Universidad del País Vasco
Euskal Herriko Unibertsitatea

Informatika **F**akultatea
Informatika **I**ngeniaritzako **G**radua

Txioen itzulpen automatikoa gaztelaniatik euskarara

Gradu-amaierako Proiektua

(Konputagailuen Ingeniaritza)

2014 - azaroa

Egilea:

Urtzi UGARTE LARRARTE

Zuzendariak:

Iñaki ALEGRIA

Kepa SARASOLA

informatika fakultatea
facultad de informática

Laburpena

Gaur egun hain erabilia den Twitter aplikazioak badu euskarazko bertsioa, baina bertan idazten diren txioak oraindik ezin dira euskaraz atzitu, ez badira euskaraz itzultzen. Beste hizkuntza nagusietan ordea, Twitter aplikazio berak aukera ematen du hizkuntzaz aldatzeko txioen testua, bere elementu eta propietateak mantenduz. Hori da proiektu honen helburua. Txioetan idatzitako testua euskarara itzultzea, txioen formatua mantenduz.

Horretarako Twitterren API-a erabiliko dugu, txio eredu ezberdinak jasotzeko, eta ondoren bertako elementuak desberdindu eta itzultzeko, beharrezkoa den heinean. Besteak beste, traolak, erabiltzaileen aipamenak eta RT-eak (berTxioak) duten eragina testuan eta itzulpenean.

Txioen testua normalizatzeko aukera desberdinak inplementatu ditugu programan, Matxin web-zerbitzura bidali aurretik eta ondoren lortzen ditugun emaitzak aztertu.

Proiektu honek hasiera batetik erakarri zidan, Twitterren API-a erabiltzeko aukera eskaintzen baitu, eta gaur egun lan mundua ikusirik, honelako API ahaltsu bat erabiltzeak asko balio baitu.

Bestalde, Gradu amaierako azken urtean izan nuen itzultzaileen funtzionamenduekin lehenengoz harremana, eta ikasgaia aukerazkoa zenez, ez genuen nahi bezain guztia ikusi. Itzultzaileen munduak ere erakartzen zidanez hartu nuen proiektu hau. Amaitu ondoren, esan dezaket nire kuriositatea asetu dudala eta bai itzulpen automatiko bai Twitter API-ekin aplikazioak sortzeko gai naizela.

Nire iritzian erabilera handiko produktu bat lortu dugu proiektu honen bidez. Etorkizunari begira, baliabide gehiago inplementatzeko aukerarekin.

Gaien Aurkibidea

Irudi eta Taulen zerrenda

1.

Proiektuaren Helburu-Dokumentua

1.1 Proiektuaren helburuak

Proiektuaren helburua, itzultzaile bat izateaz aparte, itzultzeko eman daitezkeen aurre pausu ezberdinekin probak egiteko erremienta bat da. Horretarako aplikazioak ahalik eta aukera gehienak eskaintzea nahi izan dugu ondoren emaitzak ebaluatzeko.

Bere osagaiak honakoak dira:

- Twitter API-a erabiliz:
  - Txio bilatzaile bat ondorengo kategorietan bilatzeko aukerekin:
    - Aipamenak.
    - Traola edo Hashtag-ak.
    - Erabiltzaile jakin baten txioak.
    - Hitz edo testu jakin bat.
  - Txioei buruzko informazioa atera:
    - BerTxioa den kasuetan testu originala lortu.
  - Txioaren hizkuntza bereiztu
  - Itzuli nahi den hizkuntza eta nora itzuli nahi duen aukerak eman.
  - Txioak banaka edo bilaketaren emaitza denak itzultzeko aukera eman.

- Fitxategiak erabiliz:
  - Fitxategietako testua katalogatu Twitter aplikazioko txio bat izango balitz bezala.
  - Txioen buruzko informazioa atera.
  - Itzuli nahi den hizkuntzen kontrola eraman.
  - Emaitzen fitxategian gorde itzulitako testuak.
- Txioen itzulpena erderatik euskarara:
  - Txioan dauden elementu desberdinak bereiztu itzuli nahi den testutik eta gorde itzuli ondoren berriz idazteko.
  - Testu gordina tratatu eta formala ala informala den erabaki.
    - Txio formaletan normalizatzeko aukera eman, hau da, erderako hiztegian ez dauden hitzak ordezkatu daudenekin.

    - Txio informaletan hitzez-hitz begiratu hiztegian agertzen den ala ez eta proposamenetatik lehenengoaz ordezkatu.
- Testua sintagmetan zatitu eta itzultzailera zatika pasatzeko aukera.
- Itzulpenen emaitzak fitxategietan gordetzeko aukera inplementatu.

1.2 Emangarriak

Proiektu honen parte diren hiru emangarriak ondorengoak izango dira:

- ✓ **Txio itzultzailea**: Edo sortutako aplikazioa. Aplikazio hau bezeroaren esku utziko da eta baita sortzeko erabili den kodea ere. Proiektuaren zati hau Interneten jarriko da baita ere eskuragarri aplikazioa hobetu nahi duen guztientzat.
- ✓ **Dokumentazioa**: Proiektuko informazio guztia agertzen da bertan, plangintzatik, erabili diren erremientak, hartu diren erabakiak , diseinua eta lortutako emaitzak besteak beste.
- ✓ **Erabiltzailearen eskuliburua**: Aplikazioa nola erabili eta ematen dituen aukeren gida bat izango da emangarri hau, non bere instalaziorako behar diren programak eta nola exekutatu agertuko diren besteak beste.

Emangarriak 2015-ko Azaroak 3-an egon beharko dira eskuragarri bezeroarentzat, eta bi aste beranduago igoko dira Internetera.

1.3 Plangintza

Proiektuaren plangintza dokumentu honen amaierako eranskinean[1] aurkitu ahal izango duzu. Iruzkinean jartzeko arrazoia, honelako proiektu luze batean, plangintza asko aldatzen joan garela da, eta batez ere Scrum metodologia erabiltzearen eragina.

[1] [A Eranskina] : Proiektuaren Plangintza.

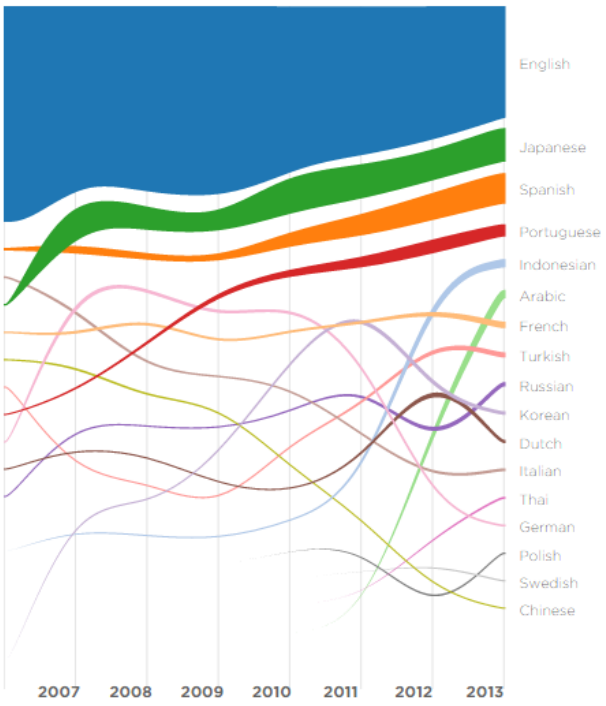
2.

Proiektuan erabilitako Tresnak

"Kapitulu honetan proiektuan zehar erabili ditugun hizkuntzaren prozesamendurako tresnak aurkeztuko dira. Hau da, erabilitako aplikazioak, liburutegiak, API-ak, baliabideak eta fitxategiak."

2.1 Sarrera

Twitter mundu mailako sare sozial bat da gaur egun, eta bertan mundu osoko erabiltzaileak aurkitu ditzakegu. Honek bertan sortzen diren mezu eta elkarrizketan hizkuntza barietate handi bat aurkitu dezakegu. Twitter aplikazioak gaur egun, 52 hizkuntza onartzen ditu, beraien artean euskara[2].



Irudia 1- Txioetan erabiltzen diren hizkuntzen portzentaiak[3]

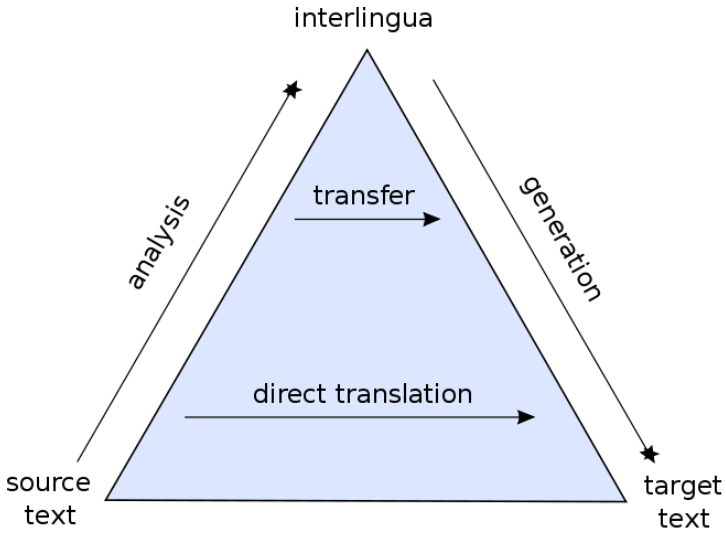
[2] Twitter Euskaraz [Link]
[3] Txioetan erabiltzen diren hizkuntzen portzentaiak **[Link]**

Honek Twitterren bezeroa euskaratzeko aukera ematen digu, baina jakina den moduan Twitter aplikazioak txioak idazteko momentuan ez du hizkuntzarik debekatzen, beraz edozein hizkuntzetako txioak irakurri ditzakegu Interneten. Hau dela eta Twitter berak Microsoftekin lanean hastean, Bing itzultzaile plataforma [4]integratu zuen txioetako testuak itzultzeko nahi den hizkuntzara. 2014ko abuztutik aurrera kendu egin duen aukera honek, Twitter itzultzaile automatiko bat gabe uzten du. Gure helburua itzultzaile automatiko hau sortzea da, eta Bingek lortu ez zuena lortzea; txioak euskarara itzultzea.

Itzulpen automatikoa (**MT**-*Machine translation*) iturburu-hizkuntzako testu batetik abiatuta helburu-hizkuntzako testu baliokide bat lortzea da makina bat erabiliz. Zailtasunak ordea gizaki baten mailara iritsiko den IA lortzea da. Besteak beste, ez baita erraza makina bati erakustea gizaki batek izan dezakeen ezagutza (testuingurua) ez eta erabakiak hartzeko gaitasuna (hitz anbiguoak).

Bi motako Itzulpen automatikoak aurki ditzakegu:

- Erregeletan oinarritutako itzulpen automatikoa [5] : (RBMT-Rule-based Machine Translation) Hiru estrategia bereiz daitezke ezagutza linguistikoaren arabera: itzulpen zuzena, transferentziazko metodoak eta interlingua bidezko itzulpena.



Irudia 2- Erregela bidezko sistemen sailkapena Vaquois-en triangeluak erakusten du [Link]

- Corpusetan oinarritutako itzulpen automatikoa [6] : Bere oinarrian aurrez egindako itzulpenak ditu. Metodo hauek eraikitzea erregeletan oinarritutakoak biano merkagoa izan daiteke kasu askotan, hizkuntzalarien menpekotasun handirik ez dagoelako; aitzitik, traba nabarmena da batez ere hizkuntza txikientzat corpus handiak biltzea.

[4] Bing Itzultzaile plataforma [Link]
[5] Machine Translation: Ingredients for Productive and Stable MT deployments [Link]
[6] Example-based Machine Translation [Link]

Twitterrek berriz berezko arazoak ditu, non bere 140 hizkiko txioak gaizki eratutako esaldiak ematen baititu emaitzatzat eta nahiz eta sintaxi sinplea izan, ez baitira estandarrak. Hau dela eta lexikoa izango da gure helburua, eta hitz ezezagunak tratatuko ditugu.

Oharra: Hitz Gakoak

Ondorengo txostenean agertuko diren hitz gakoak eta beraien azalpenak ondorengoak dira:

***Twitter**: Sare sozial bat da, non bertan erabiltzaileak beraien artean jarrai daitezkeen eta gehienez, 140 karaktereko txioak idatzi beraien oholean, non bere jarraitzaile denek irakurri ahalko duten.*

***Txioak**: Erabiltzaile jakin batek idatzitako mezu bat da, non gehienez 140 karaktere izan ditzakeen. Mezuan; aipamenak, traolak, berTxioak-ak eta URL-ak ager daitezke testu arruntaz gain.*

***Aipamenak**: Twitterreko erabiltzaileek beraien artean aipamenak egiteko @ bat idatziko dut erabiltzaile izenaren aurretik. Mezu hasieran idazten badute, mezu edo txio zuzen bat bihurtuko da, non aipatutako erabiltzaileak bakarrik irakurri ahalko duen. Testuan zehar aipatzen badio, bidali duenaren jarraitzaile guztiek ikusi ahalko dute.*

***Traolak**: Edo Hashtag ingeleraz. # ikurraren ondoren agertzen den hitza, (ez du balio bi hitz jartzea zuriuneaz baliatuta) gai bihurtuko du. Hau da, hitz hori etiketa bat bihurtuko da eta gai horri buruzko testuetan agertuko dira.*

***BerTxioak**: Beste erabiltzaile baten txioa zure oholean zabaldu nahi duzunean aukera hau erabiltzen da, non zure jarraitzaileei erakutsiko diezu jatorrizko txioa.*

2.2 Tresna orokorrak

Proiektua aurrera ateratzeko eta bere kalitatea bermatzeko, erabakiorrak dira lanerako aukeratzen ditugun tresnak. Hori dela eta proiektu honetarako, posible den heinean, bai emaitza bai prozesu osoan eskaintzen dituzten abantailak kontuan izan ditugu. Hona hemen erabilitako tresnak eta beraiek aukeratzeko arrazoiak.

