza-erroreak zein konpetentzia-erroreak sinonimotzat erabili ditugu memorian zehar.

```
402-[siuc03 ~/scriptak]% foma
Foma, version 0.9.12alpha
Copyright © 2008-2009 Mans Hulden
...

foma[0]: load MORFO.fst
2.5 MB. 70498 states, 164641 arcs, Cyclic.
foma[1]: up
apply up> etxetik
Yetxe++Etik
apply up> etxera
Yetxe++ErA
Xetxera+
apply up> etxetikan
???
apply up> zuaitzak
???
apply up>
```

Irudia 3.1: `foma` tresnaren funtzionamenduaren adibidea.

3.2.2 *foma* tresna

foma tresna software librea da eta transduktoreak sortzeko erabili dugu. Berrerabili dugun euskara estandarraren transduktorea 2009 urtean zehar migratu da software horretara arazorik gabe. Arestian aipatu den moduan, transduktorea sortzeko lexikoa zein erregela fonologikoak definitu behar dira, eta gero, horiek konpilatuta, transduktorea lortzen da. Transduktorea hitzak analizatzeko zein sortzeko erabiltzen da, eta guri interesatzen zaiguna analisia da. Adibide bat emango dugu hobe ulertzeko transduktorearen funtzionamendua. 3.1 irudian pantaila zati bat ageri da non ikusten den `foma` aplikazioa exekutatu dela, gero `MORFO.fst` izeneko transduktorea kargatu dela `load MORFO.fst` komandoaren bitartez (aurretik sortua dago transduktore hori), eta, azkenik, `up`[9] komandoaren bitartez hitz batzuk analizatu direla: *etxetik*, *etxera*, *etxetikan** eta *zuaitzak**. Transduktore hori euskara estandarraren transduktorea da, hots, lexikoa eta erregelak euskara estandarra definitzen dutenak dira. Irudian ikusten denez, lehenengo bi hitzak, *etxetik* eta *etxera*, zuzenak dira lexikoa eta erregelen arabera, eta transduktoreak horien analisia itzultzen du (*etxera* adibidean, analisi bat baino gehiago). Azkeneko biak, berriz, *etxetikan** eta *zuaitzak**, ezin ditu analizatu, eta ondorioz, "???" ikurra itzultzen du.

Transduktoreak sortzeko eta maneiatzeko erabili dugu *foma*, baina horretaz gain, tresnak oso komando interesgarria dauka: `med` komandoa (Hulden, 2009a). Komando horren bitartez posible da transduktoreari hitz bat ematea (zuzena ala ez) eta berak hitz horretatik gertuen dauden hitzak itzultzea. Hitzen arteko distantzia kalkulatu ahal izateko, aldaketa

[9] `up` komandoak adierazten du azaleko forma emango dela analisia egiteko.

posible bakoitzak zenbateko kostea duen adierazi behar da. Adibidez, honako koste hauek definitzen baditugu:

```
Insert 1
Substitute 1
Delete 1
```

adierazten ari gara karaktere bat gehitzeak, ordezteak zein ezabatzeak 1 kostua duela, eta hitz batetik gertuen daudenak kalkulatzerakoan, koste horiek hartuko dira kontuan. `med` komandoarekin erlazionatuta bi aldagai daude: `med-limit` eta `med-cutoff` aldagaiak. `med-limit` aldagaiak bilaketaren distantzia mugatzen du, hots, adierazten du zenbat distantzia egon daiteke, gehienez, ematen den hitzaren eta itzultzen diren hitzen artean. `med-cutoff` aldagaiak berriz, gehienez zenbat hitz itzuliko diren adierazten du. Jar dezagun adibide bat garbi uzteko komandoa eta aldagaien funtzionamendua. 3.2 irudian `MORFOL.fst`[10] transduktorea kargatu da (aurretik sortua) eta kosteak definitzen dituen `typo.matrix` fitxategia irakurri da (`read cmatrix typo.matrix`). Horren ondoren, bi aldagaiei 1 balioa eman zaie `set` komandoaren bitartez, eta gero `med` komandoa exekutatu da bi hitzekin: *zuhatzak* amarenganaa**. Ikusten denez, bietan hitz zuzen bat itzuli du 1eko distantziara: *zuhaitzak* eta *amarengana*, hurrenez hurren. Ematen zaion hitza zuzena bada, hitz bera itzultzen du eta adierazten du kostea 0 dela (kasu hori ez da adibidean ikusten).

Aurrerago ikusiko dugun moduan, `med` komandoa oso baliagarria izango zaigu errore tipografikoak identifikatzeko.

Oro har, beraz, *foma* tresna transduktoreak sortzeko, transduktoreekin analisia egiteko eta `med` komandoa exekutatzeko erabili dugu. Transduktore batzuk aurretik landuta ziren eta guk *fomarekin* sortu eta berrerabili ditugu. Beste batzuk, berriz, lan honetarako definitu ditugu. Bestalde, hainbat script eta Perl programa idatzi behar izan ditugu bi zeregin nagusietarako: (1) *foma* aplikazioaren sintaxiarekin bat datozen datu-fitxategiak sortzeko eta (2) *fomak* itzultzen duen erantzuna guretzat egokia den formatuan uzteko. Xehetasunetan sartu gabe, hauxe da ideia nagusia:

1. Analizatu nahi dugun hitz edo token zerrendatik abiatuta, fitxategi bat sortu non transduktore egokia kargatzen den, eta gero, token bakoitzeko lerro bat idazten den `up` komandoa aurretik duelarik. Demagun fitxategi hori `fitx.dat` izena duela.

2. Aurreko fitxategia *fomarekin* exekutatu dugu eta automataren erantzuna beste fitxategi batean jaso (erantzuna.txt):

```
foma -q -f fitx.dat > erantzuna.txt
```

3. Transduktorearen erantzuna dugula, filtro egokiak pasako ditugu gure interesekoa den beste zerrenda (edo zerrendak) lortzeko, hala nola, zerrenda batean uzteko transduktoreak analizatzen dakiena soilik.

[10] MORFOL.fst euskara estandarrari dagokion transduktorea da

```
foma[0]: load MORFOL.fst
2.5 MB. 70498 states, 164641 arcs, Cyclic.
5.7 MB. 86061 states, 372739 arcs, Cyclic.
foma[2]: read cmatrix typo.matrix
Reading confusion matrix from file 'typo.matrix'
foma[2]: set med-limit 1
variable med-limit = 1
foma[2]: set med-cutoff 1
variable med-cutoff = 1
foma[2]: med
apply med> zuhatzak
Calculating heuristic [h]
Using confusion matrix.

zuhaitzak
zuha*tzak
Cost[f]: 1
apply med> amarenganaa
amarengan*a
amarenganaa
Cost[f]: 1
apply med>
```

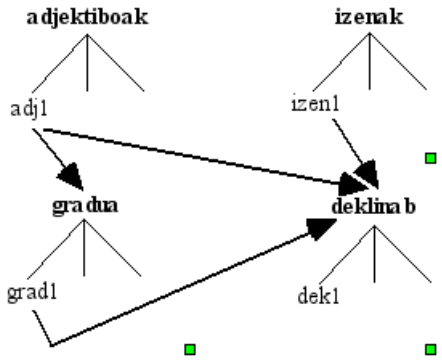
Irudia 3.2: `fomako med` komandoaren adibidea

3.2.3 Errore-gaien detekzioa: transduktore estandarra

Transduktore estandarra analizatu nahi dugun hitz zerrenda batean euskara estandar ez den guztia detektatzeko erabiliko dugu. Izan ere, guretzat, estandarra ez den guztia erroregaia da. Badakigu ez estandar guztiak multzo berean sartzea eztabaidagarria dela, guztiak ez direlako maila berekoak: *h* galtzea, *b* eta *v* nahastea, euskalkia erabiltzea euskara estandarraren ordez, ahoskatzen den moduan idaztea... baina lan honetan ez dugu kontuan hartu horrelakorik. Beraz, estandarra ez den guztia errore bat izan daiteke, eta aztertu behar dugu ea posible den errore hori identifikatzea, hots, ea posible den jakitea errore tipografikoa den, edo ezagutza-errorea den edo OCR errorea den.

Esan dugunez, euskara estandarraren transduktorea aspalditik dago garatuta (Alegria et al., 2002), eta azken urtean *foma* aplikaziora migratu da. Euskararen morfologia definitzeko egoera finituko morfologia erabiltzen da, hots, lexikoa eta erregelak definitzen dira. Lexikoaren bitartez, morfemak eta haien morfotaktika definitzen da. Erregelen bitartez, berriz, morfemak kateatzean gertatzen diren aldaketak definitzen dira.

Lexikoa definitzeko osagai nagusiak hiru dira: (1) lexikoaren sarrerak; (2) azpilexikoak, ezaugarri berdineko elementu lexikoen bildurak; (3) jarraitze-klaseak, paradigma baten osagaien morfotaktika definitzeko erabiliak. Adibide sinple batek kontzeptu horiek argitzen lagunduko digu. 3.3 irudian lexiko jakin baten eskema ageri da. Lexikoa *adjektiboak*



Irudia 3.3: Lexikoaren eskema: klaseak eta haien jarraipen posibleak.

eta *izenak* azpilexikoek osatzen dute, eta geziek morfotaktika adierazten dute, hots, adjektiboen ostean *gradua* zein *deklinab* jarraitze-klaseak joan daitezke; izenen ostean, berriz, *deklinab* soilik eta, azkenik, graduaren ostean *deklinab*. Irudiak adierazten duenez, klase bakoitzak bere sarrera lexikoak ditu (*adj1*, *izen1*, *grad1* ...). Lexiko hori definitzeko kodea 3.4 irudian ageri da. Bertan, klase bakoitzaren barruan sarrera lexiko batzuk definitu dira (gutxi batzuk, adibide moduan). Definizio horien arabera, lexiko horrekin sortzen dira, besteak beste: `ate`, `ate+ak`, `zakuR+ak`, `gorri+aren`, `gorri+ago+aren`, `mozkoR+en+ak`... eta antzeko kateak.

```
LEXICON Root
IZENAK; ADJEKTIBOAK;
LEXICON IZEN_JK
DEKLINAB;
LEXICON ADJ_JK
GRADUA; DEKLINAB;
LEXICON IZENAK
 ate  IZEN_JK;
 etxe  IZEN_JK;
 ur     IZEN_JK;
 zakuR  IZEN_JK;
```

```
LEXICON ADJEKTIBOAK
 gorri  ADJ_JK;
 mozkoR ADJ_JK;
LEXICON DEKLINAB
   #;
   +a #;
   +ak #;
   +aren #;
LEXICON GRADUA
   +ago DEKLINAB;
   +en DEKLINAB;
```

Irudia 3.4: Lexikoa definitzeko kodea. Adibide sinple bat.

Lexikoa definitu eta gero, erregelen bitartez adierazi behar da morfemak kateatzean suertatzen diren aldaketak. Adibideari jarraituz, lexikoa definitu eta gero, nolabait adierazi behar dugu R markak batzuetan r bihurtzen dela, baina beste batzuetan rr, izan ere, "zakur" eta "zakurrak" idazteko aukera eman nahi baitugu. Honako bi erregela hauek euskara estandarraren definizioan erabiltzen diren bi erregela dira:

```
define RR1 R -> r r || _ ( 1 ) MM ( h | R ) Bokal ;
define RR2 R -> r || _ ( 1 ) MorfBuk ;
```

Lehenengoak adierazten du *R* marka *rr* bihurtzen dela baldin eta atzetik morfema-muga (`MM`) eta bokal bat (`Bokal`) badatoz. Parentesi artean ageri diren sinboloak aukerakoak dira, hau da, ez dira derrigorrez ageri behar, eta adibidean azaltzen ez diren beste kasuekin erlazionatuta daude. Bigarren erregelak dioenez, *R* marka *r* bihurtzen da atzetik morfemaren bukaera (`MorfBuk`) besterik ez badago. Hala, lexikoaren definizioaren arabera sortutako `zakuR+ak` katea `zakurrak` bihurtuko da lehen erregelaren eraginez, eta `zakuR` katea, berriz, `zakur` bihurtuko da bigarrenaren eraginez.

Adibide sinple baten bidez morfologiaren definizioa nolakoa den adierazten saiatu gara transduktoreak duen informazioa eta egiten duen analisia hobeto ulertzeko. Berrerabili dugun euskara estandarraren transduktorea sortzeko behar den lexikoaren fitxategia oso handia da: 90.000 lerro inguru ditu eta bertan biltzen dira sarrera lexiko guztiak (80.000 inguru), morfotaktika eta informazio morfologikoa. Erregelak definitzen dituen fitxategia txikiagoa da. 75 erregela sekuentzial inguru ditu (lehen 23 erregela paraleloak ziren) baina ez dira azalpen sinplekoak xehetasunetan sartu gabe. Hori dela eta, bi erregela sinple besterik ez ditugu kopiatu hona adibidea azaltzeko, gero errazagoa izan dadin guk idatzitako erregela berriak azaltzea eta ulertzea.