2.2.1 Aplikazioaren garapenerako

- **Java**: Objektuei zuzendutako programazio lengoaia da. Berarekin lortzen dugun malgutasunak erakarri zigun, non sistema anitzetan erraz inplementatzeko taxutua baitago, iturburua helburuko plataformaren arabera era desberdinean konpilatu behar izan gabe etengabe.
- **JavaScript**: Interpretatutako programazio lengoaia bat da non web orrien garapenera zuzenduta dagoen. Gure aplikazioak bai Twitter eta bai Matxin itzultzaileekin jolasten duen, oso erabilgarria izango den lengoaia izango da.
- **Eclipse Kepler**: Kode irekiko software plataforma bat da. Javaz programatzeko garapen ingurune integratua (IDE) erabili dugu. Ematen dituen abantailak, bere malgutasuna izango da eta Javarekin duen integrazioa. Besteak beste, baita programarekin dugun trebantziak lagundu zuen software hau erabiltzeko erabakia hartzeko garaian.

2.2.2 Dokumentaziorako

- **Microsoft Word**: Microsoftek garatutako testu editore bat. Dokumentazioa idazteko erabili da eta programarekin dugun etxekotasunaz baliatuz aukeratu dugu, non maneiatzeko erraza eta testuen eta grafikoen inplementaziorako ematen dituen laguntzaz baliatuko garen.
- **Microsoft Excel**: Microsoftek garatutako kalkulu orrien garapenerako software bat da. Proiektuan eramandako orduen jarraipen bat egiteko erabiliko da batik bat.

2.3 API-ak (Application Programming Interface)

Gure proiektua aurrera atera dadin bi API nagusi ditugu esku artean: Twitterren API-a eta Matxin itzultzailearen API a. Bi hauek izango dira gure eguneroko ogia aplikazioa aurrera doan heinean. Twitterren API ak behar ditugun datuak hornituko dizkigu, bai fitxategietatik, bai txio bilatzailea erabiliz, eta ondoren Matxin itzultzailearen API aren esker gaztelaniatik euskarara itzuliko ditugu. Prozesu gehiago daude bi API hauek lotzeko, baina bi elementu hauek gabe ezin izango genuke aplikazioa martxan jarri.

2.3.1 Twitter-en API-a

2.3.1.1 – Sarrera:

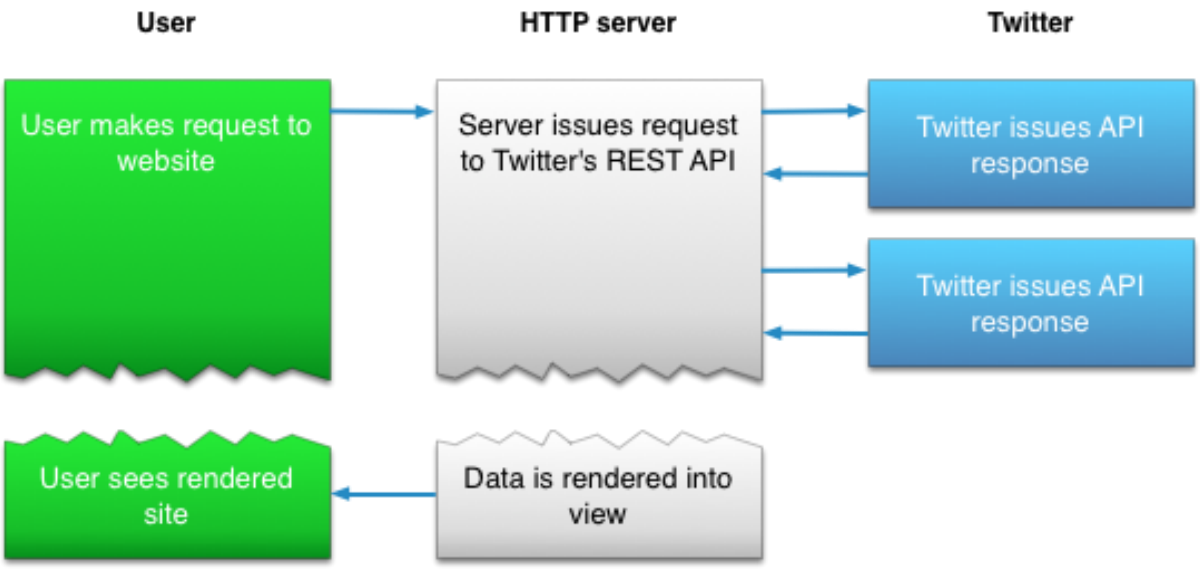
Twitter sare soziala 2006an Open Source edo Kode irekiko softwarez sortu zuten eta komunitatearekin bat egin nahietan kode irekiko bestelako softwarea ateratzen du. Twitter API publikoa horietako bat da, non Twitter zerbitzuetara atzi dezakegun guk sortutako aplikazioetatik .

Bi mota nagusi aurki ditzakegu Twitter zerbitzuarekin konektatzerako garaian: *Streaming* eta *REST API-ak*.

2.3.1.2 – Streaming/REST API:

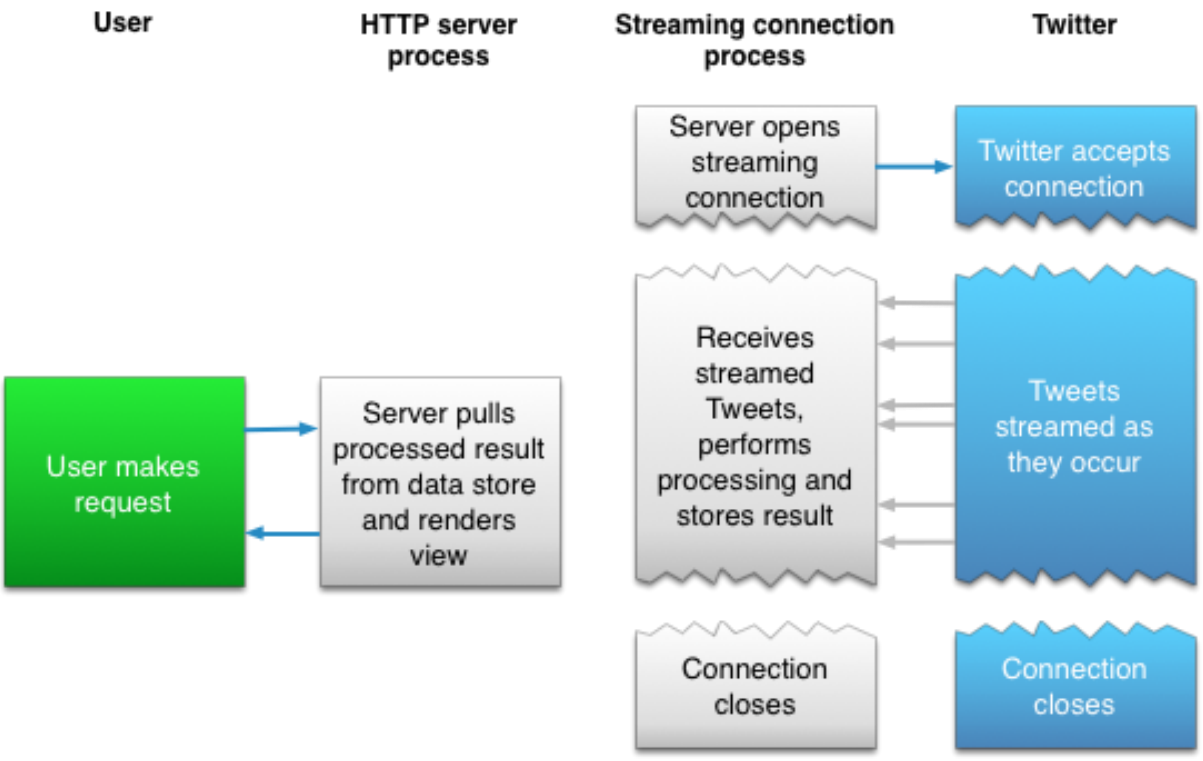
Bi aukerak aztertu behar ditugu gehien komeni zaiguna aukeratu aurretik. Hona hemen bi moduen azalpen labur bat:

- **Streaming API-a**: Tresna ahaltsua da hau. Twitter sare sozialaren pareko dabil. Bere helburua, besteak beste, datu meatzaritzarako produktu bat sortzean edo analitiketan interesatua bazaude laguntza ematea da. REST API-a baino aukera gehiago ematen ditu txio baten edo erabiltzailea baten gakoak bilatzean.



Irudia 3- REST API –aren interpretazio grafikoa [Link]

- **REST API-a**: Bigarren hau aurrekoaren bertsio sinplifikatuago bat da, non garatzaileari Twitterreko oholak, egoeren eguneraketak eta erabiltzaileen informazioa atzitzeko aukerak emango dizkion. Erremienta honen bidez garatzaileak Twitterren funtzio basikoak betetzeko aukerak ematen ditu eta aplikaziotik erabiltzaile bezala txioak sortu eta idazteko aukera ematen du besteak beste.



Irudia 4- Streaming API –a [Link]

API-en murriztapenak: kontuan izan behar ditugu bi API-ak mugatuak daudela, bata bestea baino gehiago.

- Ordu betean 150 baimendu gabeko eskaera egin daitezke eta 350 baimendutako eskaera.
- 250 zuzeneko mezu egunero gehienez.
- 2.400 txio argitaratu eguneko.

Gure aplikazioari dagokionez aurreneko murrizketa bakarrik da garrantzitsua. Aurrerago ikusiko dugun bezala, gure aplikazioa bai dago baimendua, beraz 350 eskaera egiteko gai izango gara orduko. Zenbaki handia ematen ez badu ere, gure aplikazioa zenbaki horretara iritxiko ez dela esan dezakegu.

REST API-ak baditu bere mugarriak, Stream API-arekin alderaturik. Gehienez 3.200 lortu ditzake bilaketak kontatzeko garaian, eta zenbaki hori 800 –era murrizten da bilaketak erakusterako garaian. Gure aplikaziori dagokionez, maximo bilaketek emandako emaitzak 30 izango direla erabaki zen, beraz murrizketa hauek ez dira arazo bat izango guretzat.

Bi metodoak oso erakargarriak iruditu zitzaizkigun arren, proiektuaren helburuak kontuan izanda Streaming API-aren beharrik egongo ez zela eta REST API-a aski izango zela gure betebeharrak betetzeko erabakia hartu zen.

Etorkizunari begira, Streaming API-a sartzeko aukera egongo da aldaketa handirik egin gabe, non bi API-ak nahastu bait daitezke aplikazio bakar batean.

Ondorioz, gure aplikazioan REST API-a erabiliko da, hori baita gure betebeharrentzako egokiena eta errazena, non dituen murrizketak aski diren eta arazorik gabe ibiliko garen. Lortutako emaitzak Streaming API-aren berdinak izango dira behar dugun kasuetarako.

2.3.1.3 – Baimentzea

Twitterrek bere API publikoa erabiltzeko baimendu beharra eskatzen du eta honekin murriztapenak handitzen zaizkigu kasu batzuetan. Bestalde, aplikazioak berak baimentzea eskatzen du ezer egin aurretik, Twitter enpresak jarraitzen duen politika eta besteak beste Twitter kontu batetik sortu eta eguneratzeko txioak.

Lehenengo urratsa aplikazioa erregistratzea izango da. Horretarako https://apps.twitter.com/ web orrira joan eta bertan aplikazioa sortu behar dugu.

Irudia 5 - Itzultzailea erregistratua

Aplikazioa erregistratzean bi token sortuko dizkigu eta hauekin gure aplikazioa baimentzea izango dugu kode bitartez.

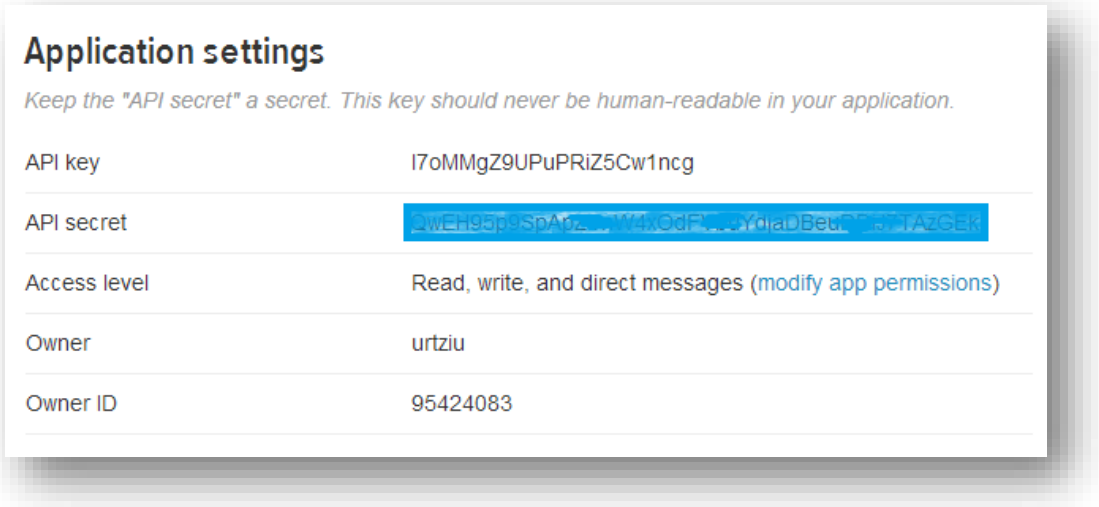
Oharra:

Gure kasuan, aplikazioa nire Twitter kontuarekin sortu nuen (@urtziu) 2013an. Ordutik, barne politika aldatu egin dute eta orain ez du uzten kontu propioek sarbide baimenak lehen bezala izaten, hau da, orain telefono kontu bat atxikitu behar duzu aplikazioak irakurri, idatzi eta zuzeneko mezuak bidaltzeko. Hau dela eta proiekturako sortutako Twitter kontuak (@TzoriItzultzail) ezin zituen lortu baimen denak. Ondorioz erabaki zen aplikazioa aurrera eramateko nire, kontu pertsonalarekin sortutako aplikazioaren tokenak erabiliko zirela.

Etorkizunean, hau aldatzeko arazorik ez dago. Sortu nahi duen kontu berritik aplikazioa erregistratu eta ematen dituen bi tokenak aldatu aplikazioaren kodean. Aurrerago ikusiko dugu non aldatu behar den.