Aipatu dugunez, euskara estandarraren transduktorea sortzeko *foma* erabili dugu. Lexikoaren definizioan ez da aldaketarik egin *fomaren* sintaxia bat datorrelako aurretik erabilitako Xerox aplikazioen sintaxiarekin. Erregelak, berriz, aldatu egin behar izan dira, lehen paraleloak zirelako eta orain sekuentzialak (Alegria et al., 2009). Transduktorea sortzeko, komando-fitxategi bat exekutatzen dugu *fomarekin* (`foma -f komandofitxategia`). 3.5 irudian komandoak ageri dira. Komandoei erreparatuz gero, hiru transduktore definitzen ditugula ikus daiteke:

1. MORFO.fst transduktorea. Transduktore estandar osoa da eta edozein hitzaren analisia egiteko erabili ahal izango dugu.

2. MORFOL.fst transduktorea. Transduktore estandarraren azaleko proiekzioa da (`.l` eragilearen bidez lortua), eta ez du hitzen analisia itzultzen. Hitz bat errekonozitzen badu, hitz bera itzultzen du irteeran, eta ez badu errekonozitzen, "???" marka itzultzen du.

3. MORFOERR.fst transduktorea. Aurrekoaren osagarria da (`-` eragilearen bidez lortua): transduktore estandarrak errekonozitzen ez duena errekonozitzen du. Hala, hitz bat emanez gero hitz bera itzultzen badu, horrek esan nahi du hitz hori ez dela estandarra, eta "???" itzultzen badu, berriz, estandarra dela.

Gure prozesurako, MORFOERR.fst transduktorea erabili dugu batez ere. Prozesu osoaren urratsak hurrengo atalean ikusiko ditugun arren, merezi du adibide txiki bat jartzea erabilera ulertzeko.

Demagun ondorengo fitxategia exekutatzen dugula *fomarekin*:

```
# lexikoaren irakurketa

read lexc /sx09a1/jirxuxen/foma/lex/xuxen-lex.txt.4.4.11
define LEX

# oinarrizko erregelek definizioa eta konposaketa lexikoarekin
source /sx09a1/jirxuxen/foma/rules/Erreg_foma.txt
define MORFO LEX .o. [ LEX.l .o. XY0 .o. ERREG .o. GARBIAZKEN ] ;
regex MORFO ;
save stack MORFO.fst
define MORFOL MORFO.l ;
regex MORFOL ;
save stack MORFOL.fst

define MORFOERR ?* - MORFO.l ;
regex MORFOERR;
save stack MORFOERR.fst
```

Irudia 3.5: Transduktore estandarra sortzeko komando-fitxategia. Komando hauek *fomarekin* exekutatzeko `foma -f komandofitxategia` exekutatu behar da.

```
load /sc01a4/users/acpetuzi/scriptak/MORFOERR.fst
up aktore
up gasteiztarra
up gustokoa
up behin
up iparrameriketako
up txiki
```

Hauxe izango da exekuzio horrek itzuliko duena:

```
???
???
gustokoa
???
iparrameriketako
???
```

Ikusten denez, hitz estandar bakoitzaren analisiak "???" marka itzultzen du, eta ez estandar bakoitza berriz, hitz bera itzultzen du. Beraz, "???" marka guztiak kentzen baditugu irteeratik, hitz ez estandarren zerrenda lortzen dugu, eta horixe da erroreak identifikatzen hasteko behar dugun abiapuntua.

3.2.4 Ezagutza-erroreen identifikazioa transduktorearen bitartez

Hitz ez estandarren artean, errore-mota bakoitzari dagozkion hitzak identifikatu nahi ditugu, horien artean, ezagutza-erroreei dagozkien hitz ez estandarrak. Gogoratu zer kontsideratzen den ezagutza-errorea: hizkuntzan konpetentzia nahikoa ez edo ezagutza mugatua dugulako egiten ditugun erroreak. Zein dira errore horiek euskaraz, eta nola detektatu, ez dugu lan honetan landu. Atal honetan, berriro, IXA taldean egindako lana berrerabili dugu. Izan ere, zuzentzaile ortografikoa garatzean, landu zuten IXAn euskarazko ohiko erroreen identifikazioa, errore "tipikoen" identifikazioa, hori funtsezkoa baita zuzentzaileak zuzenketa egokiak proposatu behar badizkio testua idazten ari den erabiltzaileari. Errore horiek detektatzeko, bai lexiko-sarrerak bai erregela berriak gehitu zituzten, horrela transduktore berri bat sortuz. Aurreko transduktorearekin egin dugun moduan, adibide pare bat emango ditugu transduktore berri honek dituen aldaketak nolakoak diren hobeto ulertzeko. 3.2.2 atalean ikusi dugu nola analizatzen duen transduktore estandarrak *etxetik* hitza (ikus 3.1 irudia): `etxe+Etik` eta ikusi dugu ere bai, transduktoreak ez duela analizatzen *etxetikan** hitza, hori ez delako estandarra. Gipuzkeraz, ordea, ohikoa da *etxetikan** forma erabiltzea, eta interesgarria da aldaera hori detektatzea zuzenketa egokia proposatzeko euskara estandarraren ikuspuntutik. Hala, aldaera hori (guretzat "ezagutza-errore") detektatzeko, lexikoan gehikuntza bat egiten da `+Etikan` atzizkia `+Etik` ordez ager daitekeela zehaztuz. Hori adibide bat besterik ez da, eta aldaketa asko daude lexiko mailan hainbat ezagutza-errore detektatzeko. Lan honetan, ezagutza-erroreak detektatzeko IXAk garatutako lexiko gehigarria dagoen moduan berrerabili dugu.

Lexikoaz gain, erregelak daude, eta ezagutza-erroreak detektatzeko erregela berriak gehitzen dira. 65 erregela inguru gehitu dira transduktore berria sortzeko, eta aurrekoan bezala, erregela pare bat emango ditugu adibide gisa . Honako bi erregela hauek ohiko bi errore hartzen dituzte kontuan: h letraren galera eta x letra j letrarekin nahastea, hurrenez hurren. Erregeletan ageri den `(->)` ikurraren parentesiek aldaketa aukerakoa dela adierazten dute, errorea ez baita beti gertatzen.

```
define HGALDU h  (->) 0   || _  ;
define XJ  x (->) j ||  [ Hasiera | Bokal  ]  _  Bokal ;
```

Beraz, errore tipikoak detektatzeko eta zuzenketa egokiak proposatzeko transduktore berriaren erregelak *fomara* migratu eta transduktorea sortu dugu gure lanean berrerabiltzeko. Transduktorea sortzeko, 3.6 irudian ageri den komando-fitxategia exekutatzen dugu *fomarekin*. Bertan ageri diren komandoei erreparatuz gero, erregeletako bi fitxategi irakurtzen direla ikusten da: `Erreg_foma.txt` fitxategia, erregela estandarrak dituena; eta `Erreg_foma_tip.txt` fitxategia, ezagutza-erroreak kontuan hartzeko erregelak dituena. Transduktoreei dagokionez, bi definitu dira:

1. MORFOTIPIKOAK.fst transduktorea. Transduktore osoa da ezagutza-erroreak kontuan hartuta. Hitz bat ematen bazaio, hitzaren analisia itzuliko du.

2. MORFOTIPZUZ.fst transduktorea. Aurrekoaren azaleko proiekzioa da eta ez du analisia itzultzen. Hitz bat analizatzeko emanez gero, hitz bera itzuliko du errekonozitzen badu, eta bestela, "???" ikurra itzuliko du.

```
# Aldaerak dituen lexikoaren irakurketa
read lexc /sx09a1/jirxuxen/foma/lex/xuxen-lex_ald.txt.4.4.11
define LEXALD

# oinarrizko erregelen definizioa eta konposaketa
source /sx09a1/jirxuxen/foma/rules/Erreg_foma.txt

#Aldaerak tratatzeko ditugun erregelen definizioa
source /sx09a1/jirxuxen/foma/rules/Erreg_foma_tip.txt

#Transduktore berriaren definizioa aldaerak kontuan hartuz
define MORFOTIPIKOAK LEXALD .o. [ LEXALD.l .o. XY0 .o. ERREG .o. TIPGUZTIAK .o. GARBIAZKEN ] ;
regex MORFOTIPIKOAK ;
save stack MORFOTIPIKOAK.fst

#Aldaerak kontuan hartuz errekonozitzen duenaren azala bueltatzeko automata
define MORFOTIPZUZ MORFOTIPIKOAK.l ;
regex MORFOTIPZUZ ;
save stack MORFOTIPZUZ.fst
```

Irudia 3.6: Ezagutza-erroreak identifikatzen dituen transduktorea sortzeko komando-fitxategia. Komando hauek *fomarekin* exekutatzeko `foma -f komandofitxategia` exekutatu behar da

Gure prozesurako, MORFOTIPZUZ.fst transduktorea erabili dugu, batez ere. Ikus dezagun adibide sinple bat transduktorearen lana zein izango den ulertzeko. Demagun ondorengo fitxategia exekutatzen dugula *fomarekin*:

```
load /sc01a4/users/acpetuzi/scriptak/MORFOTIPZUZ.fst
up filman
up infeliz
up iparrameriketako
up kurrikulumeko
up lugar
up xoxik
```

Hauxe da exekuzio horrek itzultzen duena:

```
filman
???
iparrameriketako
kurrikulumeko
???
xoxik
```

MORFOTIPZUZ.fst transduktoreak hitza errekonozitzen badu, hitz bera itzultzen du, eta errekonozitzen ez badu, berriz, "???" itzultzen du. Adibideko hitz guztiak ez estandarrak dira, baina aldaerak kontuan hartuz, transduktorea gai da horietako batzuk identifikatzeko. Irteera horretatik "???" markak kentzen baditugu, identifikatutako ezagutza-erroreen zerrenda lortuko dugu.

HAP masterra

3.2.5 OCR erroreen identifikazioa transduktorearen bitartez

OCR erroreak identifikatzeko beste transduktore bat definitu dugu. Aurrekoaren ildotik, ideia sinplea da: transduktore estandarrari hautazko erregela batzuk gehitu dizkiogu, OCR aplikazioek sortzen dituzten aldaketak detektatzeko. Baina, zein dira erregela egokiak? Nondik lortu patroiak jakiteko OCR aplikazioen aldaketa ohikoenak? Erregela horiek idazteko unean iturburu bat baino gehiago izan dugu kontuan:

- Iaz, HAP masterrean lan bat egin nuen OCR inguruan "Morfologia eta Sintaxia" ikastaroaren barruan, eta lan horretan identifikatu genituen ohikoak diren errore batzuk OCR aplikazioetan: *o*, *e* eta *c* letrak nahastu, *m* eta *rn* nahastu eta abar.
- Eleka enpresan egin dute lan OCR inguruko erroreak detektatzen, eta lankidetza baliatuz haiek erabilitako patroiak lortu genituen aurreko atalean sortutakoekin konparatzeko eta osatzeko.
- Azkenik, erreferentziazko lanean ematen dituzte OCR erroreak sortzeko erabiltzen dituzten patroiak (ikus 2.6 Taula). OCR erroreak ez daudenez hizkuntzari lotuak, letren itxurari baizik, posible izan dugu patroi horiekin gure erregela batzuk berretsi, eta berriren bat gehitu.

Hala, aurreko iturburuetatik lortutako erregela guztiak hartu eta gure transduktore berria sortu dugu. OCR aldaketak itxurarekin lotuta daude: *1* eta *l* karaktereak nahastu, *o* eta *c* nahastu... Hainbat kasutan begiztak sor daitezke erregela horien artean. Adibidez, bi OCR aldaketa hauek posible dira: *d* letra *cl* bihurtzea eta *c* letra *e* bihurtzea; bi aldaketa horiek bata bestearen atzetik egiten badira, bukaeran izango dugu *d* letra *el* bihurtu dela, eta ez dugu horrelakorik nahi. Horrelako arazoak saihesteko, 24 OCR erregelak era paraleloan definitu ditugu *foman* honako kode honen bitartez:

```
define PARAL [  c (->) e , e (->) c , c (->) o , o (->) c , l (->) i ,
i (->) l , l (->) 1 , 1 (->) l , rn (->) m , m (->) rn , ri (->) n ,
n (->) ri , cl (->) d , d (->) cl , o (->) "0" , "0" (->) o ,
rt (->) n , rt (->) it , rm (->) nn , rri (->) m , m (->) rri ,
ni (->) m , li (->) h , u (->) ir ] ;
```

Ikusten denez, gehitu ditugun erregelak askoz sinpleagoak dira ez baitute testuingurik. Aurrekoetan bezala, transduktorea sortzeko 3.7 irudiko komando-fitxategia exekutatzen dugu, eta MORFOCRZUZ.fst transduktorea lortzen dugu.