Internet bidez honako bi gakoak lortu behar ditugu:

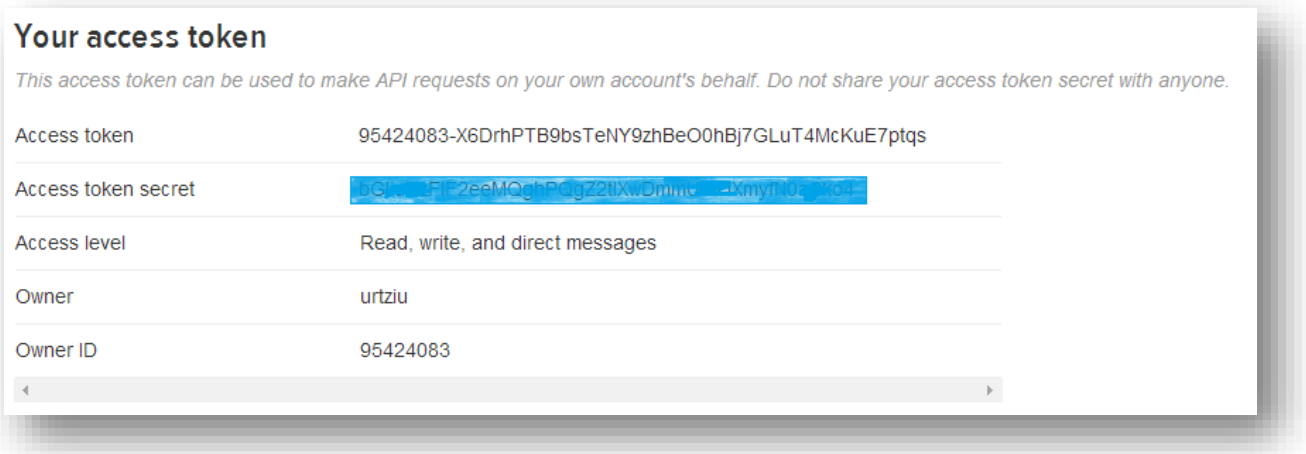
- Kontsumitzailearen gakoa: (*Consumer key*) '*API key*' izenez ere ezagutzen den gakoa da.
- Kontsumitzailearen sekretua : (*Consumer secret*) '*API secret*' izenez ere ezagutzen den bigarren gakoa da.



Irudia 6 - Bi gakoak

Gako hauekin erabiltzaile bezala bermatzen zara aplikazioan, hau da, identifikatu, eta bertan idatzi ditzakezun txioak zure izenean doaz.

Bi gako hauez aparte, bi token emango dizkigu baita '*apps.twitter*' web guneak, non aplikazioari baimena ematen dion Twitter API publikoari egindako eskaerak erabiltzen.



Irudia 7 - Bi tokenak

Gako eta Tokenen erabilera ere ez dago oso ondo agertua Twitterren dokumentazioan, baina proiektuari dagokionez kontuan izan behar dugu aplikazioak Twitter API erabiltzeko beharrezkoa den elementuak direla eta kodean gordeak egon behar dutela bai erabiltzailearen baimentzea eta baita aplikazioarenak.

2.3.2 Matxin API-a

2.3.2.1 – Sarrera:

Matxin itzultzailea [7] gaztelania-euskara itzulpenak egin ditzakeen itzultzaile automatikoa da, UPV/EHUko Donostiako Informatika Fakultateko IXA Taldeak garatutakoa. RMBT metodoan oinarritutako itzultzaile automatikoa da.

Wikipediako artikuluak dioen bezala honela funtzionatzen du:

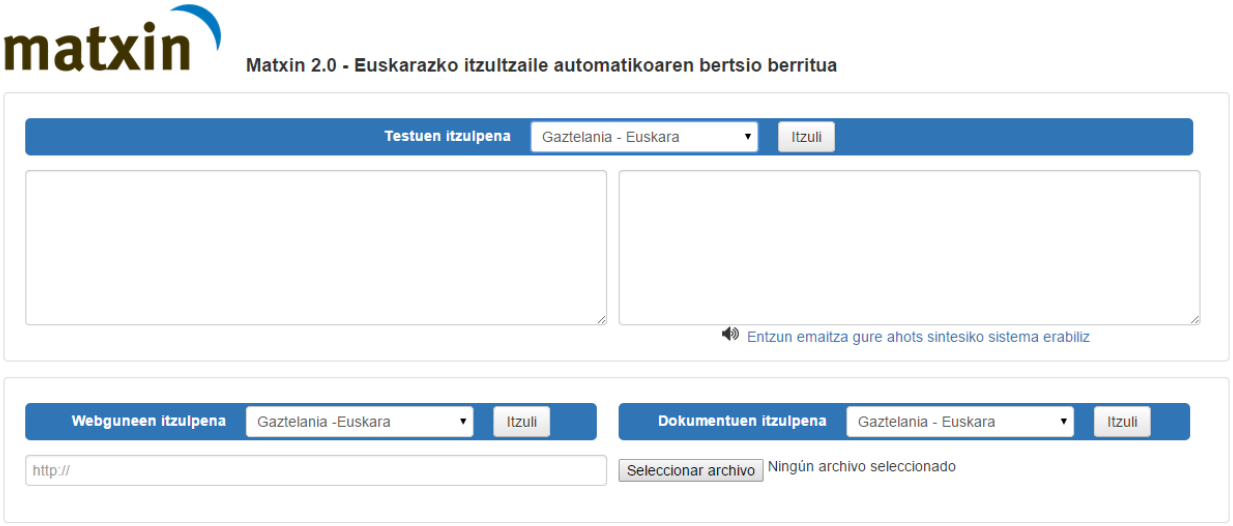
Matxin erabilera orokorreko itzulpen automatikoko sistema da, transferentzian oinarrituta. Hiru fasetan egiten du itzulpena: analisia, transferentzia eta sorkuntza. Analisi-fasean abiapuntu-testuaren analisi sintaktikoa lortzen du (espainierarako Freeling software libreko paketea erabiltzen da). Transferentzia lexikalean estaldura handiko hainbat hiztegi berrerabiliz eraikitako lexikoi elebiduna kontsultatzen da. Transferentzia estrukturalean, besteak beste, postposizioen desanbiguazioa eta aditz-kateen transferentzia burutzen da. Sorkuntza sintaktikoan hitzak eta sintagmak ordenatzen dira eta sorkuntza morfologikoan postposizioen informazioa sintagmako azken hitzari gehitzen zaio, hitzaren forma lortuz Morfeus euskararako prozesadore morfologikoarekin. Hainbat tresna eta baliabide linguistiko berrerabili direnez, bereziki landu da sistemako datuen, erregelen eta moduluen arteko datu-fluxuaren formatuen estandarizazioa.

Sistemaren arkitektura abiapuntu-hizkuntzatik independentea izateko diseinatuta dago eta inplementatuta dagoen Matxin 1.0 prototipoak espainieratik euskarara itzultzen du. Matxin 2.0 prototipoak aurreko lanaren hobekuntzak ditu eta, horrez gain, SMT sistema berria garatu da euskaratik espainierara itzultzeko.

Gure proiektuko bigarren API nagusia da, non Twitterretik jasotako txioak Matxinek itzuliko baititu.

Itzultzailea gaur egun Elhuyar hiztegiaren web gunean [8] aurki dezakezu besteak beste.

[7] Wikipedia: Matxin (Itzultzaile automatikoa) [Link]
[8] Matxin.elhuyar.org [Link]



Irudia 8 – Matxin itzultzailearen aplikazio grafikoaetako bat.

2.3.2.2 - Matxin Web zerbitzua:

Matxin itzultzaile automatikoa beharrezkoa genuenez bi aukera aztertu genituen; bata proiektuak barneratzea Matxin, eta bigarrena Interneten martxan dagoen Matxin web zerbitzua erabiltzea. Azken hau aukera hobea zela erabaki zen. Alde txarra da, Internetera konektatu behar dugu itzulpenaren emaitzak jasotzeko. Alde ona, berriz, proiektuaren pisua asko txikituko da Matxin osoa sartu gabe. Gainera, Twitter erabiltzeko Internetera sartu behar garenez, alde txarra desagertzen da.

Matxin web API [9]erabiltzeko SOAP (*Simple Object Access Protocol*) erabili dugu. SOAP-ek XML-z idatzitako mezuak elkar trukatu ditzake http protokoloa erabiliz, web zerbitzuak sortu ahal izanik. SOAP-ekin egitea zerbitzu sinple eta erabiltzeko erraza eskaintzea ahalbideratzen du, edozein programazio lengoaiarekin erabili daitekeena.

Baina gure proiektua Javan dago idatzia eta Matxinek Javarako dituen liburutegiak ez daudenez eguneratuak, eta dokumentazio falta dela eta, Perl bidez egitea erabaki zen.

Perl fitxategi bat atzituz, eta bertan SOAP erabiliz, Matxin web zerbitzua erabiltzea lortu ahal izango dugu.

[9] Matxin: an open-source transfer machine translation engine [Link]

```
#!/usr/bin/perl -w

#use SOAP::Lite +trace => [debug => sub{print "<pre>"; print@_;print "</pre>\n"}];
use SOAP::Lite;
use MIME::Base64;



#irakurri parametroak
my $testua = $ARGV[0];


$encoded = MIME::Base64::encode_base64($testua);
$response =  SOAP::Lite
    -> proxy('http://ixa2.si.ehu.es/matxin_zerb/translate_named.cgi')
    -> uri('MATXIN')
    -> EsEu($encoded)
    -> result;
print MIME::Base64::decode_base64($response);
```

Irudia 9 - Erabilitako Perl fitxategia

2.4 Liburutegiak

2.4.1 Twitter4j

Twitterren API-a atzitzeko Java hizkuntzan aukera ezberdinak ikusi genituen. Probak egin ondoren Twitter4j[10] liburutegia erabiltzea erabaki zen behar ditugun metodo guztiak baititu erabilgarri. Liburutegia ez da ofiziala, baina Twitterrek bere webgunean zabaltzen du Twitter4j izena Javarekin lotua dagoen liburutegi bakar bezala. Erabilerraza, Twitterrek gomendatua eta ondo dokumentatua dagoelako erabaki zen Twitter4j erabiltzea.

Arazorik ez da egon bere integrazioan baina egongo balitz alternatibak java-twitter [11] eta jtwitter[12] izango lirateke.

2.4.2 Jazzy SpellChecker

Jazzy[13] laburrean azalduta zuzentzaile ortografiko bat, zure Java proiektura gehitzeko aukera ematen duen liburutegi bat da. Gure kasuan Matxin itzultzailera testua bidali aurretik txio informaletan hitz bakoitza hiztegi batean parekatuko dugu eta ematen digun lehenengo proposamenaz ordezkatzeko besteak beste. Honekin bilatzen duguna da ea idatzitako zuzena den ala ez.

Zuzentzaile ortografiko gehiago ere badaude erabilgarri, beraien artean ezagunena Hunspell [14]. Google berak ere zuzentzaile ortografiko bat zabaldu zuen urte batzuk atzera, baina gaur egun jadanik ez dago eskuragarri.

[10] Twitter4j liburutegiaren web-gunea [Link]
[11] Java-Twitter liburutegiaren web-gunea [Link]
[12] JTwitter liburutegiaren web-gunea [Link]
[13] Jazzy SpellChecker web-gunea [Link]
[14] Hunspell web-gunea [Link]

3 .

Proiektua Diseinatzen

"Kapitulu honetan proiektua aurrera emateko erabili diren metodo eta hartutako erabakiak azalduko dira. Besteak beste, proiektuaren beharrak, Scrum modeloa…"

3.1 Proiektuaren planteamendua

Bezeroarekin lehenengo bilerak egitean, idea bat atera zen, proiektuaz zer bilatzen zen hitzegitean. Txioen itzultzaileaz gain, itzultzeko aukera ezberdinak emango

3.1.1 Itzultzaile baten beharrak ikusi eta aztertu

Bezeroarekin lehenengo bilerak egitean, idea bat atera zen, proiektuaz zer bilatzen zen hitzegitean. Txioen itzultzaileaz gain, itzultzeko aukera ezberdinak emango dituen aplikazio bat nahi genuen lortu. Hau da, hasiera batean ez zen adostu aplikazioak emango zituen aukerak zein izango ziren eta horrek proiektua diseinatzean eragin handia izan zuen.

Itzultzaileak aukera ezberdinak izateko gaur egungo itzultzaileei buruz irakurri eta aztertu nuenean eta hortik hainbat ideia sortu zitzaizkigun: testu *formalak eta informalak* bereiztea, *normalizatzaileak* sartzea testua txukuntzeko. Eta beste aldetik, Twitter berak eskaintzen dituen aukerak baliatu txioak testu arrunt bilakatzeko eta itzuli ondoren berriz ere *txio formatu bat* emateko.

Txio itzultzailea sortzeko beharrezkoak ziren pausuak ikusirik, eta etorkizun batean aldaketa gehiago eskaintzeko aukera dagoela ikusirik, proiektua ezin zuen diseinu lineal bat jarraitu. Softwarea garatzeko modelo asko daude, eta horietako bat, garapen azkarra deritzo. Proiektuak eskatzen dituen pausuak ikusirik, ez zen aski modelo hau jarraitzea, eta beraz Scrum[15] (azpi modeloa) erabiltzea erabaki genuen.

[15] SCRUM Development Process [Link]

3.1.2 Scrum modeloa

Software Garapen proiektuez ari bagara, proiektuen kudeaketa klasikoaren zikloa, "eskaintzaren itxiera – baldintzak zehaztea – diseinua – garapena – azken produktua entregatzea" ez da ikuspegi errealista bat egun bizi garen inguruan. Etengabe behar berriak sortzen dira, hasiera batean kontuan hartu ez diren funtzionalitateak, baldintzetan aldaketak, diseinua fintzea eta abar alde batera uzten dira "hasierako irismenetik kanpo". Hala ere, hau da bezeroak behar duena.

Arazo honi erantzuteko sortu dira Scrum [16]bezalako metodologia arinak[17], proiektuan sortzen diren aldaketei erantzuteko malgutasuna eta egokigarritasuna bermatzen dutenak. Hasieratik bezeroa kontuan hartzeari zuzenduta dauden diziplina anitzeko pertsona talde auto antolatuetan oinarritzen da metodologia hau. Bezeroak berehala ikusten du produktuaren bertsio bat, horrela, bere nahiak zeintzuk diren asmatzea errazagoa izanik. Taldekideak motibatuago eta inplikatuago daude proiektuaren azken helburuarekin eta hauen produktibitatea eta gogobetetzea areagotu egiten da.

Garapen azkarren barnean, Scrum metodologia ondorengo puntu hauetan laburbiltzen da beraz:

- ✓ "Areagotze" garapen estrategia erabiliko da, planifikatu eta exekutatu estrategiaren ordez; Non produktuaren funtsa hasieran egin eta ondoren bezeroaren ideak gehitzen joango diren.
- ✓ Emaitzaren kalitatea bermatzerako garaian erabilitako prozesuen kalitatean zentratu beharrean, proiektuko partaideen ezagutza taktikoa hobetzean zentratuko da.
- ✓ Garapeneko fase ezberdinak teilakatuz, eta ez utziz ataza bat amaitu arte itxaron eta ondoren hasi berri bat.

Puntu hauek gure proiektura itzultzen baditugu lortu duguna honakoa izango da besteak beste:

- ✓ Bezeroarekin bilera izan den bakoitzean aplikazioaren momentuko egoera erakutsi demoen bitartez eta hortik berak eskatutako berriketak gehitzen joan.
- ✓ Helburua ez denez jendeari saltzea produktua, garatuko den aplikazioa sortzeko erabilitako bideak ez dira izango hoberenak, baliteke kasu batzuetan okerreko bidetik ibiltzea baita ere, baina helburua ahalik eta itzulpen hoberenak egitea denez, balio gehiago du itzultzaileei buruzko jakintza lortzeak eta hortik aplikazioari aukera ezberdinak gehitzea.
- ✓ Garapeneko atazak teilakatu egingo dira beharrezkoa den oro, non itzultzailearen barnean, testuaren kodeketa egokia amaitu aurretik, normalizatzaileak sartzen ibiliko garen, beti ere kasu partikularrak itzultzeko gai izango den baina ez txio guztiak, eta horrela normalizatzaileak probatzea.

[16] Wikipedia: Scrum (Software Development) [Link]
[17] Scrum Is A Mayor Management Discovery – Artikulua [Link]

3.1.2.1 - Garapenean lortutako onurak eta esperientzia pertsonala:

Metodologia honek oso emaitza onak eman dizkit, batez ere proiektuan pertsona bakarrak egiten duenean lana eta proiektuaren emate data atzeratzen joan denean.

Bezeroari denbora gehiago eskainiko dugula produktuan esatean, elementu berriak sartu nahi izango dizkio eta Scrum metodologia horretan zentratzen da. Pertsona bakarra izatean, eta gure kasuan Gradu amaierako proiektu bat izatean, produktua egiteko emandako bideak kaotikoak izan dira momentu batzuetan, non inoiz eman edo ikasi gabeko ataza bat betetzean eman diren pausuak; lehenik informatu, ondoren adibideak ikusi, ondoren proiektura nola integratu, probak egin, eta proba gehiago egin direnez, momentu bat irizten da non garapenean esleitutakoa baina denbora gehiago eman dugun. Eta batez ere proba hauek edukiko duten eragina bai kodean eta bai aplikazioaren itxuran ez da egokiena izango.

Baina Scrum metodologia hortaz doa, emaitza ez da hori, baizik eta proba horiei ezker ikasi duguna eta etorkizunean emandako denbora ataza berri bati txikiagoa izango baita lortu ditugun jakintzekin.

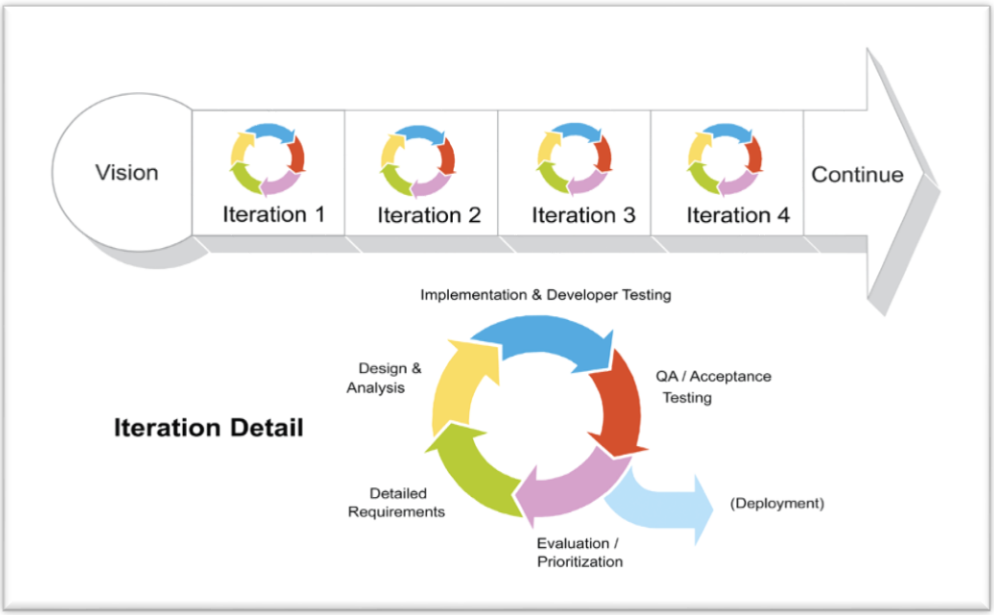
3.1.2.2 - Bezeroan ikusitako esperientzia :

Garapenerako, goian esan bezala, onura bikaina izan du metodologiak, baina bezeroaren gain izandako eraginak ere esateko asko du.

Bezero mota asko aurki ditzakegu, baina gure kasuan, bezeroekin izandako lehen bileretan ikusi genuen proiektuaren nondik norakoa, eta bezeroak ere ez baitzekien zer nahi zuen produktuaren azken emaitzan, metodologia hau aukeratu genuen.

Hortik aurrera txioen itzultzaile basikoa egin genuen eta bezeroarekin elkartzen nintzen bakoitzean demo txiki bat erakusten nion, bat, ikusteko nola zioan, proiektua ze egoeretan zegoen, eta bi, martxan ikustean, momentuan pentsatzen hasten baitzen zer gehiago egitea nahi zuen aplikazioak eta etorkizunean kokatzen zen, aplikazioa erabiltzen eta izango zituen arazo posibleak pentsatzen.

Nire iritzitan, bezeroak hasiera batean ideia gordin bat bakarrik baldin badu, hau da metodologia hoberenetariko bat, berak ere martxan ikusten baitu eta denbora ematen baitio garatzen duenari pausoka joan eta aldaketa txikiak inplementatzea denboran zehar.



Irudia 10 - Scrum metodologiaren interpretazio grafiko bat.

3.1.2.3 -Ondorioa :

Etorkizunean, beti ere bezeroaren eskariek ala uzten badute, **Scrum metodologia beti hartuko dut kontuan**, aukera asko ematen baitditu proiektua aurrera doan heinean, eta hasiera batetik, nahiz eta karga gehiena bertan egon, produktuak jadanik itxura bat izango du eta bezeroari erakusten joatean, ez dira gertatuko azken momentuko sorpresak.

3.1.3 Aplikazioa garatu aurretiko diseinu erabakiak

Buelta asko eman genituen datu eta moduluen arteko erlazioak nola jorratu erabakitzen. Proiektuak Scrum metodologia erabiltzeaz gain atzitzen dituen modeloen arteko erlazioak ez baitira linealak, eta aplikazioak ez du gaur egun geroz eta gutxiago erabiltzen den aplikazio hierarkiko baten planteamendua erabiltzen.

Gure proiektua kanpoko API eta web-zerbitzuz baliatzen da, baina ez ditu bere osotasun guztian eskaintzen. Horren ordez, gure helburuetarako beharrezkoak diren funtzioak edo moduluak bakarrik atzituko ditugu, bai pisua arintzeko eta bai bizkortasuna irabazteko.

Helburuetan argitu dugun bezala, aplikazioaren sarrerako datuak bi azpitalde nagusitan bereizi behar ditugu: Fitxategiak eta txio bilatzailea. Biek, itzuli behar ditugun txioak lortuko dizkigute. Fitxategien helburua da, txio liburutegi handiak irakurri eta normalizatze bat ala beste erabiliz, itzultzea,

eta txio bilatzailearekin lortu nahi dena berriz, kasu partikularrak edo txio berezien itzulpen hobeak lortzea. Biak Twitter API-a erabiliz lortuko dugun zerbait da.

Erabiltzaileari ordea hau dena ez zaio axola, berak aplikazioak eskainitako baliabideak erabiliz txio bat edo asko itzuliko ditu. Erabiltzaileak ez du Twitter API edo Matxin itzultzailearen berezitasunak jakin behar ezta hauek ematen dituzten mila aukera ezberdinei buruz.

Beraz, hiru atal nagusi identifikatzen ditugu:

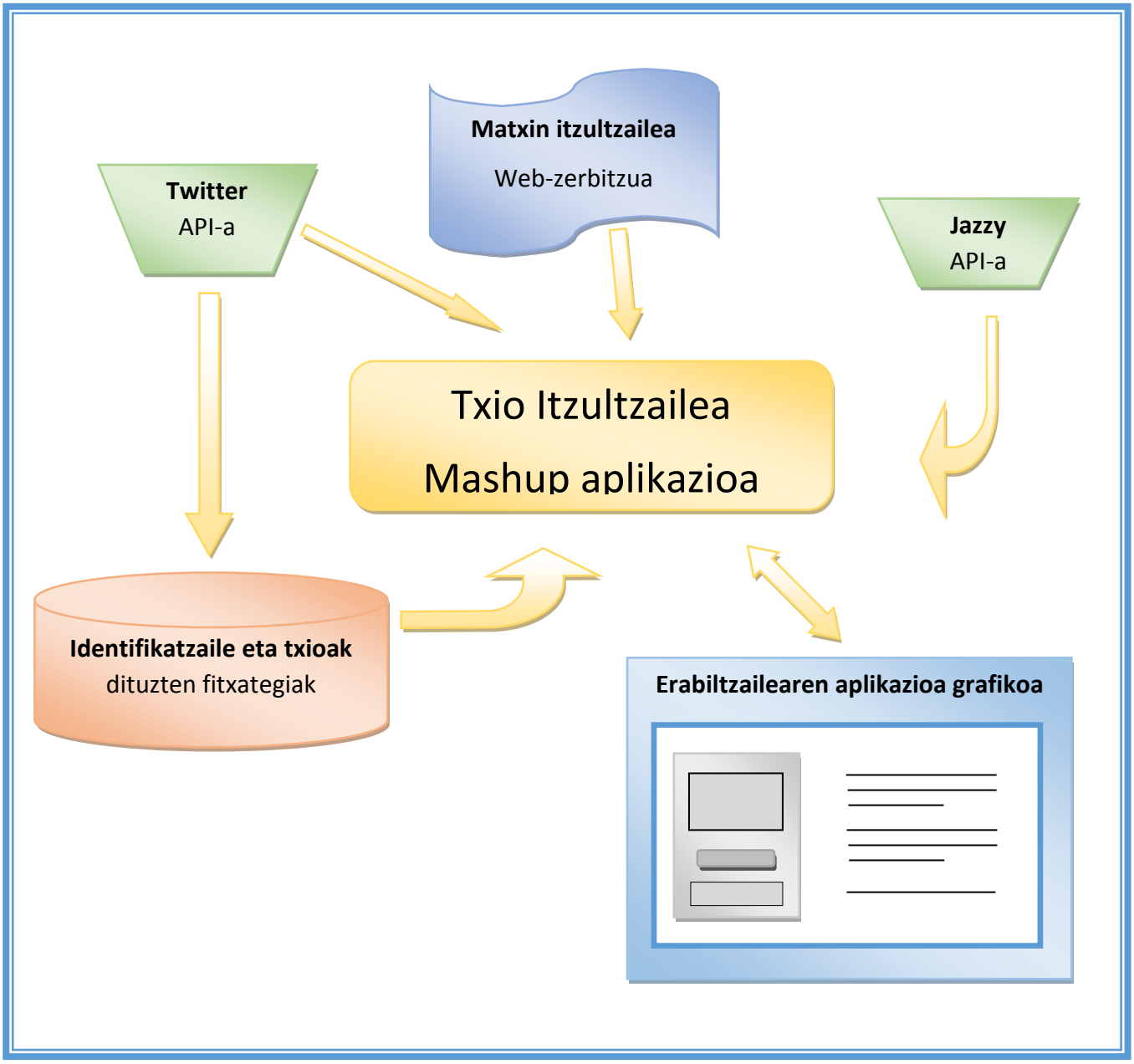
- Sarrera datuak, bai Twitter API-a erabiltzen duen txio bilatzailea edo fitxategi ezberdinak txioen identifikatzaileekin.
- Matxin web-zerbitzua, Twitter API-a eta Jazzy API-a gure beharretarako erabiltzen dituen aplikazioa.
- Erabiltzailearen interfazea, gure kasuan Javaz idatzitako aplikazio bat.

Atal nagusi hauek **Mashup** [18] garapen metodologiarekin bat datoz. Non iturri bat baino gehiagoko edukierak erabili eta nahasten dituen, zerbitzu berri eta sinpleago bat sortu eta ondoren interfaze grafiko bidez bistaratzeko. Metodo honi Agile Mashup [19]ere dei diezaiokegu, lortzen dituen printzipioak [20] direla medio.

Mashup garapen metodologia erabiltzean sarrera datuak datu baseetatik jasotzen dira orokorrean, nahiz eta metodologiak berak ez itxi biderik bestelako formatuei, gehien erabiltzen direnak datu baseak dira. Gure proiektuan aldiz, bezeroak hornitu dizkigu sarrera fitxategiak, beraien txio corpusak fitxategietan gordeak baitituzte.

Datu baseek eskaintzen dituzten baliabideez jakitunak gara, baina gure aplikazioak sarrerako datu edo irteerako datu bezala ez dugu aurre ikusten baliabide horiek erabiltzeko beharrik. Honekin esan nahi duguna da, datu baseak eta honelako estruktura edo moduluak ematen dizkiguten baliabideak gure proiektuak erabiltzen dituenerako gehiegi direla eta hauek inplementatzeak ez digula onura berezirik emango. Hau dela eta **datu baserik ez erabiltzea eta fitxategien bidez lan egingo dugula erabaki da.**

[18] Wikipedia: Mashup (Web application hybrid) [Link]
[19] Introducing an Agile Method for Enterprise Mash-Up Component Development [Link]
[20] Principles behind the Agile Manifesto[Link]



Irudia 11 – Proiektuaren Mashup metodoligiarekin lortutako eskema.