Gure prozesuan, ez estandarren artean OCR erroreak identifikatzeko MORFOCRZUZ.fst transduktorea erabili dugu. Ikus dezagun funtzionamenduaren adibide bat. Honako fitxategi hau exekutatzen badugu *fomarekin*:

```
load /sc01a4/users/acpetuzi/scriptak/MORFOCRZUZ.fst
up hcrtxe
up clituen
up iparrameriketako
```

```
#lexikoaren irakurketa
read lexc /sx09a1/jirxuxen/foma/lex/xuxen-lex.txt.4.4.11
define LEX

# oinarrizko erregelek definizioa eta konposaketa
source /sx09a1/jirxuxen/foma/rules/Erreg_foma.txt

# OCR erroreak tratatzeko ditugun erregelen definizioa
source /sx09a1/jirxuxen/foma/ocr/Erreg_foma_ocr.txt

#Transduktore berriaren definizioa: OCR erregelak kontuan hartuz
define MORFOCR LEX .o. [ LEX.l .o. XY0 .o. ERREG .o. GARBIAZKEN .o. PARAL ] ;
define MORFOCRZUZ MORFOCR.l ;
regex MORFOCRZUZ ;
save stack MORFOCRZUZ.fst
```

Irudia 3.7: OCR erroreak identifikatzen dituen transduktorea sortzeko komando-fitxategia.

hauxe da lortzen den erantzuna:

```
hcrtxe
clituen
???
```

Ikusten denez, automatak hitza errekonozitzen badu, hori bera itzultzen du, eta ez badu errekonozitzen "???" marka itzultzen du. Kentzen badira "???" markak, posible izango dugu jakitea zer identifikatzen den OCR erregelen bitartez.

3.2.6 Errore tipografikoen identifikazioa: `med` komandoa eta transduktorea

Errore tipografikoak identifikatu ahal izateko aurreko prozesu bera jarrai daiteke: transduktore estandarrari erregela batzuk gehitu horien bitartez errore tipografikoak detektatzeko. Ikusi dugu erreferentziazko lanean zein diren errore tipografikoak: ezabatzeak, gehitzeak, ordezteak eta transposizioak. Errore horiek detektatzeko erregelak definitzea ez da zaila baina erregela horiek izugarri zabaltzen dute automataren tamaina: edozein letra ezaba daiteke, edozein biren artean beste bat ager daiteke, edozein letra ordez daiteke beste edozeinekin... eta *foma* aplikazioak beste aukera eskaintzen digu, askoz hobe, errore tipografiko gehienen identifikazioa egiteko, eta hori `med` komandoa da (Hulden, 2009a). Hori dela eta, errore tipografikoen identifikazioa beste era batera egin dugu, eta bi urratsetan banatu dugu.

`med` komandoa

foma tresna aurkeztu dugunean 3.2.2 atalean, `med` komandoaren funtzionamendua azaldu dugu. Gogoratu, posible da hitz bat emanda gertuen dauden hitz zuzenak aurkitzea transduktore estandarra erabiliz. Errore tipografikoak identifikatzerakoan mugatu egingo dugu

aldaketa tipografikoen kopurua hitz batean, eta muga hori 1ekoa izango da, hots, letra bat alda daiteke: ezabatu, ordeztu edo gehitu, baina bat bakarrik. `med` komandoa erabiltzeko lehen azaldu dugun koste funtzioa definituko dugu: `Insert 1`, `Substitute 1`, `Delete 1` eta distantziaren muga 1era jarriko dugu `med-limit` aldagaiaren bitartez. Horrez gain, hitz bat errore tipografiko gisa identifikatzeko nahikoa da hitz zuzen bat aurkitzea 1 distantziara. Gure helburua ez da hitza zuzentzea, jakitea baizik existitzen dela hitz estandar bat 1eko distantziara, hori baita baldintza errore tipografiko gisa identifikatzeko hitz ez estandar hori. Beraz, `med-cutoff` aldagaiari 1 balioa emango diogu eta horrela, `med` komandoak hitz bat aurkitzen duenean bilaketari utziko dio.

Demagun hurrengo fitxategia exekutatzen dugula fomaren bitartez:

```
load /sc01a4/users/acpetuzi/scriptak/MORFOL.fst
read cmatrix /sc01a4/users/acpetuzi/scriptak/typo.matrix
set med-cutoff 1
set med-limit 1
med bezela
med kurrikulumeko
med neskatilren
med genukela
```

Hauxe da irteeran jasoko dugun erantzuna:

```
bezala
bezela
Cost[f]: 1

Reached cutoff of 1.

neskatil*en
neskatilren
Cost[f]: 1

genukeela
genuke*la
Cost[f]: 1
```

Ikusten denez, ez badu aurkitzen hitz estandar bat 1 distantziara “Reached cutoff of 1” mezua azaltzen da, eta bestela, aurkitu duen hitza ageri da, eta atzetik ez estandarra eta kostea. Irteera hori jasotzen badugu, posible dugu Perl programa baten bitartez zerrenda batean uztea `med` bitartez identifikatzen direnak soilik. Aurreko adibidean: *bezela*, *neskatilren* eta *genukela*.

Transposizioen transduktorea

Transposizioak ezin dira detektatu `med` bitartez distantziaren mugak 1 balioa badu, baina errore tipografikoen barruan daude. Beraz, horiek identifikatzeko beste zerbait egin behar dugu, eta berriro, transduktore berri bat sortzea izan da bidea. Horretarako, erregelak gehitu behar dizkiogu transduktore estandarrari. Transposizioak detektatzeko, alfabetoko

```
#lexikoaren irakurketa
read lexc /sx09a1/jirxuxen/foma/lex/xuxen-lex.txt.4.4.11
define LEX

# oinarrizko erregelek definizioa
source /sx09a1/jirxuxen/foma/rules/Erreg_foma.txt

#Errore tipografiko bakarra tratatzeko erregelen definizioa
source /sx09a1/jirxuxen/foma/tipogr/rule_truke.txt

define MORFOTRUKE [ LEX.l .o. XY0 .o. ERREG .o. GARBIAZKEN ].l .o. Osoa ;

define MORFOTRUKEZUZ MORFOTRUKE.l ;
regex MORFOTRUKEZUZ ;
save stack MORFOTRUKEZUZ.fst
```

Irudia 3.8: Transposizioak identifikatzen dituen transduktorea sortzeko komando-fitxategia.

letra bakoitzeko erregela bat definitu dugu non transposizioa markatzen dugun sinbolo baten bidez. Adibidez, *a* letraz hasten den bikote batean, transposizioa gerta daitekeela adierazteko bi erregela hauek erabiliko ditugu:

```
define TrukeA  [ "a" Alfabeto (->) "4" ... "a"  ] ;
define Kendu4 [ "4" Alfabeto -> 0 ] ;
```

Lehendabiziko erregelak transposizioa markatzen du `4` sinboloaren bitartez eta *a* letra bikotearen eskuinean gehitzen du. Gero, bigarren erregelak `4` sinboloa eta atzetik datorren letra ezabatzen ditu. Bi erregela horiek aplikatzen badira, *a* letraz hasten den bikotearen transposizioa ikusten da. Adibidez, *artea* hitza *4aratea* bihur daiteke lehenengo erregelaren eraginez, eta *4aratea* tokena *ratea* bihurtzen da bigarrenaren eraginez. Hasiera eta bukaera soilik hartuta ikusten da *artea* hitza, *ratea* bihur daitekeela, hots, hasierako *ar* bikotearen transposizioa suerta daitekeela.

Letra guztietarako horrelako erregela bat definituz gero, edozein bi letraren arteko transposizioa definitzen dugu. Azkenik, falta den gauza bakarra hitz batean gerta daitezkeen transposizio kopurua mugatzea da, lehen beste errore tipografikoekin egin dugun antzera. Ez dugu transposizio bat baino gehiago detektatzerik nahi eta horretarako honako erregela hau erabiliko dugu:

```
define Mugatu [  "4" => .#. Alfabeto* _ ] ;
```

Erregela horrek adierazten du ezin dela `4` sinboloa egon, aurretik zer edo zer badago alfabetokoa ez dena, hau da, gure kasuan beste `4` sinboloren bat badago. Eragina zuzena izan dadin, erregela hori `Kendu4` izeneko erregelaren aurretik egon behar du, ordena kontuan hartuta. Horrela mugatzen da transposizio kopurua hitz batean.

Transduktore berria sortzeko komandoak 3.8 irudian ageri dira. Bertan ikusten denez, MORFOTRUKZUZ.fst automata sortzen dugu gero horrekin transposizioak detektatzeko. Ikus dezagun adibide sinple bat transduktorearen funtzionamendua ulertzeko. Demagun ondorengo fitxategia exekutatzen dugula *fomarekin*:

```
load /sc01a4/users/acpetuzi/scriptak/MORFOTRUKEZUZ.fst
up irentis
up zirkunstatzia
up algortimo
up amaregnana
up ziurasko
```

Hauxe da exekuzioaren erantzuna:

```
irentis
???
algortimo
amaregnana
???
```

Beraz, transposizioen bitartez identifikatzen dituen hitz ez estandarrak, bere horretan itzultzen ditu, eta identifikatzen ez dituenen ordez, "???" marka itzultzen du.

3.3 Errore-detekzioaren mugak nola estimatu

Ikusi dugu aurreko ataletan erroreak detektatzeko tresnak eta metodoa nolakoak diren. Detekzio horren kalitatea neurtzeko beharrezkoa da haren mugak neurtzea. Erreferentziazko lanean, muga horiek izendatzeko *underproduction* eta *overproduction* terminoak erabili dituzte[11], haiek errore-hiztegiak sortu egiten baitituzte. Gure metodoan, berriz, detekzioa ez da sorkuntzan oinarritzen eta termino horiek ez dira agian egokienak. Beraz, beste bi termino erabiliko ditugu baina muga berberak estimatzeko: (1) dauden erroreetatik zenbat geratzen diren detektatu gabe: horri azpidetekzioa deituko diogu; eta (2) detektatzen direnetatik zenbat ez diren benetako erroreak: horri gaindetekzioa deituko diogu.

Azpidetekzioa. Erroreen identifikazioa egiteko, gure abiapuntua da lehendabizi ez estandarra den guztia detektatzea transduktore estandarraren bitartez. Beraz, euskararen lexikoa zein erregelak erabiltzen ditugunez, gutxi izango dira detektatuko ez ditugun erroredun hitzak. Beste kontua da estandar ez den guztia multzo berean sartzea, hor kasuistika handia dagoelako, baina gai hori planteatzea lan honetatik kanpo geratzen da, momentuz. (Ringlstetter et al., 2006) lanean behartuta zeuden azpisorkuntzaren azterketa egitera, bazekitelako hainbat kasutan mugatzen ari zirela erroreen sorkuntza errore-hiztegiak handiegiak ez izatearren (ikus 2.4 atala). Euskararen kasuan ez dugu horrelako muga, gure abiapuntua, beti, ez estandarra den guztia identifikatzea baita, gero hortik identifikatzeko errore-mota bakoitza. Guretzat arazoa da ez identifikatzea zein errore-motak sortu duen token ez estandar horiek, eta ondorioz, detektatu gabe geratzea. Hori gertatzeko aurreikusten ditugun arrazoiak dira:

[11] azpisorkuntza eta gainsorkuntza terminoekin itzuli ditugu ingelesezko termino horiek

- Ezagutza-errore “berriak” izatea, hots, erabili dugun transduktoreak jasotzen ez dituena ez zirelako aldaketak edo erregelak egin horiek detektatzeko.
- Errore tipografikoak izatea baina 2 baino gehiagoko distantziara.

Asko badira multzo horretan: ez-estandarrak eta identifikatu gabeak, eskuzko analisia egingo dugu maiztasun altuenekoekin eta zoriz aukeratutako beste batzuekin. Kopuruen arabera erabakiko dugu zenbat hartu (ikus 5.5.2 atala).

Gaindetekzioa. Gaindetekzioa analizatu behar dugu ikusteko errore gisa identifikatzen dugun guztia benetako errorea den. Erreferentziazko lanean ikusi dugun bezala, guri ere gertatuko zaigu errore gisa identifikatutako batzuk benetako erroreak ez izatea: agian neologismoak direlako eta oraindik ez daudelako lexikoan sartuta, edo euskalki jakin bateko hitzak direlako... Hitzak zenbat eta laburragoak izan, errazagoa da errore tipografiko gisa kontsideratzea hala ez den hitz bat. Gaindetekzioa neurtzeko proposamena aurrekoaren antzekoa da: identifikatzen diren erroreak maiztasunaren arabera ordenatu, eta horien artean eskuzko analisi bat egin bai maiztasun altuenekoekin eta zoriz aukeratutako beste batzuekin. Kopuruen arabera erabakiko da eskuzko analisirako zenbat adibide hartu (ikus 5.5.1 atala).