3.2 Fitxategiak eta hauen formatuak

Twitterren API-az aparte fitxategietatik txioak jasotzeko aukera ere badu aplikazioak Mashup metodologiaren barne, baina horretarako sarrera bezala erabiliko diren fitxategi hauen formatua diseinatu behar da. Bi aukera nagusi ditu gure aplikazioak:

3.2.1 Identifikatzaileen fitxategiak

Honako hauek .txt fitxategi arruntak dira. Bertan txioen datu base antzera, proba kasu asko gordeta izango ditugu. Fitxategi hauek honako formatua izango dute:

Id	Testua
• Twitterrek txioari emango dion identifikadorea izango da	• Bertan txioaren mezua agertuko da • Txio arrunten elementuak izan ditzake: @, # ...

- ✓ Identifikadorea: 18 karaktereko zenbaki bat izango da, non honekin txioa atzitu ahal izango dugun bere informazioa ateraz, non testuaz gain erabiltzailearen informazioa ere lortuko dugun.
- ✓ Testua: Testu arrunta da, txio batek izan ditzakeen elementu guztiekin, bai berTxioa, traolak, aipamenak, URL … Guk hau testu bezala gorde eta ondoren prozesatu egin beharko dugu behar bezala tratatu eta Matxin itzultzailera bidaltzeko.

```
318735263864070144      Siempre me pasa lo mismo.
318769856369553408      QUIERO SABER EL SECRETO DE EL HELADO!!!! UNIVERSO ESCUCHAME QUE LUEGO NO TE ENTERAS!!
318493746733932544      Ven que vamos hacer un pacto yo y tu sonrisa
319011023124307968      Yo me entierrro, me pierdo x la Tierra PASO DE LA SITUACIÓN
319092814313041920      Ni contigo, ni sin ti.
```

Irudia 12 – Identifikatzaileen fitxategi baten adibidea (.txt)

3.2.2 Txioen fitxategiak

Fitxategi hauek .csv formatukoak izango dira (*Comma-separated values*[21]), non bertan txioen jakin batzuen testua, erabiltzailearen izena eta txioen hizkuntza agertuko diren. Fitxategi hautako txioen testua kodetua dago UTF-8 formatuan.

Hizkuntza	Testua	Erabiltzailea
• Txioa idatzita dagoen hizkuntza agertuko da. "ES", hau da gaztelania hizkuntza bakarrik hartuko dugu kontuan gure proiektuan.	• Bertan txioen mezua agertuko da UTF-8 formatuan kodeatu. • Txio arrunten elementuak izan ditzake: @, #, ...	• Txioa idatzi duen erabiltzailearen izena agertuko da.

- ✓ Hizkuntza: "es" gaztelerarako, "ca" katalanez dagoenean, "en" ingeleraz, eta "Pt" portugesez denean. Identifikadore hauek ze hizkuntzetatik itzuli nahi dugun bereizteko erabiliko ditugu. Gure aplikazioak "es" motakoak bakarrik egingo ditu.
- ✓ Testua: Testu arrunta da, txio batek izan ditzakeen elementu guztiekin, bai berTxioa, traolak, aipamenak, URL … Guk hau testu bezala gorde eta ondoren prozesatu egin beharko dugu behar bezala tratatu eta Matxin itzultzailera bidaltzeko.
- ✓ Erabiltzailea: Txioa idatzi duen erabiltzailearen pantaila izena agertuko da bertan. Itzuli eta gero nork idatzi duen jakiteko erabiltzen da.

80	esSegunda parte. Ajax-BarÃƒÂ§a (2-0) http://t.co/zFItSUKyKQ #fcblive #fcbelperiodico_cas
81	esÃ‚Â¡Empieza la segunda parte! #FCBliveFCBarcelona_es
82	esRecorda que pots seguir la transmissiÃƒÂ³ del partit a travÃƒÂ©s de RÃƒÂ dio BarÃƒÂ§a: http://t.co/YC77eES7Pw #fcbliveFCBarcelona_cat
83	esRecuerda que puedes seguir la transmisiÃƒÂ³n del partido a travÃƒÂ©s de Radio BarÃƒÂ§a: http://t.co/gTve76AWLK #fcbliveFCBarcelona_es
84	enThe second half gets under way at the Amsterdam Arena, with Ajax leading 2-0 #fcbliveFCBarcelona

Irudia 13 - Txioen fitxategi baten adibidea (.csv)

Fitxategi honetako testua kodetua dator, hau da karaktere bereziak UTF-8 formatuan daude, beraz itzultzailera pasa aurretik kodeketaz aldatu beharko dugu.

[21] Wikipedia: Comma-separated values [Link]

3.2.3 *Irteerako fitxategiak*

Irteerako fitxategiak, Twitterren API-az bilatutako txioen itzulpenak gordetzen ditugun fitxategiak izango dira. Bi fitxategi izango ditugu: bata emaitza indibidualekin eta beste bilatu diren txio guztien itzulpen automatikoa. Bientzako formatua berdina izango da:

Bi lerro txio bakoitzeko, non lehenengoan testu originala egongo den itzuli nahi den hizkuntzan formatu honekin:

@Erabiltzailea	Testua
• Bertan mezua idatzi duenaren erabiltzaile izena egongo da.	• Itzuli nahi dugun testua.

Eta bigarren lerroan itzulpena, itzuli duen hizkuntzan, formatu honekin:

@Erabiltzailea	RT	Itzulpen testua	#	URL
•Txioaren erabiltzaile izena.erTxioa bada, itzulpena baino lehen RT karaktereak agertuko dira.	•BerTxioa bada mezu originala, itzulpen hasieran RT karaktereak agertuko dira. •Ondoren noren berTxioa den agertuko da @baten ondoren.	•Txio originalaren testua itzulita agertuko da	•Txio originalean agertzen ziren traolak agertuko dira mezuaren ondoren.	•Traolen antzera txio originalean agertzen ziren URL-ak atsikituko dira.

```
Originala: @Eslianuth01: Se colocan un reloj y no sirve.
Itzulpena: @Eslianuth01:  Erloju bat jartzen dute eta ez da zerbitzatzen.
Originala: @cortes_luis: RT @kristianCeleita: "No existe nada completamente equivocado en el mundo. Hasta un reloj parado consigue estar
Itzulpena: @cortes_luis:  RT @2 'Ezer ez da existitzen guztiz munduan huts egina. Erloju bat gelditu arrazoi izatea lortzen du birritan
Originala: @10zorijose: Eres arena de otro reloj
Itzulpena: @10zorijose:  Beste erloju bateko harea zara
Originala: @TxoriItzultzail: Hoy hace buen tiempo y pienso ir a la #playa con los amigos. #titzuli
Itzulpena: @TxoriItzultzail:  Denbora egin ona eta hondartzara joatea lagunekin pentsatzen dut gaur.   #playa #titzuli
```

Irudia 14 - Irteera fitxategi baten adibidea

4.

Aplikazioaren Garapena

"Kapitulu honetan aplikazioaren garapena azalduko da, eta diseinuan hartutako erabakiak izan duten eragina ikusiko dugu. Beti ere Scrum eta Mashup metodologiak jarraituz bezeroak txioak itzultzeko erabiltzen duen interfaze grafikoaren atzean, programa nola dabilen ikusiko dugu."

4.1 Algoritmo nagusia

Diseinuan ikusi dugu Mashup-ak bideratu digun eskema, non hiru atal nagusiak eta beraien arteko erlazioak agertzen diren. Eskema horri jarraituz garatu dugu aplikazioa, baina Scrum metodologia jarraituz, bezeroarekin egindako lehenengo bileretarako zatika eramatea erabaki zen, hau da, bileran, demo antzera, aplikazioak zer egin zezakeen erakutsi, eta horretarako zatika garatu dugu Mashup aplikazioa, alde berean erabiltzailearen interfaze grafikoaren lehenengo bertsioak eginez.

Lehenik eta behin sarrera datuak jasoko ditugu, txio bilatzailea erabiliz ala fitxategiak irakurriz. Aukera hori interfazeko menu nagusian agertuko da, ze hizkuntzetatik nora itzuli nahi den aukerarekin batera. Gogoratu gure kasuan gaztelaniatik euskarara egingo dugula.

Itzultzailerako txioak aukeratu ondoren, txioak tratatuko ditugu, bere elementuak bereizi eta gordetzeko, testua bakarrik itzuliko baitugu eta bere elementuak; traola, berTxioak edo URL-ak besteak beste alde batera utzi eta itzuli ondoren berriz ere eransteko.

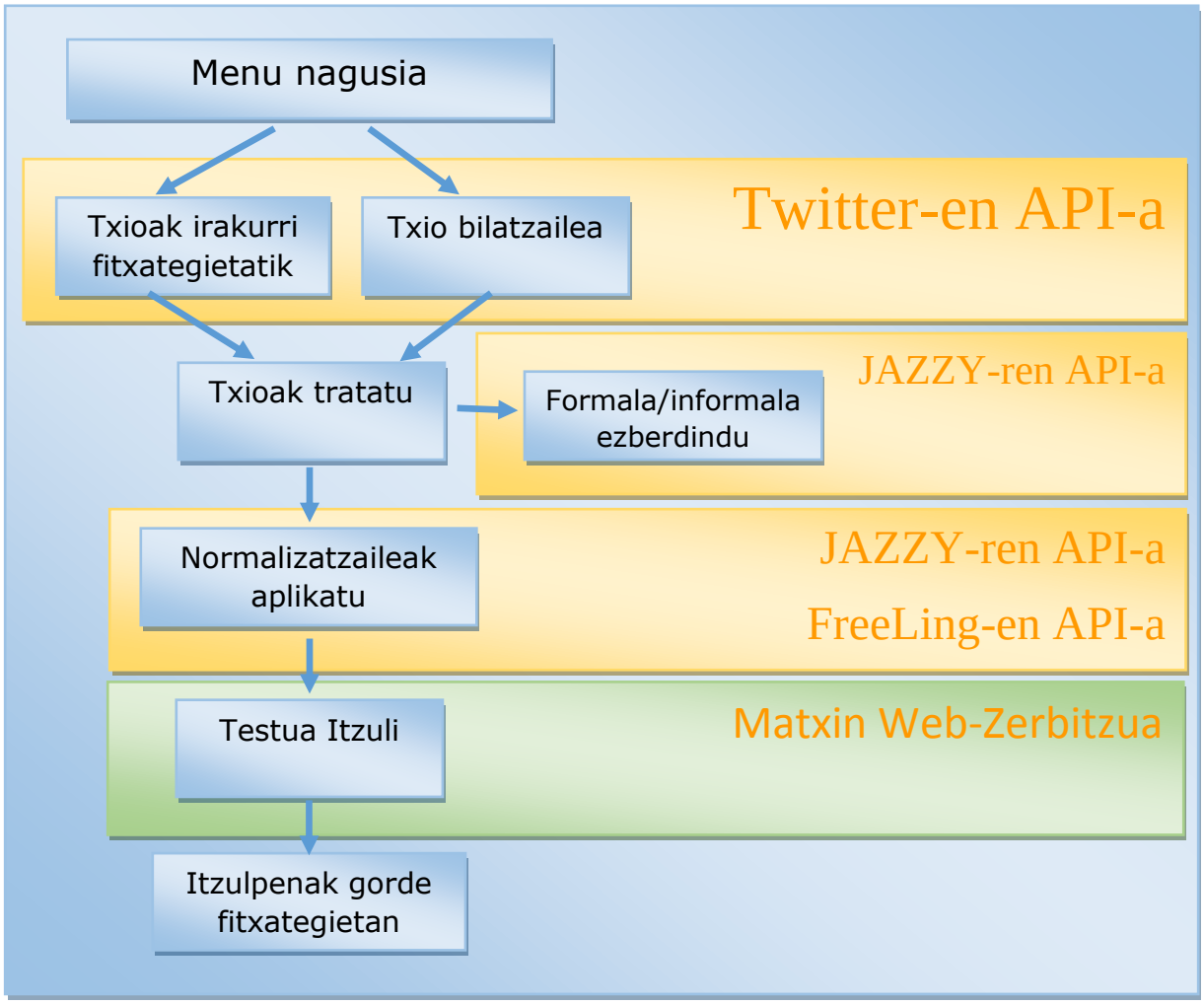
Txioetako testua lortu ondoren, itzulpen bat lortzeko lehen urratsa, testu formala ala informala den ezberdintzea izango da, biak ez baitira berdin tratatzen itzultzerako orduan.

Bigarren urratsa normalizatzaileak aplikatzea izango da, non gure aplikazioak bi aukera emango dituen; testu formaletan normalizatzaile arruntarekin edo FreeLing-ekin [22], edo informaletan, normalizatzaile arrunta edo hitzez hitzeko itzulpena.

Azkenik testua prest dagoenean Matxin itzultzaileari bidaliko diogu, web-zerbitzu bidez atzitzen duguna, eta itzulitakoa bezeroari erakutsiko diogu, irteerako fitxategi batean gordetzeko aukera emanez.

Urrats ezberdinak ikusi ondoren ordena berezi honetan lan egitea zela era eraginkorrena erabaki genuen, non besteak beste Twitter API-an jarriko dugun arreta, bera gabe ezin gaitugu itzultzailearen bestelako funtzioak ez probatu ez bezeroari erakutsi.

[22] FreeLing 3.1 – An Open Source Suite of Language Analyzers [Link]



Irudia 15 - Algoritmo nagusia eta atzitzen dituzten API eta web-zerbitzuak

4.2 Txioak bilatu Twitter API-az baliatuz

Sarrerako datuak jasotzeko lehenengo aukera txio bilatzaile bat da non Twitter API-a erabiliz, hitz gakoak, aipamenak, traola duten hitzak edo gaian eta erabiltzaile baten txioak bilatzeko aukera emango digun. Bilaketak, 15 emaitza itzuliko dizkigu gehienez, non bertan txioa nork idatzi duen eta ondoren txioaren edukia agertuko den. Txio bat aukeratu dezakegu informazio gehiago jakiteko bai erabiltzaileari buruz ala txioa argitaratu zen eguna eta ordua jakiteko besteak beste. Txio bat aukeratu ondoren itzultzeko prozesua hasiko dugu.

4.2.1 Twitter API-aren integrazioa

Twitter API-a, dokumentu honetan lehenago azaldu dugun bezala REST API-a erabiltzen hari gara, eta erabiltzen hasi aurretik baimentzea bermatu behar dugu. Horretarako kontsumitzaile gakoa eta kontsumitzaile sekretua idatzi behar ditugu kodean. Baimentzeak aukera asko ematen ditu Twitter API-arekin, beraien artean txioak argitaratzeko aukera, baina gure aplikazioan txio bilatzaile bat bakarrik behar dugunez ez dugu baimentzearekin ezer gehiago ukituko.

Gure lan instrumentua Eclipse dela gogoratuz eta Javaz hari garela idazten jakinik, API-a dakarren .jar fitxategia Eclipseko gure proiektuan kargatu behar dugu. Javako liburutegi hau lortzeko Twitter4j-ra [23] joko dugu, non liburutegi ez ofiziala dela gogoratzen bai digute beraien web orrian baina Twitter API berak gomendatzen du bere dokumentazioan Javarekin lan egin behar badugu.

Twitter4j ongi integratu ondoren gure programan, bere aukerak erabiltzen hasi gaitezke.

4.2.2 Txio bilatzailea sortu

Txio bilatzaile bat ez zegoen bezeroaren eskaeretan baina aurreneko demoan bere erabilgarritasuna erakutsi ondoren proiektuaren helburuetan sartzea erabaki genuen. Demoan sortutako bilatzaileak hitz gakoekin bakarrik funtzionatzen zuen baina bilaketa modu gehiago gehitzeko aukerak ematen zituela eta ikusirik Twitter API-ak, beste hiru gehitu genizkion.

[23] Twitter4j Liburutegia [Link]

Bilatzailearen erabilera oso sinplea da.

```
//Bilak_testu= bilatu nahi dugun testua izango den.

Query query = new Query(Bilak_testua);
QueryResult result = twitter.search(query);
```

Eskaera bat egingo diogu Twitterri bilatu nahi dugun testuarekin. Lortutako emaitzak ondoren erabiltzaileari erakutsiko dizkiogu interfaze grafikoan.

4.2.3 Bilatzailearen aukera ezberdinak aztertu

Gure aplikazioak ordea, aukera ezberdinak eskaintzen dituenez, bilaketa testua aldatu egin beharra dugu eskari edo bilaketa aukera bakoitzeko:

- ✓ Hitz gakoa: Eskera arrunta da, non txio baten barruan, traola moduan, URL baten barnean ala erabiltzaile baten izenean agertu daitekeen hitz edo esaldi bat izango da.
- ✓ Traola: Txio bat zertaz doan jakiteko aukeretako bat dira traolak, baina beste mila modutara erabiltzen dituzte Twitterreko erabiltzaileek. Hori dela eta bilaketa testuari Traola bilaketa bat bihurtzeko "#" ikurra gehitu behar diogu hasieran :

```
Bilak_testua= "#"+Bilak_testua;
```

- ✓ Aipamenak: Txio batzuk orokorrean idatzi beharrean norbaiti doaz idatziak, mezu bat bihurtuz, baina kasu honetako gehienak aipamenak dira, non pertsona bat izendatu ahal dezakegun. Beraien artean mota arraroenak berTxio-ak dira non nik idatzitako txio bat, norbaitek berTxio bezala zabaltzen badu, nire izena aipamen bezala bihurtzen baita. Twitter API-ari egindako eskaeran bilaketa testua aldatu behar dugu hasieran "@" ikurra idatziz:

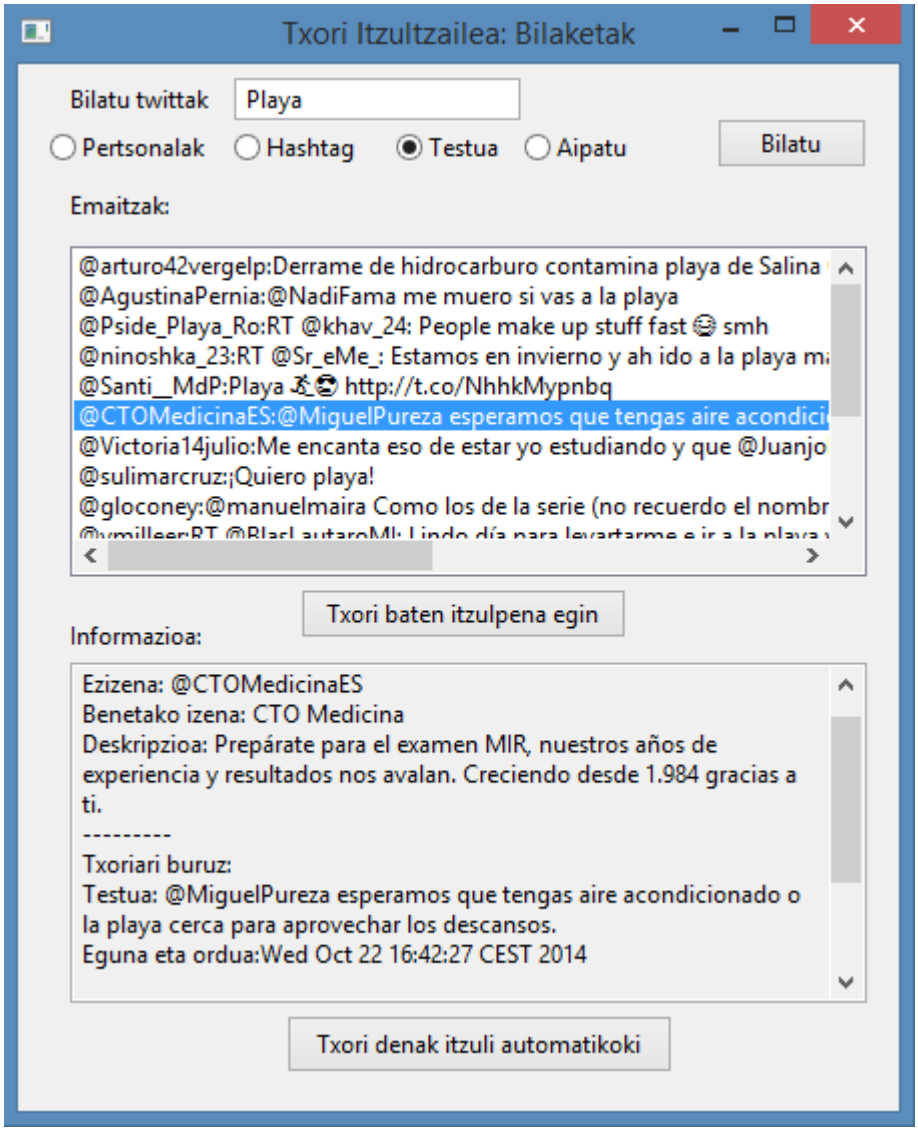
```
Bilak_testua= "@"+Bilak_testua;
```

- ✓ Erabiltzaile jakin baten txioak: Kasu berezienak dira hauek kodean inplementatzerako garaian. Ulertzeko aldiz errazenak, non bilaketa hauen funtsa, guk nahi dugun erabiltzailearen txioak lortzea den. Eskari hau egiterako orduan ez da aski bilaketa testua aldatzea, baizik eta eskaera mota aldatu behar da "from:" aukera gehituz, honela geratuz:

```
//Bilak_testu= bilatu nahi dugun testua izango den.

Query query2 = new Query("from:"+Bilak_testua);
QueryResult result = twitter.search(query2);
```

Emaitza hauek, lehen esan bezala, erabiltzaileari erakutsiko zaizkio interfaze grafikoan lista batean, non txioak hautatzeko aukera emango dien eta aplikazioaren beheko aldean informazioa agertuko den.



Irudia 16 - Bilaketaren emaitzak interfazean

4.3 Fitxategiak irakurri

Sarrerako datuak lortzeko bigarren aukera, fitxategi bidez irakurtzea da. Fitxategi hauek bezeroak hornitu dizkigu, non batean txio batzuen identifikadoreak agertzen diren txioaren testuaz jarraiturik eta bigarrengoan berriz, bestelako txio batzuk daude, goian azaldu bezala lehenik idatzita dauden hizkuntza, ondoren txioaren testua eta azkenik erabiltzaile izena, ordena horretan agertzen direnak.

Fitxategien formatua ezagututa beraz, orain gure lana bertako testu arrunta jaso eta txio bat bihurtzea izango da, ondoren berarekin jolastu eta azkenik itzultzeko. Hau egiteko fitxategietako elementuak bereizi behar ditugu.

4.3.1 Txioak dituen fitxategiak irakurri

Fitxategi hauek ".csv" formatuan etorriko dira. "Comma-Separated Values" izeneko fitxategi hauen ezaugarri nagusienetako bat, izenak dioen bezala, barneko taula itxurako datuak koma ikurraz bereizirik agertuko direla izango da. Datu hau erabiliko dugu fitxategiko elementuak bereizi eta txio bihurtzeko.

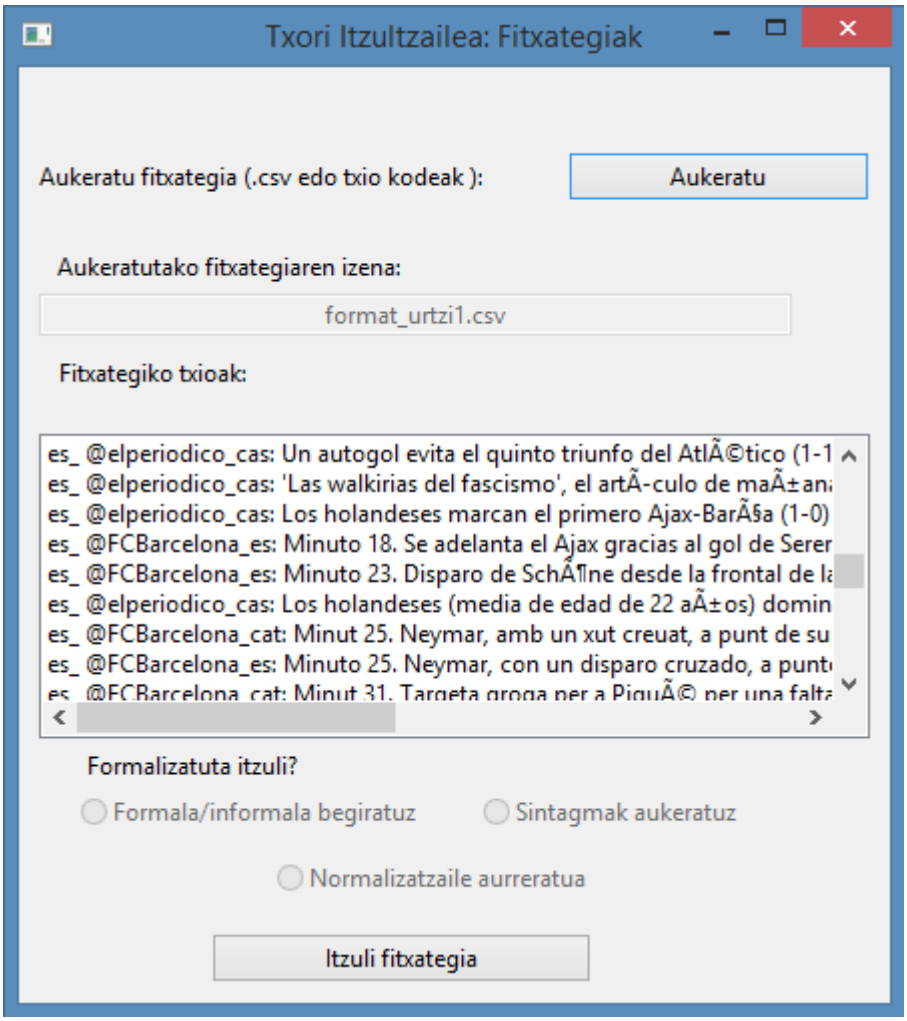
Izandako lehen arazoa hemen agertu zen, non fitxategian elementuak ez baizeuden komaz bereizirik. Baina komarena izena bakarrik da, ez da arraroa honelako fitxategiak beste bereizgailu batekin etortzea. Gure kasuan tabulazio bidez daude bereiziak. Hortaz konturatzean erraza da fitxategiko datuak guretzat lortzea:

```
// Non gure kasuan cvsSplitBy tabulazioa izango den, hau da: \t
                    line.split(cvsSplitBy);
```

Hiru elementuz osatutako fitxategia denez, txio bat sortuko dugu datu horiekin gure aplikazioaren barnean; hizkuntza, erabiltzailea eta testua gordeez. Datu hauek testu arruntak izango dira, hau da, String-ak. Itzultzen hasi aurretik erabiltzaileari pantailaratuko dizkiogu datu hauek, berak hartu dezan erabakia benetan itzuli nahi dituen txioak hauek direla, eta lehen begirada bat eman dezan bertan agertzen diren datuei. Erakutsiko zaizkion txioak ordea ez dira fitxategiak dituen txio denak, hizkuntzaren iragazki bat pasatzen bait diogu lehenik, aplikazioak gordea duen hizkuntzarekin alderatuz, honela itzuli nahi dituen txioak hasiera batean aukeratutako hizkuntzan idatziak egon behar dute itzultzaileak aukera izateko.

Irakurketa azkar bat egingo du, lortutako datuak irakurri bakarrik egin behar baititu.

Fitxategi honetako txioak, 2.kapituluan ikusi bezala kodetuak daude UTF-8-an.



Irudia 17 – Txioen fitxategiaren irakurketa

4.3.2 Identifikatzaileak dituen fitxategiak irakurri

Identifikatzaileak dituen fitxategia ".txt" arrunt bat da, hau da bertan dagoena testu hutsa izango da. Beste fitxategiarekin alderatuz honek bi elementu bakarrik ditu baina informazio gehiago. Identifikatzailea Twitter API-ari bidaltzen badiogu txioaren informazio oso lortu genezake, eta hori da egiten duguna.

CSV formatuaren antzera, honako fitxategi hau ere tabulazio bidez dago bereizia, beraz kodean, prozesu berdina erabiliko dugu identifikadorea eta testua jaso eta gure aplikazioan txio bezala gordetzeko. Prozesu hau berehalakoa da.