4 Esperimentuak

Aurreko atalean gure azterketaren baliabideak planteatu ditugu eta azterketaren oinarria zein den esan dugu, hau da, zer tresnak erabiliko ditugu erroreen detekzioa egiteko. Atal honetan prozesua zein den adieraziko dugu argi uzteko jarraitu beharreko urratsak.

1. Corpuseko fitxategi bakoitza hartu eta testu soila bihurtu. Elhuyarretik pasa dizkiguten corpusetan, fitxategien formatua HTML da kasu guztietan (PDF jatorria dutenak HTML kodera bihurtu dituzte) eta testu hutsa lortzeko `w3m` aplikazioa erabili dugu.

2. Testu-fitxategi bakoitza tokenizatu eta filtratu. Erreferentziazko lanean erabili dituzten irizpideak jarraituko ditugu, eta ondorioz, ez ditugu analizatuko letra larriz hasten diren tokenak, 5 letra baino gutxiago dituzten tokenak eta digituak edo karaktere arraroak dituzten tokenak[12].

3. Aurreko urratsean lortutako token zerrenda analizatu transduktore estandarraren bitartez estandarra ez den guztia detektatzeko eta zerrenda batean uzteko. Horretarako, Perl lengoaian idatzi behar izan ditugu script batzuk transduktorearen erantzuna tratatzeko. Azkenik, beraz, token ez estandar zerrenda bat lortu dugu dokumentu bakoitzeko.

4. Token ez estandarren zerrenda bakoitza ezagutza-erroreak detektatzen dituen transduktoreari eman diogu, eta erantzuna scriptekin manipulatu eta gero, zerrenda berri bat lortu dugu non ezagutza-erroreen bitartez identifikatzen ditugun token ez estandarrak geratzen diren.

5. Urrats honetan errore tipografikoak identifikatu nahi ditugu eta horretarako bi urrats eman ditugu. Lehendabiziko urratsean, `med` komandoaren bitartez hiru motako errore tipografikoak (ezabatu, gehitu, ordeztu) identifikatu ditugu eta horiei dagozkien token ez estandarren zerrenda bat lortu dugu dokumentu bakoitzeko. Bigarren urratsarekin, transposizioak detektatzen dituen transduktorea erabili dugu eta horrela identifikatu ditugu transposizioei dagozkien erroreak eta beste fitxategi batean utzi ditugu. Azkenik, scripten bitartez bi zerrenda horiek fusionatu ditugu, bakar batean biltzeko errore tipografikoekin identifikatzen ditugun tokenak.

6. Azkeneko urratsean OCR erroreak identifikatu nahi ditugu eta horretarako OCR erroreak detektatzen dituen transduktorea erabili dugu. Aurreko urratsetan bezala, dokumentu bakoitzeko OCR erregelek identifikatzen dituzten token ez estandarren zerrenda lortu dugu scripten bitartez.

[12]Digituen kasuan, salbuespen bat egin dugu 0 edo 1 digituak dituzten tokenekin, izan ere, OCR aplikazioen errore bat izan ohi da “o” letra “0” digituarekin nahastea, eta baita “l” letra “1” digituarekin. Hori dela eta, ez ditugu bi digitu horiek besterik dituzten tokenak kenduko (baleude, noski).

Aurreko urratsen ondoren, corpuseko dokumentu bakoitzeko informazio hau daukagu:

- Token zerrenda bat oinarrizko tokenizatzailea eta filtroa pasa eta gero lortutakoa.
- Analizatu dugun token zerrenda, aurreko zerrendatik hainbat token kendu eta gero: letra larriz hasten direnak, digituak dituztenak (0 eta 1 dituztenak izan ezik) eta luzera minimoa ez dutenak.
- Analizatutakoen artean ez estandarrak direnen zerrenda.
- Ez estandarren artean, ezagutza-erroreen bitartez identifikatzen ditugunen zerrenda.
- Ez estandarren artean, errore tipografikoen bitartez identifikatzen ditugunen zerrenda.
- Ez estandarren artean, OCR erroreen bitartez identifikatzen ditugunen zerrenda.

Zerrenda horiekin guztiekin, hainbat kalkulu estatistiko egin ditzakegu bai dokumentu bakoitzeko bai corpus osoarekiko.

4.1 Sistemaren doikuntza

Aurreko prozesua corpus batean egin eta gero, lehen emaitzak analizatu genituen prozesuaren funtzionamendua egiaztatzeko. Esan dugunez, hiru errore motak identifikatzeko unean automatei eman diegun hasierako informazioa bera izan da kasu guztietan: jatorrizko dokumentuan token ez estandarrak direnen zerrenda, hau da, transduktore estandarrak errore gisa markatu dituenak. Bestalde, hiru errore mota ezberdinak identifikatzen ditugunean: ezagutzakoak, tipografikoak eta OCRkoak, emaitzak ez dira disjuntuak izango eta egongo dira token ez estandarrak automata bat baino gehiagorekin identifika daitezkeenak. Adibidez, *zuaitza** tokena identifika daiteke ezagutza-errore gisa, *h* galtzea ezagutza errore gisa kontsideratuta dagoelako, baina posible da ere bai, errore tipografiko gisa identifikatzea, *h* letra (beste edozein bezala) ezabatua izan delako. Efektu hori agerian geratzen zen lehendabiziko estatistiketan dokumentu batzuen zerrenden kopuruak aztertu genituenean: errore tipografikoak beti ziren gehiengoa, diferentzi handiz (askotan, erroreen % 90 inguru errore tipografiko gisa identifika zitekeen). Efektu hori saihesteko, emaitzetako ebakidura horiek tratatzea erabaki genuen eta zerrenden artean banaketak egin, horrela argi izateko zenbat errore identifikatzen diren mota bakoitzeko. Hauxe izan zen gure irizpidea hori egiteko unean: lehentasuna eman ezagutza-erroreen bitartez identifikatutakoari, hau da, token bera bi eratara identifikatzen bada, ezagutza-errore eta errore tipografiko gisa, edo ezagutza- eta OCR errore gisa, ezagutzakoa dela kontsideratuko dugu, hots, beste zerrendetatik kendu egingo dugu. Beste bi zerrenden artean, tipografikoen eta OCRkoen artean, ez genuen tratamendurik egin, datuetan ikusi genuelako OCR bitartez identifikatutakoa oso zerrenda motza zela beti.

Lehen analisi horretan ere, identifikatu gabe geratzen zirenen tokenen azterketa egin genuen, ikusteko nolakoak ziren horiek eta ikusteko ea emaitza arraroak edo horrelakorik

ikusten genuen. Agian, azterketa horren bitartez posible izango zen zer edo zer hobetzea gure errore-analisian. Asmo horrekin, identifikatzen ez genituen ez estandar token guztiekin zerrenda bakar bat egin genuen, eta maiztasunaren arabera ordenatu genuen. Zerrenda horretan ageri ziren, adibidez: *guciac*, *porque*, *language*, *google*, *zonbeit* ... Logikoa zen horiek bertan egotea eta ezin izatea identifikatzea, baina ageri ziren beste batzuk ere bai, erroreak ez zirenak: *bakar-bakarrik*, *oso-oso*, *guzti-guztia* ... Arazotxo bat genuen "bikotedun" token horiekin, transduktore estandarrak ez baitzituen estandartzat jotzen, eta, guk ezin izango genuen horiek inoiz identifikatu erroreak detektatzeko gure erregelekin. Beraz, filtro bat egitea erabaki genuen token horiek ez tratatzeko ez estandar gisa. Filtro hori hasieran pasako genuen token bikotedun horiek detektatzeko eta zerrendatik kentzeko, transduktore estandarrari pasa baino lehen. Hauxe da filtro berriak egiten duena:

1. Bikotedun tokenak izan daitezkeenak aurkitu. Hala izateko baldintza nagusiak bi dira: tokenak gidoi ("-") karakterea dauka eta gidoi aurreko zatia, gidoi ostean ageri da berriro, agian, zer edo zer erantsita (gehienetan, deklinazioa). Adibide batzuen bitartez errazagoa da baldintza azaltzea. Baldintza horien arabera, ***oso-oso**ak*, ***bakar-bakar**rik*, ***sarri-sarri***, ***ber-ber**a*... bikotedun-gaiak dira. Ordea, *login-to*, *nondik-norakoak*, *bait-da*... ez dira bikotedun-gaiak ez dituztelako bi baldintzak betetzen.

2. Bikotedun-gai horietan, gidoi osteko zatia transduktore estandarraren bitartez analizatu dugu, eta transduktoreak token estandarra dela badio, orduan filtroak token bikotedun hori "onartuko" du, eta bestela ez.

Filtroak onartzen dituen token bikotedunak hasierako zerrendatik kentzen ditugu, hau da, ez ditugu transduktore estandarraren bitartez analizatzen, aurreko analisian ikusi baitugu transduktoreek ez dituztela identifikatzen. Beraz, token horiek hasieran filtratu ditugu, hau da, atalaren hasieran ikusi ditugun urratsen arabera, lehenengo bi urratsen ondoren filtro berria pasa dugu eta token bikotedun horiek kendu ditugu hirugarren urratsarekin jarraitu baino lehen.

Dena den, filtratzen den guztia gorde egiten dugu, gero aztertu ahal izateko filtroak ongi funtzionatzen duen. Informazio hori, bikote onartuen zerrenda, dokumentu bakoitzeko gorde dugu lehendabizi, eta gero, fitxategi bakar batean bildu ditugu guztiak eta maiztasunaren arabera ordenatu. 4.1 taulan HTML corpusean maiztasun altueneko 10 bikotedun onartuak ageri dira. Horiek guztiak filtratzea zuzena dela esan dezakegu, baina filtroa ez da perfektua eta badira batzuk agian kendu beharko ez genituzkeenak. Adibidez: *ezarian-ezarian* (3 aldiz), *dauden-daudenean* (2 aldiz), *xut-xuta* (2 aldiz), *tokian-tokian* (2 aldiz) ... Posible izango litzateke filtro hori gehiago lantzea bikote batek bete behar dituen baldintzen inguruan, baina lehen hurbilpen gisa, ontzat eman dugu hemen azaldutakoa.

Lehen emaitzak analizatuz, beraz, aurreko bi arazoak detektatu genituen: errore tipografikoak barne hartzen zituztela ezagutza-errore asko, eta, token bikotedunak identifikatu gabeko errore-zerrendan ageri zirela bukaeran. Horiei konponbidea eman ondoren berriro corpusen analisiari ekin genion. Hurrengo atalean lortutako emaitzak azalduko ditugu.

Token bikoteduna	**Maiztasuna**
bakar-bakarrik	49
oso-oso	48
berdin-berdin	46
apurka-apurka	40
pixkanaka-pixkanaka	38
zuzen-zuzenean	27
egunero-egunero	27
labur-labur	26
guzti-guztiak	22
emeki-emeki	17

Taula 4.1: Filtroak onartu dituen token "bikotedun" batzuk. Hauek kendu egin ditugu hasierako token zerrendatik eta horrela transduktore estandarrak ez ditu analizatu. HTML corpus orokorrean maiztasun altueneko 10ak ageri dira taulan, ondoan haien maiztasun absolutua ageri delarik

5 Emaitzak

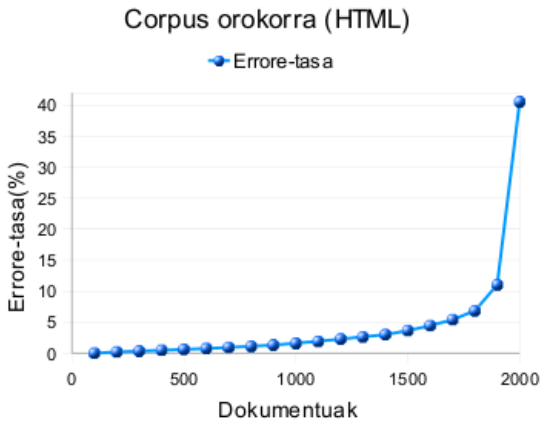
Atal honetan corpusetan aurkitutako erroreen berri emango dugu bai grafikoen bitartez, bai hainbat batez bestekoren bitartez. Emaitzen ebaluazio zehatza egitea ezinezkoa denez datu asko direlako, eskuzko analisiak egin ditugu hainbat terminoren inguruan eta analisi horietatik ondorioztatutakoa azalduko dugu bukaeran.

5.1 Erroreen distribuzioa HTML corpus orokorrean

5.1.1 Errore guztiak batera

Corpusak azaldu ditugunean, komentatu dugu HTML corpus orokorra dela aztertu dugun corpusik handiena. 2000 dokumentuz osatutako corpusa da, eta 5.1 irudian ikus daiteke nolakoa den errore-tasa corpus horretan. Formari dagokionez, irudi hau konparagarria da 2.1 irudiarekin (ikus 2.6 atala), baina balioei dagokionez grafikoak ezberdinak dira: 2.1 irudian, errore-tasa kalkulatuta dago 1000 tokeneko, eta guk, berriz, 100 tokeneko kalkulatu dugu. Horrek adierazten du euskarazko corpusaren errore-tasa, oro har, dezente altuagoa dela, baina hori ez da harritzekoa bi hizkuntzen egoera soziolinguistikoa kontuan hartzen bada. Dena den, ingelesan gertatzen den antzera, corpusaren dokumentuen hiru laurdenak baino gehiagok errore-tasa txikia daukate, %5 baino txikiagoa hain zuzen.

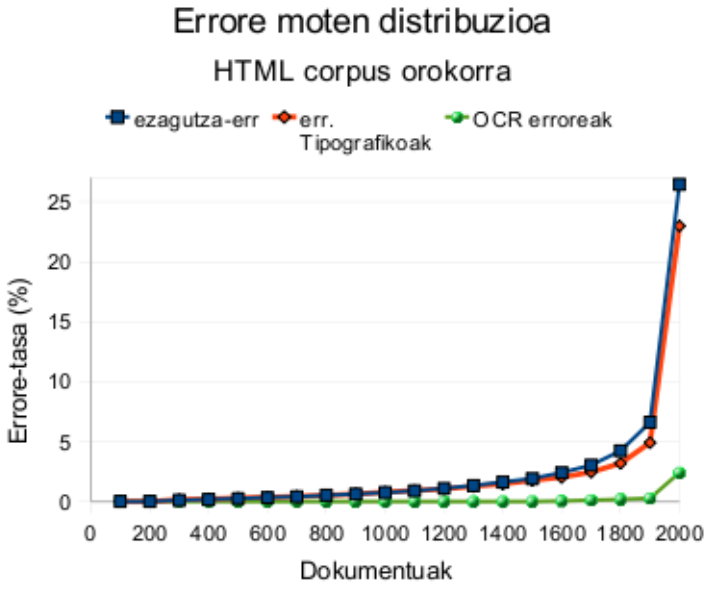
Dokumentu guztien errore-tasaren batez bestekoa eginez gero, lortzen den balioa 3,09 da. Erreferentziazko lanean egiten duten moduan, dokumentu txarrenak kentzen baditugu ikusi ahal izango dugu zenbateko eragina duten batez bestekoan dokumentu horiek. Hala, dokumentuen % 10 txarrenak kentzen baditugu, batez bestekoa 1,89ra jaisten da, eta % 20 txarrenak kenduz gero berriz, batez bestekoa 1,44ra jaisten da.



Irudia 5.1: HTML corpus orokorreko dokumentuen errore-tasa. Kontuan izan errore-tasa 100 tokeneko adierazten dela.

5.1.2 Errore mota bakoitza

HTML corpuseko hiru errore moten distribuzioa 5.2 irudian ageri da. Oso nabaria da OCR erroreak oso gutxi direla beste biekin konparatuta: ezagutzakoak eta tipografikoak. Beste bi horien artean, ordea, aldea ez da inportantea eta antzeko distribuzioa dute corpusean. Konparatzen baditugu grafikoko emaitzak (Ringlstetter et al., 2006) lanaren emaitzekin (ikus 2.6 atala), alde handia sumatzen dugu errore tipografikoen eta ezagutza-erroreen balioei dagokienez. Haiek diotenez, ingeleserako erroreen % 93 errore tipografikoak dira. Gure kasuan, berriz, errore tipografikoak eta ezagutza-erroreak antzeko balioetan daude. Esan dugu lehen, guk lehentasuna eman diegula ezagutza-erroreen identifikazioari, eta horregatik, token akastun bat ezagutza-errore bati badagokio, ez da izango errore tipografikoa ere. Dirudienez, haiek ez dute horrelako banaketarik egin eta horregatik da hain altua errore tipografikoen portzentajea. Bestalde, OCR erroreen tasari begira, argi geratzen da HTML corpusean oso gutxi direla mota horretakoak. Ikusi beharko dugu ea PDF formatuko corpusean gehiago ote diren.

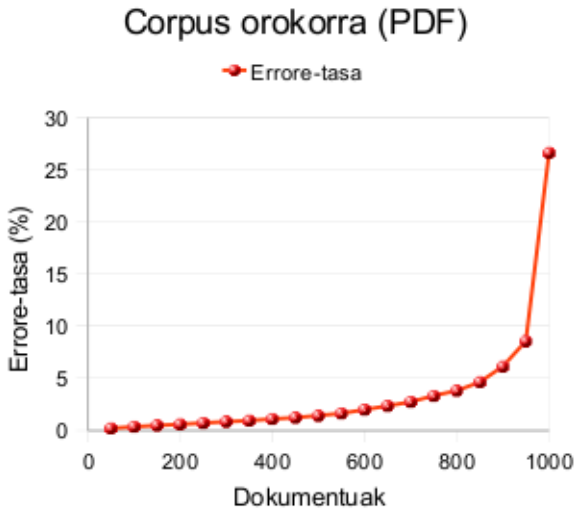


Irudia 5.2: HTML corpus orokorreko dokumentuen errore-tasa, motaren arabera. Oso agerikoa da OCR errore kopurua oso txikia dela besteekin konparatuta.

5.2 Erroreen distribuzioa PDF corpus orokorrean

5.2.1 Errore guztiak batera

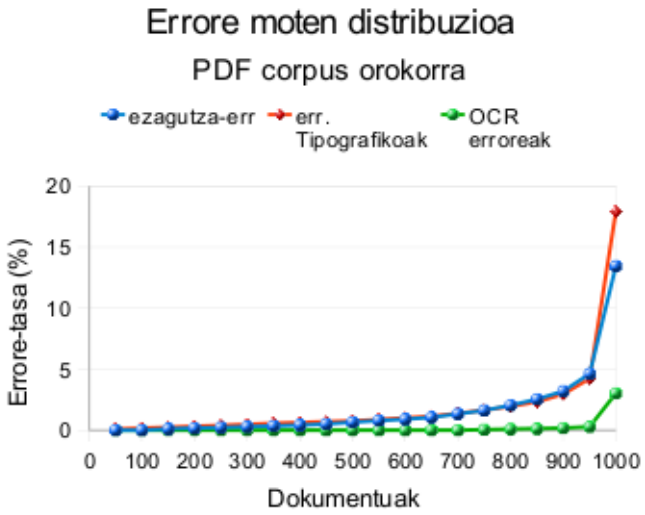
Aztertu dugun PDF corpusak 1000 dokumentu ditu, edozein gaikoak. 5.3 irudian ikus daiteke errore-tasaren balioa corpus horretan. Formari dagokionez, HTML corpusaren antzekoa da baina balioak txikiagoak dira. Hori hobeto ikusten da batez bestekoak kalkulatzen badira. Dokumentu guztien errore-tasaren batez bestekoa 2,62 da. Aurrekoan bezala, kentzen baditugu eskuin aldeko dokumenturik txarrenak, balio hori jaisten da. Dokumentuen % 10 txarrenak kenduta, batez bestekoa 1,75era jaisten da, eta % 20 txarrenak kenduz gero, 1,37ra. HTML eta PDF corpusen batez bestekoak konparatuz gero ondorioztatzen da PDF corpusean errore gutxiago daudela eta emaitza hori bat dator ingeleserako (eta alemanerako) lortu dituzten emaitzekin (ikus 2.7.1 atala).



Irudia 5.3: PDF corpus orokorreko dokumentuen errore-tasa. Kontuan izan errore-tasa 100 tokeneko adierazten dela.

5.2.2 Errore mota bakoitza

Hiru errore moten distribuzioa PDF corpusean 5.4 irudian ageri da. Ikusten denez, ez da gertatu guk esperokoa: OCR motako erroreak oso gutxi dira corpus honetan ere. Dirudienez, web-ean ez dago dokumentu asko euskaraz eta OCR aplikazioen bitartez sortua, edo ez behintzat zuzendu gabekoa. Horixe da emaitzetatik ondoriozta dezakeguna, behintzat. Bestalde, beste bi errore motei dagokienez, biek dituzte berriro antzeko balioak, grafikoan bata bestearen gainean baitaude ia beti. Konparatzen badugu grafikoa HTML corpusekoarekin (5.2 irudia) antzekoak dira: HTML corpusean nabaritzen da ezagutza-erroreak



Irudia 5.4: PDF corpus orokorreko errore mota bakoitzaren distribuzioa. Kontuan izan errore-tasa 100 tokeneko adierazten dela.

tipografikoak baino pixka gehiago direla beti, eta PDF corpusean, berriz, bi errore-motak balio bertsuetan daude bukaeran izan ezik.

5.3 Erroreen distribuzioa bi corpus orokorretan: laburbilpena

Bi corpus orokorretan maizen azaldu diren ezagutza-erroreak eta errore tipografikoak zerrendatu ditugu 5.1 taulan. Bertan ikusten denez, ezagutza-erroreak antzekoak dira bi corpusetan, 15 horien artean 8 errepikatzen baitira (letra beltzez ageri dira errepikatutakoak). Errore tipografikoetan, berriz, ez da horrelakorik ikusten nahiz eta baten bat errepikatzen den.

OCR erroreak askoz gutxiago dira, grafikoetan ongi ikusi den moduan, eta gainera, gaindetekzioaren arazoa suertatzen dela ikusi ahal izan dugu maizen suertatzen direnen analisia egitean, hots, identifikatzen dira OCR erregelekin, agian benetan OCR erroreak ez direnak. Gainera, hainbat kasutan OCR erregelak erregela tipografikoen kasu partikularrak dira eta horregatik, hainbat errore bi eratara identifikatzen dira. Esaterako, *kriston** terminoa identifikatzen da, bai OCR errore gisa, bai errore tipografiko gisa. Lehenengo kasuan, *kristori* terminoari lotzen zaio (*n* letra *ri* bihurtu da). Bigarren kasuan, *kristok*, *kristoz* edo *kristo*, termino zuzenen erroretzat har daiteke (guztiak 1eko distantziara). Seguraski ez da ez bat ez bestea eta ezagutza-erroretzat hartu beharko litzateke, baina une honetan ezagutza-erroreak identifikatzen dituen transduktoreak ez du horrelako erregelarik. OCR errore gutxi identifikatzen direnez, eskuz analizatu ditugu horrelako erroreak dokumentu batzuetan, kopuru altuena duten dokumentuetan hain zuzen, eta analisi horren arabera aurreko ondorio berara iristen gara: dirudienez, gure corpusetan ez dago OCR

HTML ezagutza-err	**PDF ezagutza-err**	**HTML tipografikoak**	**PDF tipografikoak**
bainan	**batzu**	bloga	**sobre**
bezela	**batetan**	blogean	haratago
batetan	**bainan**	haratago	hamaseib
batzutan	**batzutan**	**sobre**	hamabostm
batzu	**euskera**	eguin	**todos**
euskera	social	blogak	langue
diran	hamabostko	zitun	version
hortan	basque	kriston	estud
zeozer	handia	daben	tiempo
emaiten	lengua	**todos**	urria-abendua
gustora	**alternatiba**	anhitz	omen-en
alternatiba	vasca	bethi	siglo
gustoko	**hortan**	tiene	trabajo
geiago	euskeraz	dagoan	hitzordena
hontan	**bezela**	urthe	mismo

Taula 5.1: Bi corpus orokorretan maizen ageri diren ezagutza-erroreak eta errore tipografikoak. Mota bakoitzeko 15 errore zerrendatzen dira, eta bi corpusetan errepikatzen direnak letra lodian ageri dira.

aplikazioetatik sortutako dokumenturik, edo ez behitzat zuzendu gabekoak. Errore batzuk badirudite OCR sortuak: *lkasiak** (*ikasiak*), *duguia** (*dugula*), *sisternatikoki** (*sistematikoki*)... Baina beste asko zalantzazkoak dira eta askotan errore tipografiko sinpleagoak dirudite nahiz eta testuingurua ikusi gabe ezin ziurtatu horrelakorik. Halakoak dira, adibidez, *ingurun** eta *ordun** terminoak. OCR erregelekin *ingururi* eta *orduri* lotzen zaizkie hurrenez hurren, baina agian *inguruan* eta *orduan* terminoei dagozkie. Beraz, OCR gutxi dago dokumentuetan eta gainera hala identifikatu den guztia ezin da ziurtatu ongi dagoenik.

Ondorioz, OCR erroreen azterketa sakonagoa egin beharko litzateke OCR aplikazioak sortutako dokumentuetan oinarrituta. Horrela doitu ahal izango zen une honetan daukagun transduktorearen erregelak, zehatzagoa izateko errore horien detekzioan. Etorkizuneko lanetako geratzen da azterketa hori.