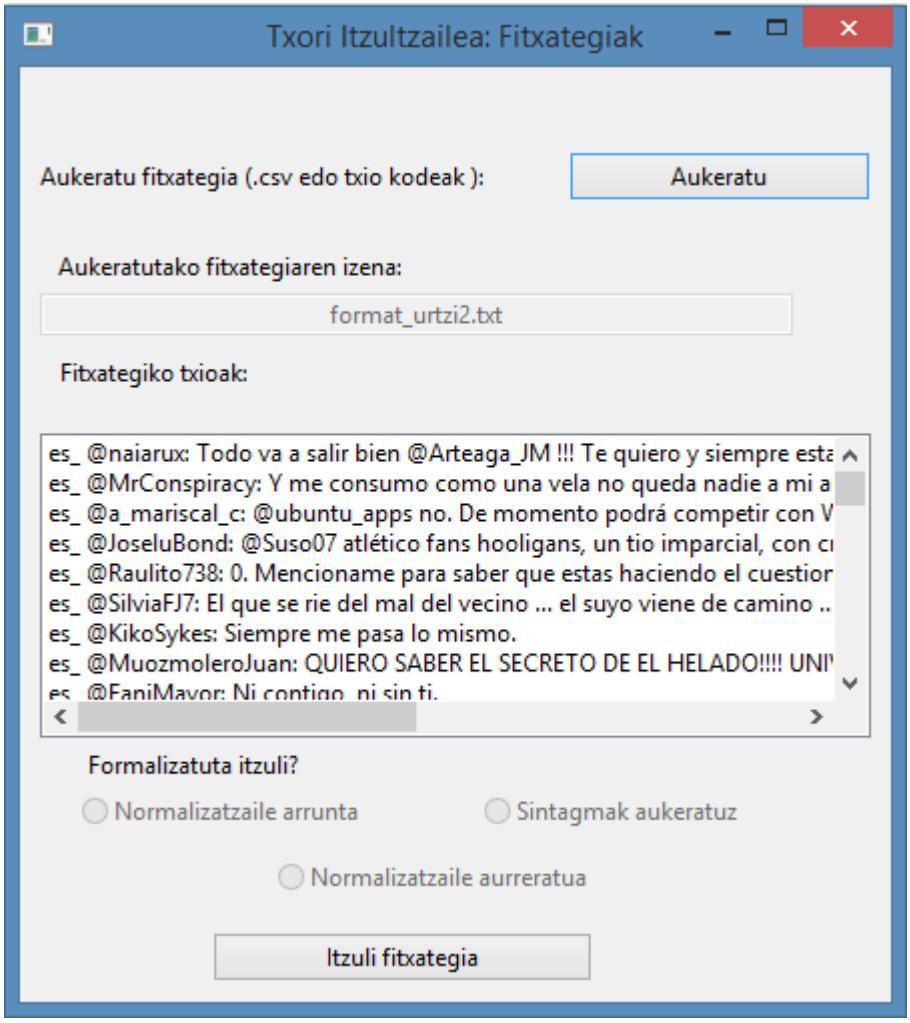
Erabiltzaileari txioak erakusterako garaian berriz, ezin dugu lehen bezala egin, ez baitigu eskura ez erabiltzaile izena ez txioak datozen hizkuntza. Hau dela eta identifikatzailearekin jokatu behar dugu. Aplikazioa baimendua dagoenez, txio bilatzailearekin egin dugun bezala, ez dugu berriz ere baimendu beharrik, baina Twitter API-arekin dugun konexioaz baliatu behar gara identifikadore horiek bilatzeko eta honela beraien datuak lortzeko. Prozesu honek, aurrekoarekin alderatuz, denbora gehiago emango du,

Twitter-API-arekin alderatu behar baitu identifikadore bakoitza. Baina hau lortu ondoren, txioaren beharrezko informazioa izango dugu eskura.

```
//Non Ident, behar dugun txioaren identifikatzailea izango den.
twitter.showStatus(Long.parseLong(Ident));
```

Txioaren datuak lortu ondoren, interfaze grafikoan erakutsiko ditugu, beti ere hizkuntzarekin alderatu ondoren, hau da, gure kasuan gaztelaniazko txioak bakarrik erakutsiko dizkiogu.

Identifikatzaileaz baliatuz lortu ditugun txioen testuak ez daude kodetuta UTF-8-an beraz itzultzailera bidaltzerakoan arazoak aurreztuko dizkigu.



Irudia 18 – Identifikatzaileen irakurketa

4.4 Txioak maneiatu

Sarrera datuak jaso ondoren gure txioak itzultze prozesura sartzen ditugu. Leheneko urratsa txioaren testua lortzea izango da, hau da, testu hutsa, txio bat osatzen duten elementuak identifikatu eta alde batera utzi ondoren geratzen zaiguna. Txioaren elementuak identifikatzerako orduan kontuan hartu behar ditugu Twitterren aurki ditzakegun aukera anitz guztiak.

Elementu nagusiak honakoak dira: testu arrunta, traolak, aipamenak, berTxioak, URL-ak eta karaktere bereziak. Elementu hauek eta beraien arteko bat-egiteak zailtzen dute txioaren benetako testua lortzea, eta itzultzailera bidalitakoa errealitatean txioak esan nahi duena izatea. Elementuak banaka identifikatzen dira bai Twitter API-ak ematen dituen laguntzekin bai, guk sortutako metodoekin. Identifikatu ondoren gorde egin behar ditugu, testu arruntaren itzulpena ez baita gure helburu nagusia, baizik eta itzulitako itzulpena txioarena izatea. Honekin esan nahi dena da irteerako datuak txioak direla eta hasierako txioan elementu hauek itzulitako txioan agertu behar direla emaitza ontzat hartzeko.

Arazo nagusi batzuk agertu dira ataza hau aurrera eramatean non traola batzuk testuaren parte izango diren eta beste batzuetan ez duten ezer axola txioan esan nahi denarekin. Bestalde, bigarren arazo bezala berTxioak egiteko era ezberdinak daudela konturatu ginen non testuaren hasieran, berTxio bihurtzen duen "RT" idatzi aurretik norberak bere testuan bestelako hitzen bat gehitzen badu (kasu askotan "." Bat gehitzen dute) Twitterrek ez du berTxio bezala ezagutuko.

Elementuak identifikatu eta arazo hauek bereizteko moduan txioak tratatzera joko dugu.

4.4.1 Txioak tratatu

4.4.1.1– Txioaren hizkuntza

Txio baten elementuak identifikatu eta bereizten hasi aurretik txioaren hizkuntza zein den jakitea komeni da, guk itzuli nahi dugun hizkuntza ez bada lana aurrezten baitio programari. Fitxategietatik irakurritako txioak jadanik ze hizkuntzetan idatzia dagoen testua badakarte, baina identifikadoreetatik baliatutako eta txio bilatzailetik lortutako txioak ordea ez.

Txio hauen hizkuntza zein den jakiteko Twitter API-ak eskaintzen digun baliabide bat erabiliko dugu. API-ak aukera ematen digu txioa idatzi den hizkuntza zein den jakiteko eta horretarako berak egileak nondik idatzi duen begiratuko du eta txioa idatzia dagoen hizkuntzaren susmoarekin alderatuko du. Egindako probetan emaitza onak izan ditu eta gaztelaniazko txioak identifikatzeko gai da. Alde txarra berriz hizkuntza nagusi asko bereizten baldin baditu ere, euskararekin arazo bat du. Twitter API-ak ez du euskara hizkuntza bezala identifikatzea lortzen. Konturatzen gara proiektuaren etorkizunerako bide hau ez dela egokiena eta bigarren gomendio bezala erabiltzera pasa beharko litzatekela, baina gure proiektuaren helburuetarako aski izango da, esan bezala gaztelerako kasuetan emaitza onak ematen baititu.

Txio bat gazteleraz idatzia egon daiteke, baina testu erdian esaldi bat izan dezake ingeleraz idatzia komatxo artean. Twitter API-ak testu hau hizkuntza bat ala beste den bereizteko probabilitate bat erabiltzen du: %60. Testuaren ehuneko 60-a hizkuntza batean baldin badago bera dela esango digu, eta gainera bere gomendioa berrizteko txioa idatzi duen erabiltzailearen kontuaren informazioaz baliatuko da bere kokapena jakiteko eta leku horretan hitz egiten den hizkuntza erabiliko du laguntza bezala.

Beraz, txioaren hizkuntza asmatu ondoren testuaren tratamendua hasiko dugu.

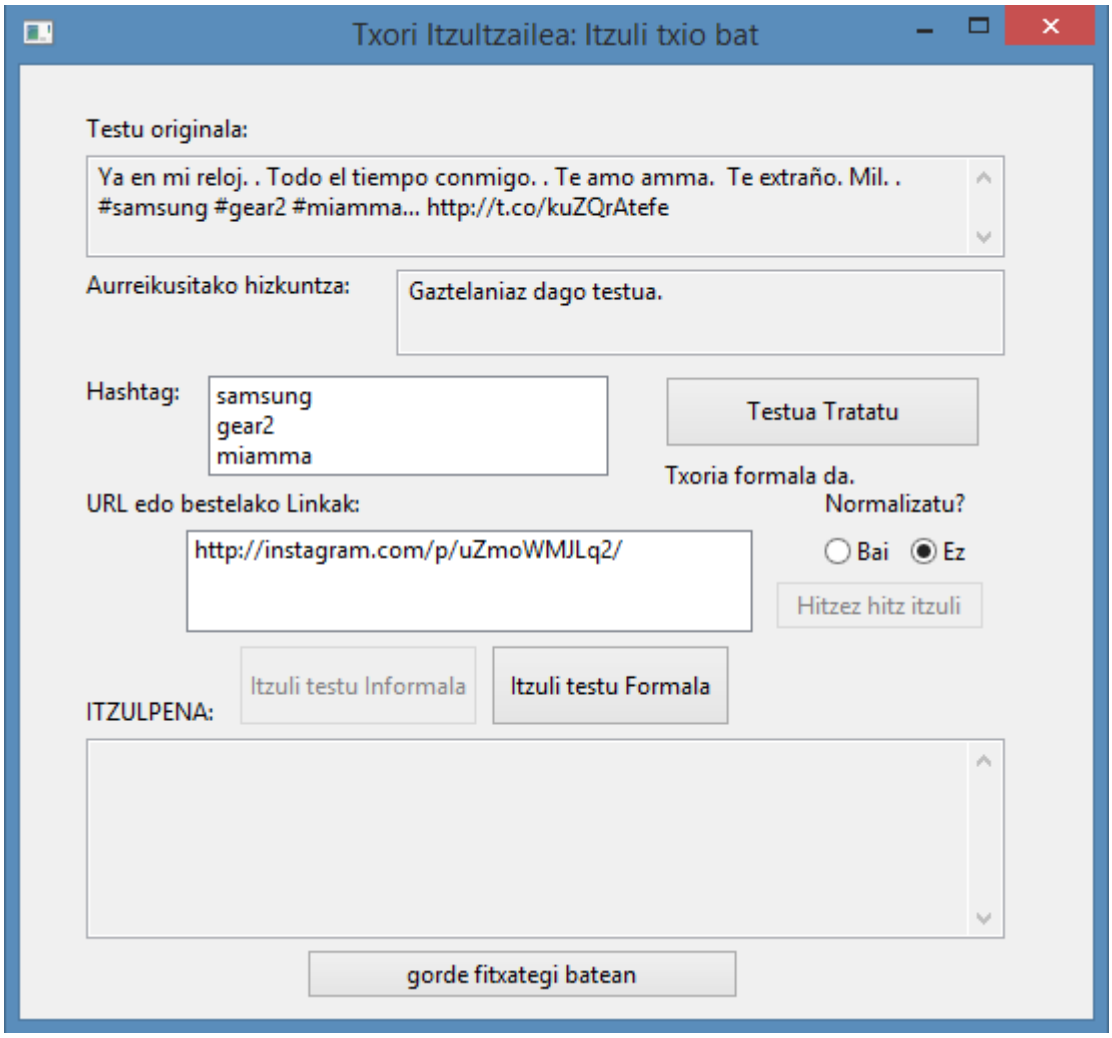
4.4.1.2– Txioen elementuak

Twitter API-az baliatuz hainbat elementu bereiztuko ditugu: URL-ak, berTxioak, karaktere bereziak eta traolak.

- ✓ **URL-ak:** Txioetan URL-ak oso normalak dira eta ez dute itzulpenean eragiten, ez baitira itzuli behar, eta irteerako formatuan azaldu den bezala itzulpen testuaren amaieran agertzearekin aski da. URL gehienak argazkiak edo web-orri baterako estekak dira.
- ✓ **Karaktere bereziak**: Emotikonoak bezala ere ezagutzen dira. ASCII barnean aurki ditzakegun karaktere berezi hauek itzulpena kaltetu dezakete, Matxin itzultzaileak ez baititu ulertzen eta "?" karaktereaz itzultzen baitigu. Karaktere berezi gehienak Twitter API-ak baliatzen digun metodoaz bereiz ditzakegu, baina gaur egun txioak idazteko dauden Twitter kliente mordoak lan hau zailtzen du eta ez du beti asmatzen.
- ✓ **berTxioak**: API-az baliatuz berTxio arruntak bereiz ditzakegu baina lehen azaldu bezala arazo bat dugu eskuz editatzen diren "RT"-ekin. Hau ekiditeko beste elementuak testutik bereizi ondoren bilaketa bat egingo dugu "RT" karaktereak bilatzeko. Ala bada itzulpenaren hasieran RT bat gehituko diogu emaitzari berTxio bat dela jakinarazteko.
- ✓ **Traolak**: "#" ikurraz bereizten ditugun hitz solteak traolak dira, eta orokorrean txioa zertaz doan esaten dute. Traola hauek batzuetan testuaren parte dira eta itzuli egin behar dugu, besteetan esaldi amaieran jartzen den hitz-gako bat da. Hau zuzentzeko, Jazzy API-a erabiliko dugu, non gaztelaniazko hiztegi batekin alderatuko dugun traola daraman hitza eta hiztegian baldin badago itzuli egingo dugu, ala ez bada, itzulpenetik kanpo geratuko da. Bi kasuetan traola originala itzulpenaren amaieran agertuko da gainontzeko elementuekin batera.

Bereizteko geratzen den elementua aipamena da eta hau Twitter API-aren laguntza gabe tratatuko dugu. Aurreko elementuak identifikatu ondoren eta gerorako gorde ondoren, testutik kenduko ditugu, honek testu hutsa bakarrik utziko, aipamenak izan ezik. Aipamenak "@" ikurraz identifikatu ahal ditzakegu eta hori jakinda bi leku ezberdinetan agertuko zaizkigu. Esaldi hasieran agertzen denean a bildu bat, zuzeneko mezu bat bihurtzen da txioa, eta testu erdian dagoenean berriz aipamenak, baina gure kasuan biak berdin tratatuko ditugu, itzuli ondoren txioan agertzen diren izen edo aipamen denak amaieran idatzita itzuliko baititugu.