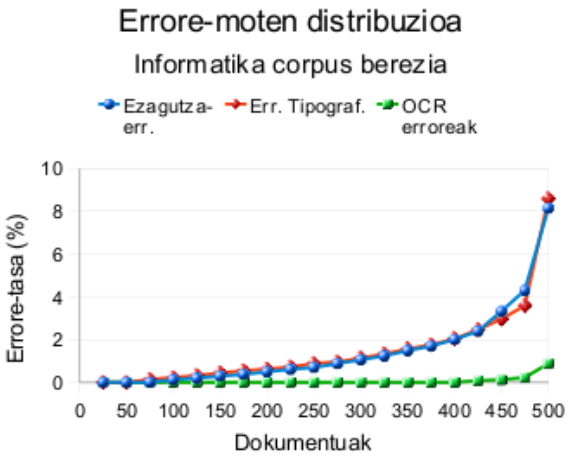
5.4 Erroreen distribuzioa corpus espezifikoetan

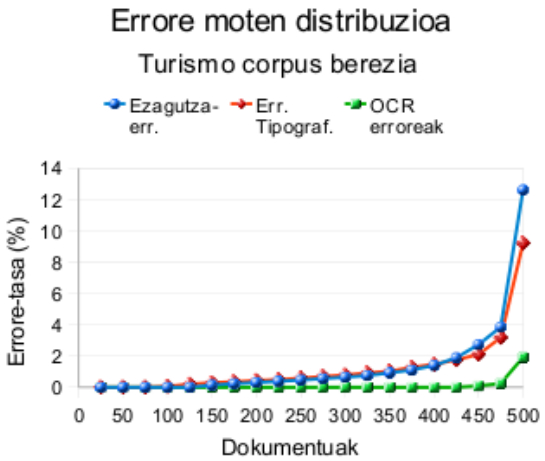
Hiru corpus espezifiko ditugu, eta horien batez-besteko errore-tasa kalkulatuz gero, hauek dira lortzen ditugun emaitzak: *Informatika* corpusaren errore-tasa, 2,43; *Turismo* corpusaren errore-tasa, 1,88 eta *Bioteknologia* corpusaren errore-tasa, 1,48. Corpus guztietan, beraz, HTML corpusaren baino tasa baxuagoa daukagu eta hori ez da (Ringlstetter et al., 2006) lanean lortutakoa. Haiek ingeleseko azterketan kontrakoa lortzen zuten: corpus bere-

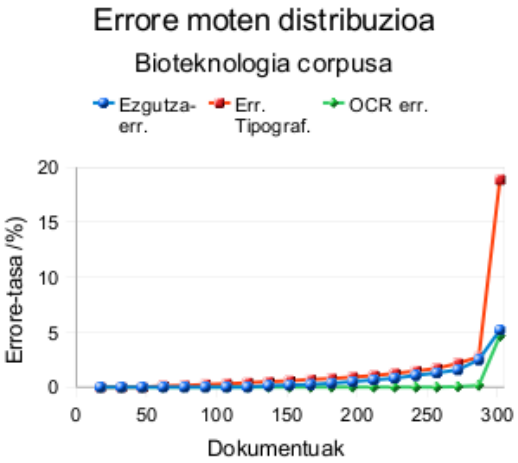
zi bat izan ezik, Neurologiakoa, beste guztietan errore-tasa altuagoa zen HTML corpusean baino[13]. Haien emaitzak 2.7 atalean laburbildu ditugu. Hiru corpusen artean, tasa baxuena Bioteknologiako corpusekoa da. Kontuan izan behar da corpus hau txikiagoa dela ez zirelako dokumentu gehiago aurkitu, gaia teknikoagoa baita eta ez da erraza dokumentu pila euskaraz topatzea. Tasaren arabera, badirudi dokumentu horiek hobeto idatzita daudela, besteak baino.

Hortik aurrera, corpus bakoitzaren errore-moten banaketa irudi baten bidez adierazten badugu, 5.5 irudiko grafikoak lortzen ditugu. Grafiko guztietan portaera bera azaltzen da: ezagutza-erroreak zein errore tipografikoak maila berean daude corpus guztietan, eta oso gutxi dira OCR erroreak beste biekin konparatuta.

[13]Tasak konparatzean, ez da ahaztu behar zenbakiak ez garela berdin ematen ari: haiek neurtzen dute beti tasa hori 1000 tokeneko eta guk 100 tokeneko.







Irudia 5.5: Erroreen distribuzioa corpus berezietan.

5.5 Detekzio-mugen estimazioa: gaindetekzioa eta azpidetekzioa

Erroreen detekzioaren mugak estimatu nahi ditugu atal honetan eta horretarako HTML corpus orokorrarekin lortutako emaitzak hartu ditugu kontuan, hori baitugu corpusik handiena: 2000 dokumentu eta 3,2 milioi hitz. Ezin ditugu muga bi horien balio zehatzak kalkulatu corpusa oso handia delako, eta horregatik, erreferentziako lanean egiten duten moduan, estimazio bat egingo dugu: lagin bat hartu eta bertan zer gertatzen den eskuz aztertu horrekin estimatzeko zenbatekoak diren bi mugak.

5.5.1 Gaindetekzioa

Esperokoa da identifikatutako erroreen artean okerrak egotea. Oker diogunean ez gara pentsatzen ari OCR errore bat ezagutza-errore gisa identifikatzea. Ez, oker diogunean adierazi nahi dugu posible dela benetako erroreak ez direnak errore gisa identifikatzea eta atal honetan hori maiz gertatzen ote den jakin nahi dugu. Horretarako, guztiz beharrezkoa da eskuzko analisi bat egitea identifikatutako erroreen artean. Analisi hori egin baino lehen, HTML corpus orokorrean identifikatutako errore guztiekin (edozein motakoak) zerrenda bakar bat egin dugu. Zerrenda hori maiztasunaren arabera ordenatu dugu eta horrela 32.668 token ezberdin dituen zerrenda lortu dugu. Horietatik, 8.309 dira corpusean behin baino gehiago agertzen direnak, hau da, ia laurdena; bi aldiz baino gehiago ageri direnak hartuz gero, zerrendak 4.201 token ditu. Dena den, 4.000 inguru horiek askotxo direnez eskuz analizatzeko, beste irizpide hau jarraitu dugu: eskuz aztertu errore guztien % 15a estaltzen dituzten tokenak, eta gero, gainontzekoen artean, aukeratu beste 100 token zoriz. Erroreen % 15a estaltzeko, maiztasun altueneko 249 erroreak aztertu ditugu. Gehiegitxo dira guztiak hemen zerrendatzeko, baina nolakoak diren ikusi ahal izateko, maiztasun altueneko 20 erroreak kopiatu ditugu 5.2 taulan. Gainontzeko %85 erroreen artean, 100 aukeratu ditugu zoriz eta guztira, beraz, 349 hitz analizatu ditugu eskuz. Hauek dira analisi horren emaitzak:

1. Zerrendan ageri dira neologismo gisa kontsidera daitezkeen tokenak eta oker identifikatzen ditugunak errore gisa. Adibiderik nabarmenena *blog* hitzari dagokio. 5.2 Taularen 7. lerroan *bloga* tokena ageri da eta 9. lerroan *blogean*. *Blog* hitza ez dagoenez euskarazko hiztegian, hasierako urratsean token horiek ez dira estandartzat hartzen eta gero, erroreak identifikatzeko garaian, errore tipografiko gisa identifikatzen dira: *bloka* eta *blokean* hitzen "mutazio" gisa. Lehen 249 erroreen artean, *blog* hitza beste hainbat aldiz ageri da: *blogak* (65 aldiz), *blogaren* (28 aldiz), *blogen* (26 aldiz), *blogari* (21 aldiz) ... Antzeko kasuen bila, ez dugu antzekorik aurkitu analizatu ditugun 349 tokenen artean, nahiz eta zalantzan izan garen bi kasuetan: *linux* eta *nanohodiak* ageri dira identifikatutako errore gisa, baina kasu horietan errorea ez da suertatzen hitzak hiztegian ez daudelako. *linux* tokenak letra larriz hasi behar du ongi egoteko definitu den lexikoaren arabera, hau da, *Linux* behar du; eta *nanohodiak* tokenak marratxoa behar du erabili dugun euskara estandarreko transduktoreak onartzeko, hau da, *nano-hodiak* behar du. Beraz, horiek ez dira zehatz-mehatz *blog*

Token erroreduna	Maizt. absolutua	Err. portzentaje metatua
bainan	263	0,42
bezela	251	0,81
batetan	243	1,20
batzutan	200	1,51
batzu	170	1,78
euskera	164	2,04
bloga	146	2,27
diran	140	2,50
blogean	130	2,70
haratago	121	2,89
hortan	101	3,05
zeozer	91	3,20
emaiten	90	3,34
gustora	88	3,48
alternatiba	87	3,62
gustoko	86	3,75
geiago	85	3,89
hontan	81	4,01
haundia	79	4,14
pixkat	73	4,25

Taula 5.2: Maiztasun altueneko 20 erroreak. Azkeneko zutabean adierazten da metatutako portzentajea zenbatekoa den.

adibide bezalakoak, nahiz eta kontsidera zitekeen marratxoa aukerazkoa izatea *mikro*, *nano* eta antzeko aurrizki teknikoekin.

Laburbiltzeko, analizatutako 349 tokenen artean, neologismoei 7 token dagozkie, hau da, % 2. Interesgarria izan daiteke horiekin zerrenda bat egitea IXA taldean kontuan har ditzaten eztabaidatzeko hitz horiek lexiko estandarraren barruan sartu behar diren.

2. Hainbat kasutan, errore gisa identifikatutako tokenak hitz zuzenak dira beste hizkuntzetan, baina testuingurua analizatu gabe ezin da jakin token horiek beste hizkuntzetako hitzei dagozkien, edo euskaraz gaizki idatziak dauden hitzei. Adibide batzuk emango ditugu. Identifikatutako 349 tokenen artean daude, adibidez: *sobre*, *contra*, *lengua*, *vasco*, *tiene*, *social*, *castellano*, *hacer*, *basque*... Denak dira hitz zuzenak beste hizkuntzetan, gaztelaniaz edo frantsesez, baina gure prozesuan errore mota baten bat gisa identifikatzen ditugu, gure transduktoreen arabera posible delako hitz estandar zuzenak token horietan bihurtzea: *kobre*, *kontra*, *lehengoa*, *basko*, *aiene*, *sozial*, *Castellano*, *hasiera*, *Basque*...

   Zenbakiak ematearren, token horiek zenbatu ditugu eta 42 dira, hau da, analizatu-

takoen % 12,03. Zalantzazkoa da token horiek horrela tratatu behar diren ala ez. Agian, web-etik dokumentuak jasotzen direnean, garbiketa prozesua gehiago doitu behar da, ez uzteko beste hizkuntzetako atalak corpusaren barruan. Dena den, hori erabakitzeko analisi sakonagoa egin beharko litzateke ikusteko non azaltzen diren hitz horiek, ea beste hizkuntzetako pasarteak dituzten dokumentuetan dauden, edo zer gertatzen den. Agian, ez litzateke interesgarria halako dokumentuak corpusaren barruan uztea. Edozein kasutan, hitz horiek filtratu nahi badira erroreen analisira ez iristeko, ziurtatu egin behar da benetan beste hizkuntzei dagozkiela, eta hori dokumentu osoa edukita egin behar da.

3. Azkenik, eskuz analizatutako 349 horien artean euskalkien adierazgarri diren hainbat token aurkitu ditugu, hala nola: *emaiten*, *daben*, *emoten*, *bethi*, *dogun*, *degun*, *dabela*... Horiek zuzenak dira euskalki jakin baten barruan, baina ez ordea estandarraren barruan eta horregatik iritsi dira zerrenda horretara. 5.3 taulan euskalkien adierazgarri diren hitzak zerrendatu ditugu, eta haien ondoan zenbat dokumentutan ageri diren. Azkeneko zutabe hori interesgarria iruditzen zaigu ikusteko aztertzeko ea euskalkiekin lotzen diren hitzak dokumentu gutxiagotan biltzen diren.

Token erroreduna	**Maiztasun absolutua**	**Zenbat dok.**
emaiten	90	35
daben	52	23
emoten	49	28
bethi	35	14
urthe	33	8
dogun	33	20
degun	33	11
dabela	32	16
bekhatu	28	5
gitxi	25	14
bategaz	22	14
dozue	21	11
doguz	21	11
dabez	19	10

Taula 5.3: Euskalkien adierazgarri kontsideratu ditugun hitzak, haien maiztasun absolutua HTML corpus orokorrean, eta zenbat dokumentuetan ageri diren.

Hori agerian uzteko, 5.4 taula prestatu dugu, non antzeko maiztasuna duten beste 5 errore ageri diren eta bakoitza zenbat dokumentutan. Bi tauletako dokumentu kopurua ikusirik, agerikoa da bigarren taulako adibideak asko gehiago sakabanatzen direla corpusean: 35/65; 23/40; 28/41.... Ez bada nahi euskalkien barruan zuzenak diren hitzak errore gisa tratatzea, zer edo zer egin behar da analisi-prozesua hasi baino lehen. Interesgarria litzateke tresnaren bat izatea dokumentu bat euskalki bat

Token erroreduna	Maiztasun absolutua	Zenbat dok.
zeozer	91	65
kitto	50	40
laister	49	41
somatzen	35	32
segundu	34	19

Taula 5.4: Errore "arruntak", euskalkiekin lotura berezirik ez dutenak. Ondoan, haien maiztasun absolutua ageri da eta zenbat dokumentuetan ageri dira.

erabiliz idatzita dagoen detektatzeko. Hizkuntza identifikatzeko tresna landuta dago IXA taldean, *LangId* tresna, baina orain planteatzen dena euskalkiaren identifikazioa da. Bide horretatik, posible izango litzateke errore-analisia beste abiapuntu batetik hasi, eta ez euskara ez estandarraren abiapuntutik.

Euskalkien mugak lausoak dira eta hitz batzuekin ez dago garbi euskalki baten adierazgarri diren edo ahozko dialektakalak diren. Horien artean ditugu, adibidez: leike, holan, halan, hunen... Hitz horiek dokumentua herri-mailan idatzita dagoenaren adierazgarri izan daitezke, eta agian, maila horretan, ez genituzke errore gisa kontsideratuko. Horretarako, dokumentuaren maila hasieran detektatuko beharko genuke.

Guztira, euskalkietako hitzak zein dialektalak batera zenbatuta, 36 hitz ditugu analizatutako 349 horien artean, hau da, % 10,3.

Aurreko kopuru guztiak laburbiltzeko 5.5 taulara jo daiteke. Kasu guztiak batuz gero, gaindetekzioaren balioa %24 da. Gogoan izan, euskara estandarraren ikuspuntutik, gaindetekzioa ez dela hain altua. Balio hori kalkultazeko, hitz dialektal guztiek gaindetekzio gisa kontsideratu ditugu baina ez dira gaindetekzioa euskara estandarrari begira.

Token multzoa	Neolog.	Beste hizkuntz.	Euskalk. + Dialekt.	Gaindetek.
% 15 (249)	7	38	35	% 32,13
100 zoriz	0	4	1	% 5
Guztira	7	42	36	% 24,36

Taula 5.5: Eskuz analizatutakoen artean aurkitu ditugu neologismoak, beste hizkuntzetako hitzak eta dialektalak gehi euskalkietakoak. Azkeneko zutabean gaindetekzioaren balioa ageri da horiek guztiak gaindetekzio gisa hartuta, eztabaidagarria izan daitekeena.

5.5.2 Azpidetekzioa

Esan dugunez, azpidetekzioa suertatzen da errore bat dagoenean eta gu identifikatzen ez dugunean. Hori zenbat aldiz gertatzen den estimatzeko, gaindetekzioan egin dugun antzeko analisia egin dugu. Orain, aztertu behar duguna da nolakoak diren ez estandarrak diren hitzak baina identifikatu ez ditugunak, hau da, hiru motatako batean sartu ez ditugunak. Corpuseko dokumentu guztien halako hitzak zerrenda bakar batean bildu ditugu eta zerrenda hori maiztasunaren arabera ordenatu dugu. Lortu dugun zerrendak 17.228 hitz ezberdin ditu. Horietatik, 3.516 gutxienez bi aldiz ageri dira corpusean, hots, % 20a; bi aldiz baino gehiago ageri direnak hartuz gero, zerrendak 1.541 hitz ditu. Horiek ere asko direnez, lehendabizi eskuz analizatuko ditugu maiztasunari begira zerrendaren %10 estaltzen dutenak. Horiek 155 hitz dira. Zenbat daude horien artean identifikatu beharko genituzkeenak eta ez ditugunak identifikatzen? Horietako hitz asko beste hizkuntzetako hitzak dira, eta ongi dago horiek ez identifikatzea, hori gaindetekzioa izango bailitzateke. Baina badira hainbat hitz, euskarazko hitzen itxura edo euskarazko hitzen mutazio gisa sailkatuko genituzkeenak, hala nola (parentesi artean hitzaren maiztasun absolutua corpusean): *dirade* (36), *formakuntza* (33), *kannabis* (31), *guciac* (26), *hamaseiitag* (24), *hamaseith* (20), *flandriarren* (15), *espektatibak* (15) *zonbeit* (14)... Guztira, horrelako 35 token aurkitu ditugu 155 tokeneko zerrenda horretan. Denak ez dira mota berekoak eta sailkapen bat egiten saiatu gara:

1. Batzuk tokenizazio okerrarengatik edo bi hitzen elkarketagatik suertatzen dira: *hamaseiitag* (24), *hamaseigarrenorr* (16), *hamaseisparams* (12), *hamaseisignature* (12), *hamabosgarrenurtean*... Halako 18 kasu ditugu, eta horietako asko "hamasei" hitzaz hasten dira: *hamaseisparams*, *hamaseisignature*, *hamaseisigh*... Oso token bereziak direnez non gertatzen diren aztertu dugu, zer edo zer arraro egongo delakoan. Halaxe izan da. Aurreko 18 token horietako 12 dokumentu bakar batean daude eta guztiak berean[14], gainera . Hala, dokumentu horrekin gertatzen dena aztertu dugu, hasieratik. Elhuyarretik jaso dugun jatorrizko HTML dokumentua bistaratuta, badirudi dokumentuak blog batean duela jatorria. Gero azaldu diren hitz berezi horiek bilatuz gero, ez ditugu formato horretan aurkitzen. Hori bai, dokumentuan zehar oso karaktere-kate luzeak ageri dira argazkien edota bideoen erreferentzia diruditenak. Zertaz ari garen ulertzeko, hona hemen HTML dokumentu baten baten zati bat:

```
<br/>Amaitzeko, berenjenak haragiarekin beteko ditugu, gazta birrindua gainetik
botako diegu eta labean berriko sartuko ditugu gainerretzeko.
<br/>
<object class="BLOG_video_class" classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=6,0,40,0"
height="266" id="BLOG_video-f10eff5d76b0c5e6" width="320">
<param name="movie" value="http://www.blogger.com/img/videoplayer.swf?videoUrl=http
%3A%2F%2Fv1.nonxt3.googlevideo.com%2Fvideoplayback%3Fid%3Df10eff5d76b0c5e6%26itag%3D5
%26begin%3D0%26len%3D86400000%26app%3Dblogger%26et%3Dplay%26el%3DEMBEDDED
%26ip%3D0.0.0.0%26ipbits%3D0%26expire%3D1265307608%26sparams%3Did%252Citag
%252Cip%252Cipbits%252Cexpire%26signature%3D3709CDA704828789B43FC31DC930C80FDB4D1497
.4D09F0B5C839A0FA2F44AF282982631D85D4E5FF%26key%3Dck1&nogvlm=1&thumbnailUrl=
```

[14]Zehazki, dokumentuaren identifikadorea 191 da

```
http%3A%2F%2Fvideo.google.com%2FThumbnailServer2%3Fapp%3Dblogger%26contentid
%3Df10eff5d76b0c5e6%26offsetms%3D5000%26itag%3Dw320%26sigh
%3DmyAYWkVhurrLEwDEz8uwOD54zoU&messagesUrl=video.google.com%2FFlashUiStrings.
xlb%3Fframe%3Dflashstrings%26hl%3Den"/>
```

Ikusten denez, hasieran testua dago eta horrek ez du arazorik sortuko, baina testu horren atzetik datorren HTML zatia, ez du ongi filtratzen erabili dugun `w3m` aplikazioak (ikus 4 ataleko urratsak) eta hori exekutatu eta gero aurreko atalaren ordez dagoen testu hutsa hauxe da:

```
Amaitzeko, berenjenak haragiarekin beteko ditugu, gazta birrindua gainetik
botako diegu eta labean berriko sartuko ditugu gainerretzeko.
embed(http://www.blogger.com/img/videoplayer.swf?videoUrl=
http%3A%2F%2Fv1.nonxt3.googlevideo.com%2Fvideoplayback%3Fid%3Df10eff5d76b0c5e6
%26itag%3D5%26begin%3D0%26len%3D86400000%26app%3Dblogger%26et%3Dplay%26el
%3DEMBEDDED%26ip%3D0.0.0.0%26ipbits%3D0%26expire%3D1265307608%26sparams%3Did
%252Citag%252Cip%252Cipbits%252Cexpire%26signature%3D3709CDA704828789B43FC
31DC930C80FDB4D1497.4D09F0B5C839A0FA2F44AF282982631D85D4E5FF%26key%3Dck1
&nogvlm=1&thumbnailUrl=
http%3A%2F%2Fvideo.google.com%2FThumbnailServer2%3Fapp%3Dblogger%26contentid
%3Df10eff5d76b0c5e6%26offsetms%3D5000%26itag%3Dw320%26sigh%3DmyAYWkVhurrLEwDEz8uwOD54zoU
&messagesUrl=
video.google.com%2FFlashUiStrings.xlb%3Fframe%3Dflashstrings%26hl%3Den)
```

Beraz, hasierako etiketak ongi kendu ditu, baina karaktere kate luzearen ordez zaborra geratu da. Hori dela eta, prozesuaren hurrengo urratsean tokenizatzaileak hainbat karaktere interpretatzen ditu aurreko katean eta horren ondorioz ageri dira lehen aipatutako hitz arraro horiek: *embed*, *hamaseiitag*, *hamaseibegin*, *hamaseilen*...

Hori ez gertatzeko, beste filtroren egin beharko litzateke eta horretarako aurreikusten diren aukerak bat baino gehiago dira: (1) filtro bat "a priori" pasa, halako kateak dituzten dokumentuak corpusetik kentzeko edo kate horiek dokumentutik kentzeko; (2) filtro bat "a posteriori" pasa eta dokumentu batean sailkatu ez diren errore asko aurkituz gero, dokumentu hori corpusetik kendu. Bigarren kasuan, noski, filtroa pasa eta gero, corpus berria prozesatu beharko litzateke estatistikak berregiteko.

2. Beste batzuk euskalkiak, aspaldiko euskara edo dialektalak dira: *dirade* (36), *guciac* (26), *zonbeit* (14), *baitirade* (13), *eztakit* (11), *doguzan* (11)... Zerrendatu ditugun 6 adibideak dira 35 horien artean horrelakoak.

3. Erabiltzen diren hitz batzuk, zuzenak dirudite euskaraz, baina ez dira estandarrak: *formakuntza* (33), *kannabis* (31), *kannabisa* (16), *flandriarren* (15), *espektatibak* (15), *ekintzaldi* (14), *nondik-norakoak* (12) .... Guztira, 11 hitz sartzen dira multzo horretan.

Zoriz aukeratutako beste 50 hitzen artean, 15 dira euskara kutsua dutenak baina ez ditugunak identifikatu, eta 15 horien artean ez dago kasu berririk. Tokenizazio okerreko kasu baten bat dago (*etaezabaketen*), baina, multzo horretan aurkitzen duguna, batez ere, dira:

1. Gaztelaniako hitz "euskeratuak", gainera, errore tipografikoekin. Adibidez: *prezedenterik*[15], *sobrakarga*, *lapidarioak*, *domestikazio*, *difizilenetarik*, *deskontestualizatua*...

2. Euskarazko hitzetatik abiatuta "asmatutako" termino berriak: *sortzezkotasunaren*, *orozapaltzailea*,...

3. Distantzia handiagoa duten errore tipografikoak izan daitezkeenak (batzuetan ezin da ziurtatu horrelakoak diren): *irakurketzeko*, *goiakaldean*, *areoplanua*, *zeikuna*

4. Ezagutza-errore bat baino gehiago jasotzen duten kasuak: *naspillatzen*.

Guztira beraz, eskuz analizatu ditugun 205 hitzen artean, 50 hitz azpidetekzioari dagozkio, hots, azpidetekzioa % 25 dela esan dezakegu. Azpidetekzioa jaisteko transduktoreak aldatzea plantea daiteke horrela errore gehiago identifikatzeko. Dena den, erabaki hori hartu baino lehen interesgarria izango litzateke kontuan hartzea errore horien maiztasuna eta baita aztertzea beste corpusetan ageri diren. Halere, errore gehiago identifikatzeko aukerak hauek izan daitezke:

- Bururatzen zaigun lehen aukera, sinplea da: errore tipografikoen distantzia handitu. Distantzia 2ra handituz gero identifikatuko ditugu, adibidez: *dirade* → *dira*, *hamaseith* → *hamasei*, *zonbeit* → *zenbait*. Eta 3ra handituz gero, gehiago detektatuko dira: *guciac* → *guztiak* ... Dena den, distantzia asko handitzen bada, hitzen bilaketa asko zabaltzen da eta probak egin beharko lirateke ongi doitzeko distantziaren kalkulua. Orain, kalkulu hori oso sinplea da: aukera guztiek (ordeztu, ezabatu, gehitu) kostu bera dute eta ez dugu aztertzen aldaketa bat baino gehiago duten aukerak.

- Ezagutza-erroreak detektatzen dituen transduktorea osa daiteke *formakuntza* → *formazio* bezalako kasuak detektatzeko. Maiz suertatzen bada horrelako aldaketa eta ez bakarrik hitz horretan, erabaki daiteke lexikoa zabaltzea adierazteko `+kuntza` atzizkia batzuetan ageri dela `+zio` atzizkiaren ordez. Bestalde, dialektalak edo euskalkienak detektatu nahi izanez gero, transduktore hori bera aldatzea izango litzateke egokiena. Adibidez, *eztakit* → *ez dakit* bihurtzeko, edo *nondik-norakoak* → *nondik norakoak* bihurtzeko. Horien antzekoa da *pixkat* hitza eta erabili dugun transduktoreak *pixka_bat* bi hitzen aldaera gisa detektatzen du.

- Errore batzuk sarrera lexikoen falta adierazten dute, nahiz eta eztabaidagarria izan kasu horiek guztiak hiztegian sartu behar diren. Goiko adibideen artean, horrelakoak kontsidera daitezke *kannabis* eta *flandriar* hitzak. Dena den, hori ez da arazoa detekzioari begira.

Azpidetekzioa jaisteko aukerak, beraz, badira, baina ez da ahaztu behar transduktoreen zabaltze horiek izan dezaketen kontrako efektua: gaindetekzioa areagotzea. Probak egin beharko lirateke eta emaitzak ongi neurtu erabaki okerrak ez hartzeko.

[15] adibide horietan ez da maiztasunik jarri oso baxua delako: 1 edo 2

6 Ondorioak eta etorkizuneko lanak

HAP masterraren tesi gisa aurkeztu dugun lan honen helburu nagusia web dokumentuak filtratzeko tresna automatiko bat lortzea zen, horrela kalitate minimoko web corpusak osatu ahal izateko. Kalitate minimo bat bermatuta egonez gero, hala osaturiko corpusetatik lor daitekeen informazioa egokiagoa izango da kasu gehienetan, batez ere, informazio linguistikoaren bila ari bagara. Bigarren helburua, erroreen tipologia estimatzea zen: zenbat errore dagoen mota bakoitzetik eta nola identifikatu mota bakoitza.

Helburu bi horiek lortu ditugu. Esperimentuen bidez ikusi dugu posible dela errore asko dituzten testuak detektatzea, eta posible dela baita errore mota bakoitzaren estimazioa egitea. Egin dugun lanaren metodologia finkatzeko, 2006an *Computational Linguistics* aldizkarian argitaratutako lan sakon bat hartu dugu erreferente gisa: (Ringlstetter et al., 2006) lana, eta horretan oinarrituta gure lanerako hainbat erabaki eta ezaugarri finkatu ditugu: corpus kopurua, corpusen ezaugarriak (tamaina, formatua, gaia ...).

Metodologiaz gain, ezinbestekoa zen datuak edukitzea errore ortografikoak analizatu ahal izateko, hots, beharrezkoak genituen euskarazko hainbat corpus web-etik eskuratuak. Horretarako, lankidetza ezin hobea izan dugu Elhuyarreko Igor Leturiarekin. Igor aditua da *Web as corpus* gaian euskaraz, eta bere aurreko lanetan (Leturia et al., 2008) osatuak zituen euskarazko hainbat corpusen lagina utzi digu guk eskatutako ezaugarrien arabera. Hori funtsezkoa izan da lan hau aurrera ateratzeko.

Metodologia finkatuta eta datuak izanik, tresnak behar genituen errore ortografikoen analisia egiteko eta tresna horiek transduktoreak izan dira. Erreferentziazko lanean errore-hiztegiak sortzen zituzten erroreak identifikatzeko, baina euskararen kasuan ezin zen horrela planteatu lana hizkuntzaren ezaugarriak direla eta. Hori dela eta, gure analisian erabili ditugun tresnak transduktoreak izan dira, egoera finituko teknologiak lortzen dituen oinarrizko tresnak. Aukera hori egokia da hizkuntza flexionatuekin lan eginez gero. Transduktore batzuk landuak ziren aurretik eta beste batzuk lan honetan zehar garatu ditugu, eta guztiak sortzeko `foma` tresna erabili dugu. Kode irekiko tresna da foma, 2009an argitaratua, eta bere egilearekin, Mans Hulden-ekin harremanetan jarri izan gara zalantza bat baino gehiago argitzeko. Errore-hiztegien ordez transduktoreak erabiltzeak hainbat onura izan ditu: (1) azkarragoa zen, ez baitziren zerrenda luzeak prestatu behar; (2) aurretik egindako hainbat lan berrerabili dugu; (3) garatu behar izan ditugun transduktoreak azkar garatu ahal izan ditugu aurreko esperientziagatik; (4) erroreen detekzioan estaldura handia lortu dugu, ez baitugu lan egin errore zerrenda mugatuekin.

Emaitzei begira, gure analisiaren emaitzak erreferentziazko lanekoekin konparatuz gero, ondorioa da errore kopurua adierazten duen grafikoaren forma oso antzekoa dela euskaraz eta ingelesez (ikus 2.1, 5.1 eta 5.3 irudiak), hots, dokumentu gehienak errore-tasa baxuetan daude, baina batzuk askoz tasa altuagoa dute. Dena den, tasa horren balioa aztertuz gero, ingelesaren balioa nabarmen txikiagoa da (haien tasa 1.000 tokeneko kalkulatzen dute, eta guk 100 tokeneko): alde batetik, gure metodoak errore gehiago identifikatzen ditu, eta bestetik, ingelesa eta euskararen egoera ez da konparagarria, euskararen estandarizatze prozesua berria baita oraindik. Horrez gain, gure lanean hartu ditugun erabaki batzuk, ingelesez hartutakoen berdinak izan dira, eta agian hori hobeto doitu beharko litzateke.

Esaterako, ingeleseko analisian hitzen luzera minimoa 5 letretan jarri zuten. Guk erabaki bera hartu genuen, baina kontuan izanik euskara eranskaria dela, hitzen luzera handiagoa da batez beste, eta horrek ere errore gehiago detektatzera eraman gaitu. Adibidez, *blog* hitza laburra da eta ez du luzera minimo hori, baina euskaraz ageri denean *blogak*, *blogean*... eta antzeko formak ageri dira, eta horiek errore gisa detektatzen dira. Beste luzera hartuz gero, agian ez genituzke horiek izango detektatukoen artean.

Bestalde, OCR erroreen inguruan lortutako emaitzen arabera eta 5.2.2 atalean esan dugunez, etorkizunerako OCR erroreen azterketa sakonagoa egin behar da OCR aplikazioek sortutako dokumentuetan oinarrituta, ez baitago batere garbi une honetan ditugun erregelak ongi funtzionatzen duten halako dokumentuetan.

Bukatzeko, lan honetan egin dugun eskuzko analisiak detektatutako hainbat errorekin gure susmoak berretsi ditu. Euskalkien erabilera ageri da web-eko testuetan, baina euskara estandarraren ikuspuntutik euskalkietako hitz asko errore gisa tratatzen dira. Horrek beste galdera sortzen digu: euskarazko testuen bila ari bagara web-ean, identifikatuko al ditugu esaterako, zuberotarrez idatzitakoak euskara gisa? Ezezkoan, datu asko galtzen ari gara. Oso lan interesgarria izango litzateke euskalkien identifikazioa egin ahal izatea, ez hartzeko erroretzat euskalkien hitzak. Gure ustez, gainera, euskalkien identifikatze horrek baliagarria izan liteke testuaren erregistroa identifikatzeko: erregistro zaindua, kaleko erregistroa ... Ildo horiek irekita dauden une honetan eta etorkizuneko lan interesgarritzat jotzen ditugu.

Etorkizunari begira, tesiaren erronka planteatzen zaigu eta lan honetan ikasitakoa oinarri sendoa izatea espero dugu aurrera jarraitu ahal izateko. Une honetan, transduktoreetan oinarrituta bi gai ditugu planteatuta:

- *"real word errors"* deritzon arazoaren identifikazioa eta zuzenketa. Arazo hori errore ortografiko baten ondorioz lortzen den hitza hizkuntzaren barruan dagoenean suertatzen da: *nahiz* $\leftrightarrow$ *naiz*, *esker* $\leftrightarrow$ *ezker*, *orri* $\leftrightarrow$ *horri*...

- aldaeren edota euskalkien azterketa eta eurei dagozkien paradigma morfologikoen inferentzia.

Aurrerantzean, beraz, *Hizkuntzaren Azterketa eta Prozesamendua* masterrarekin abian jarri dugun bidea jarraitzea espero dugu, horrela aurkeztu ahal izateko etorkizunean doktoretesia.

Erreferentziak

I. Alegria, I. Etxeberria, N. Ezeiza, eta M. Maritxalar. Morfología de estados finitos en software libre: aplicación al euskera1 finite state morphology using free software: application to basque. *Procesamiento del lenguaje natural*, 43:359–360, 2009.

Iñaki Alegria, Maxux Aranzabe, Nerea Ezeiza, Aitzol Ezeiza, eta Ruben Urizar. Using finite state technology in natural language processing of basque. In *Implementation and Application of Automata*, pages 1–12. Springer, 2002. URL `http://dx.doi.org/10.1007/3-540-36390-4_1`.

M. Baroni eta S. Bernardini. BootCaT: bootstrapping corpora and terms from the web. In *Proceedings of LREC 2004*, pages 1313–1316, 2004.

M. Hulden. Fast approximate string matching with finite automata. *Procesamiento del lenguaje natural*, 43:57–64, 2009a.

Mans Hulden. Foma: a finite-state compiler and library. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics: Demonstrations Session*, pages 29–32, Athens, Greece, 2009b. Association for Computational Linguistics. URL `http://portal.acm.org/citation.cfm?id=1609057`.

K. Kukich. Techniques for automatically correcting words in text. *ACM Computing Surveys (CSUR)*, 24(4):439, 1992.

I. Leturia, A. Gurrutxaga, N. Areta, I. Alegria, eta A. Ezeiza. EusBila, a search service designed for the agglutinative nature of basque. *Improving Non English Web Searching (iNEWS'07)*, page 47, 2007a.

I. Leturia, I. A Antton Gurrutxaga, eta A. Ezeiza. CorpEus, a 4web as corpus' tool designed for the agglutinative nature of basque. In *Building and Exploring Web Corpora: Proceedings of the 3rd Web as Corpus Workshop, incorporating Cleaneval*, page 69, 2007b.

I. Leturia, I. San Vicente, X. Saralegi, eta M. L. de Lacalle. Collecting Basque specialized corpora from the web: language-specific performance tweaks and improving topic precision. In *Workshop Programme*, volume 20, page 40, 2008.

P. Resnik eta N.A. Smith. The web as a parallel corpus. *Computational Linguistics-Special Issue on the Web as Corpus*, 29(3):349–380, 2003.

C. Ringlstetter, K. U Schulz, eta S. Mihov. Orthographic errors in web pages: Toward cleaner web corpora. *Computational Linguistics*, 32(3):295–340, 2006.

X. Saralegi eta I. Alegria. Similitud entre documentos multiling\ües de carácter científico-técnico en un entorno web. *Procesamiento del lenguaje natural*, (39):71, 2007.

S. Sharoff. Creating general-purpose corpora using automated search engine queries. *WaCky*, pages 63–98, 2006.

F. A Smadja eta K. R McKeown. Automatically extracting and representing collocations for language generation. In *Proceedings of the 28th annual meeting on Association for Computational Linguistics*, pages 252–259, 1990.

V. Sornlertlamvanich eta H. Tanaka. The automatic extraction of open compounds from text corpora. In *Proceedings of the 16th Conference on Computational Linguistics*, pages 1143–1146, 1996.

Christian Strohmaier, Christoph Ringlstetter, Klaus U. Schulz, eta Stoyan Mihov. A visual and interactive tool for optimizing lexical postcorrection of OCR results. In *Computer Vision and Pattern Recognition Workshop*, volume 3, page 32, Los Alamitos, CA, USA, 2003a. IEEE Computer Society. doi: http://doi.ieeecomputersociety.org/10.1109/CVPRW.2003.10031.

Christian M. Strohmaier, Christoph Ringlstetter, Klaus U. Schulz, eta Stoyan Mihov. Lexical postcorrection of OCR-Results: the web as a dynamic secondary dictionary? In *Document Analysis and Recognition, International Conference on*, volume 2, page 1133, Los Alamitos, CA, USA, 2003b. IEEE Computer Society. ISBN 0-7695-1960-1. doi: http://doi.ieeecomputersociety.org/10.1109/ICDAR.2003.1227833.