Hau amaitzean testu hutsa bakarrik geratuko zaigu eta hau itzultzera joko dugu.



Irudia 19 – Txio baten hizkuntza eta elementu ezberdinak

4.4.2 Txio formalak eta informalak bereiztea

Itzultzerako garaian bi testu mota aurkituko ditugu, non modu ezberdinetan itzuli nahi izango ditugun. Mota hauek testu formalak eta informalak dira eta itzulpenaren kalitate mailan esateko asko dute. Baina itzulpena nola egiten den ikusi aurretik nola ezberdindu ikusiko dugu Jazzy API-a erabiltzen.

Jazzy API-a 2. Kapituluan azaldu bezala "Spell Cheker" bat da, hau da zuzentzaile ortografikoa. Bere integrazioa proiektura Twitter4j –en berdina da, non beharrezkoa den .jar liburutegiak kargatuko ditugun Eclipsen. Liburutegi hauek gure aplikazioarentzako zuzentzaile ortografiko bat gehitzeko baliabideak eskainiko dizkigu. Jazzy API-a guk emandako hiztegi batez baliatzen da lan hau egiteko, beraz itzuli nahi dugun hizkuntzaren hiztegi bat erantsi beharko diogu aplikazioari "Spell Checkerrak" eragina izan dezan.

Bezeroarekin izandako bileretako batean erabakitzat hartu genuen txio bateko testua formala izateko hitzen %60-a hiztegian agertu behar du, aplikazioak honela ontzat emanda. Informala izateko berriz ehuneko horren betik izan behar du.

Metodo hau ez da perfektua, baina gure helburuetarako egokiena da. Perfektua ez dela esaten dugu zeren testu laburretan, adibidez, bi hitzeko esaldi batean, hitzetako bat gaizki egoteak informala bihurtuko du, baina itzultzerakoan formalki itzultzeak onura gehiago izango zuen. Hau dela eta beste bide batetik joan beharko ginen, baina bezeroaren beharretarako metodo hau oso egokia da, eta proiektuan zehar ikusi dugunez emaitza egokiak eman dizkigu.

Erabiltzailearen aplikazio grafikoan atal honek informala ala formala den bakarrik adieraziko dio erabiltzaileari, eta horren arabera itzultzeko bi aukera emango dizkio:

2. **Testu formalean**: Normalizatzailea erabiltzeko aukera emango diogu, nahiz eta lehenengo aukera ezezkoa izango den. Aukera hau emaitzak ikusirik erabaki genuen. Emaitza hoberenak testu formaletan normalizatzailerik erabili gabe lortzen baitira.
3. **Testu informalak**: Testu informaletan derrigorrezkoa da normalizatzailea erabiltzea itzulpen zuzen bat lortzeko. Normalizatzaileak ondoren ikusiko ditugu nola aldatzen duten testua itzulpen hobeak lortzeko, baina bigarren aukera bat ere emango zaie, non hitzez hitz itzultzea izango den.

Testu informalak itzultzean aurkitzen dugun arazo handiena esaldiaren egiturarekin topatuko dugu non azterketa lexiko edo semantiko bat egiten hasten bada Matxin erotu eta emaitza kaxkarrak emango dizkigun. Hori dela eta, esan dugun moduan, hitzez hitzeko aukera emango diogu erabiltzaileari, nahiz eta itzulitako emaitzak zentzu asko ez izan guretzako, hitzen itzulpen arruntak lagundu egiten baitu jatorrizko txioaren esanahia ulertzen.

4.5 Txioen normalizazioa

Normalizatzaileak txioko testua aldatuko dute itzulpen hobeago bat lortzeko. Horretarako normalizatzaile ezberdinak inplementatu daitezke, bakoitzaren beharretarako. Gure kasuan normalizatzaile arrunt bat sortu genuen banakako itzulpenak orokorrean eta testu informalekin laguntzeko. Aplikazioak aukera eskaintzen du normalizatzaile gehiago eta hobeak batzeko, bai itzulpena bai aplikazioaren funtzionalitateak areagotzeko.

Normalizatuko dugun testua jadanik ez da txio bat, bertako elementuak kendu baititugu, baina oraindik arazo bat ager daiteke. Gorago azaldu dugun moduan, karaktere bereziak bilatzean txio batean, Twitterrek ez ditu denak identifikatzen, eta baten bat normalizatze testura iristen bada emaitza ezezagunak ekarriko dizkigu. Hau da, normalizatzaileari pasatako testua okerrago utziko du.

4.5.1 Normalizatzaile arrunta

Guk sortutako normalizatzea da honako hau. Bere helburua testu informalak ahalik eta hobekien txukuntzea da eta testu formalek izan ditzaketen akats gutxiak zuzentzea. Horretarako Jazzy API-a erabiliko dugu besteak beste.

Normalizatzaile bat sortzerako garaian lehenik gure testuak izan ditzakeen akatsetan zentratu behar gara, eta hortik abiatuta konponbide bat bilatuko duten baliabideak sortu. Gure kasuan Twitter esaldi motzezko testuetan oinarritzen denez, sintaxi sinplea edo sinplifikatua izango du. Informaletan sinplifikatua baina ez estandarra. Hori dela-eta lexikoa izango da gure helburua, eta hitz ezezagunak tratatuko direnak. Hori esanda, akats ortografikoetan oinarrituko gara. Jadanik ikusi dugun Jazzy API-a erabiliko dugu hiztegiarekin parekatzeko gure testuko hitzak banan bana eta horietako bat gaizki badago bera zuzentzen saiatuko gara hiztegian antzeko hitz bat bilatuz. Hortaz API-a bera arduratuko da, gomendioak emateaz.

```
        //Non "word" gaizki dagoen hitza izango den eta "threshold" itzultzea
nahi ditugun gomendio kopurua.

        spellChecker.getSuggestions(word, threshold);
```

Fitxategietatik jasotako txioak itzultzerako orduan gomendio kopurua bakarrera jaitsi dugu, ezin baitizkiogu erabiltzaileari erakutsi aukera denak eta bat aukeratzeko eman. Aukera hau interesgarria izango litzateke banan banako txioekin, baina proiektuaren helburuetan adostu zen, normalizatzaile arrunt honek lehenengo gomendioa erabiliko zuela.

Gomendatutako hitza, akats ortografikoa zuen hitzarekin ordezkatuko dugu, eta honek itzultzailera, berak ulertuko duen hitz bat bidaliko du.

- ✓ Alde onak: Akats ortografiko arruntak direnean lehenengo gomendioarekin primeran geratzen da %80 batean eman diren probekin. Beste kasu nabari bat hitzetan letrak errepikatzen direnean ematen da. Adibidez "locoo" hitza edo "Aaamigo" hitzak % 90 batean asmatzen du gomendioarekin. Honek itzulpena asko hobetzen du.

- ✓ Alde txarrak: Asmatzen ez duenean, itzulpena oso kaotikoa bihurtzen da ez baitu esaldiak zentzurik eta beraz ez da ulertzen jatorrizko txioaren esanahia.

Hau jakinik, aurre ikusten hasi ginen gure testu formal eta ez formaletan izango zuen eragina. Testu formaletan akats ortografikoak urriak dira baina aurkitu ditzakegun hitz arraroak beste hizkuntza batekoak dira, batez ere ingeleraz daudenak. Hitz ingeles hauek normalizatzailetik ez baditugu pasatzen Matxinek izen berezi bezala hartzen ditu eta itzulpenak oso ondo geratzen dira. Bestalde normalizatzailetik pasa ondoren eta hiztegian antzeko hitz bat bilatzen badu Matxini iritsitako esaldia ez da erabat ona izango eta beraz itzulpena ez da ulertuko.

Testu informaletan berriz, normalizatzailea beharrezkoa ikusten dugu, akats ortografikoak baitira gehienbat ematen diren akatsak eta hauek zuzentzeko oso ondo dator normalizatzailea. Baina honek ez du beti asmatzen. Teorian emaitza oso onak eman beharko lituzke baina hurrengo kapituluan ikusiko dugun bezala, benetan lortutako emaitzak ez dira batere onak. Hau dela eta, ez normalizatzeko aukera bat ere emango dugu txio informalekin, beti ere esaldi osoaren itzulpena egin beharrean, hitzez hitz egiten bada.

4.6 Itzulpena egin Matxin bidez

Txioa testu bihurtu dugu eta testua eraldatu egin dugu akatsak zuzentzeko itzultzaileari dena errazteko. Gure proiektua gaztelaniatik euskarara itzultzea denez, erabiliko dugun itzultzailea Matxin API-a izango da. Matxin gure proiektuan integratzeko aukera badago, baina honek bizkortasuna irabaztearekin, pisuan ere irabaziko zuen eta hori ez da erabiltzaileak bilatzen duena. Hau dela eta Matxinekin izango dugun iterazioa bere web-zerbitzutik izango da, non Perl bidez egingo dugun gure eskaera eta itzulpena itzuliko digun.

4.6.1 Kodetu, deskodetu testua

Garapenean zehar arazoak izan ditugu, baina testuaren kodeketa izan da lan gehiena eman digun konpontzen. Kontuan izan behar dugu Mashup bat egitean batez ere, erabiltzen ditugun elementu nagusiak elkarrekin erlazionatzean, beraien arteko deiak berdin kodetuak daudela, gure kasuan Gaztelaniako txioekin lan egiten baitugu. Gaztelaniazko testuetan azentu-marka asko daude eta hauek ez dute onartzen edozein kodeketa.

Arazoaz konturatu ginen irakurtzen ditugun bi fitxategietako baten testua UTF-8 bidez kodetua bait zegoen eta bestea ez, hau dela eta Matxinek itzuli ondoren emandako emaitzak okerreko kodeketaz itzultzen baitzituen.

Beraz hau konpontzeko, aplikazioan emandako API eta datuen arteko erlazioak kodeketa berdinera erabiltzera aldatu behar izan ditugu. Aplikazioaren barnean bi kodeketa nagusi aurkitu ditzakegu:

- ✓ **UTF-8 kodeketa** [24] : Edozein karaktere Unicode-ra kodetzeko gai da. Erraza da identifikatzen. Aplikazioan Jazzy API-ak erabiltzen duen hiztegia kodeketa honen bitartez dago gordea, hau dela eta, tratatutako testua eta normalizatutako testua UTF-8 kodeketara bihurtu behar dugu. Twitter API-ak ere kodeketa honetaz baliatzen da, beraz identifikadoreak dituen fitxategia eta txio bilatzaileko txioak itzultzailera bidali aurretik deskodetu egin beharko ditugu.
- ✓ **ISO-8859-1 kodeketa**[25]: ISO arauaren barnean, alfabeto latinoa kodifikatzeko erabiltzen diren arauak dira. Matxin itzultzailearen web-zerbitzuari bidaltzen zaion testua kodeketa honen bidez egin behar da azentu-markak eragina izateko eta itzulpen egokiak izan ditzan.

Arazoak Matxinekin zetozen, bere API-ak UTF-8 kodeketako testuak erabiltzen baititu baina guk erabiltzen dugun web-zerbitzuak ez ditu onartzen azentu-markak eta “ñ” –ak. Hori dela eta hau ulertzeko Gorka Labakarik jarri ginen harremanetan, non Matxin itzultzaileari aldaketa batzuk egin zizkion ikusteko gure arazoa zein zen. Konturatzean bidalitako testuak ISO-8859-1 kodeketa izan behar zuela, arazoa desagertu zen.

[24] Wikipedia: UTF-8 kodeketa [Link]
[25] Wikipedia: ISO-8859-1 kodeketa [Link]

Bi kodeketa[26] hauek teorian bai azentu-marka eta bai alfabeto latinoko karaktere bereziak kodetzeko gai dira, baina praktikan ISO-8859-1 kodeketa izan da bide bakarra.

Aplikazioaren barnean, kode zati honekin lortu dugu:

```
byte[] utf8BytesStream;

//Non "ItzuliTest", guk itzuli nahi dugun testu normalizatua izango den.

utf8BytesStream = ItzuliTest.getBytes("UTF8");
String KodetuTest = new String(utf8BytesStream, "ISO-8859-1");

//Eta "KodetuTest" Matxin itzultzaileari bidaliko diogun deira prestatutako
testua izango den, ISO-8859-1 kodeketa eragin ondoren.
```

4.6.2 Matxin itzultzailea atzitu

Algoritmoan zehar aurrera emanda, jadanik txioa testu huts bihurtu dugu, eta testua normalizatu egin dugu itzulpena hobetzeko. Testu hau itzultzailera bidali aurretik beharrezko kodeketa eta deskodeketak erabili ditugu, beraz dena prest dugu itzulpen bat lortzeko.

Matxin itzultzaile automatikoak gaztelaniatik euskarara itzultzen ditu testuak, eta gure helburuekin bat datorren API bat du. Horrez gain, mashup aplikazioak pisu handiagoa ez izate arren, Internetez atzi daitekeen web-zerbitzu bat ere eskaintzen du, eta gu hontaz baliatuko gara. Web-zerbitzua atzitzeko Perl hizkuntzan idatzitako kode zati bat erabiliko dugu. Honek SOAP protokoloa erabiliko du eta barne kodetze bat, Base64 kodeketaz, itzulpena testu bihurtzeko.

Perl fitxategia honela geratuko da:

```
use SOAP::Lite;
use MIME::Base64;

#Irakurri parametroak
my $testua = $ARGV[0];
#Kodetu
$encoded = MIME::Base64::encode_base64($testua);
$response =  SOAP::Lite
    ->
proxy('http://ixa2.si.ehu.es/matxin_zerb/translate_named.cgi')
    -> uri('MATXIN')
    -> EsEu($encoded)
    -> result;
#Deskodetu
MIME::Base64::decode_base64($response);
```

Baina Perl fitxategiari deia gure aplikazio beratik egin behar dugu, eta dei hau, zoritxarrez itzulpen bat nahi dugun aldiro egin behar dugu. Honek exekuzio denbora esan nahi du, aplikazioa

[26] Which is better UTF-8 or ISO-? [Link]

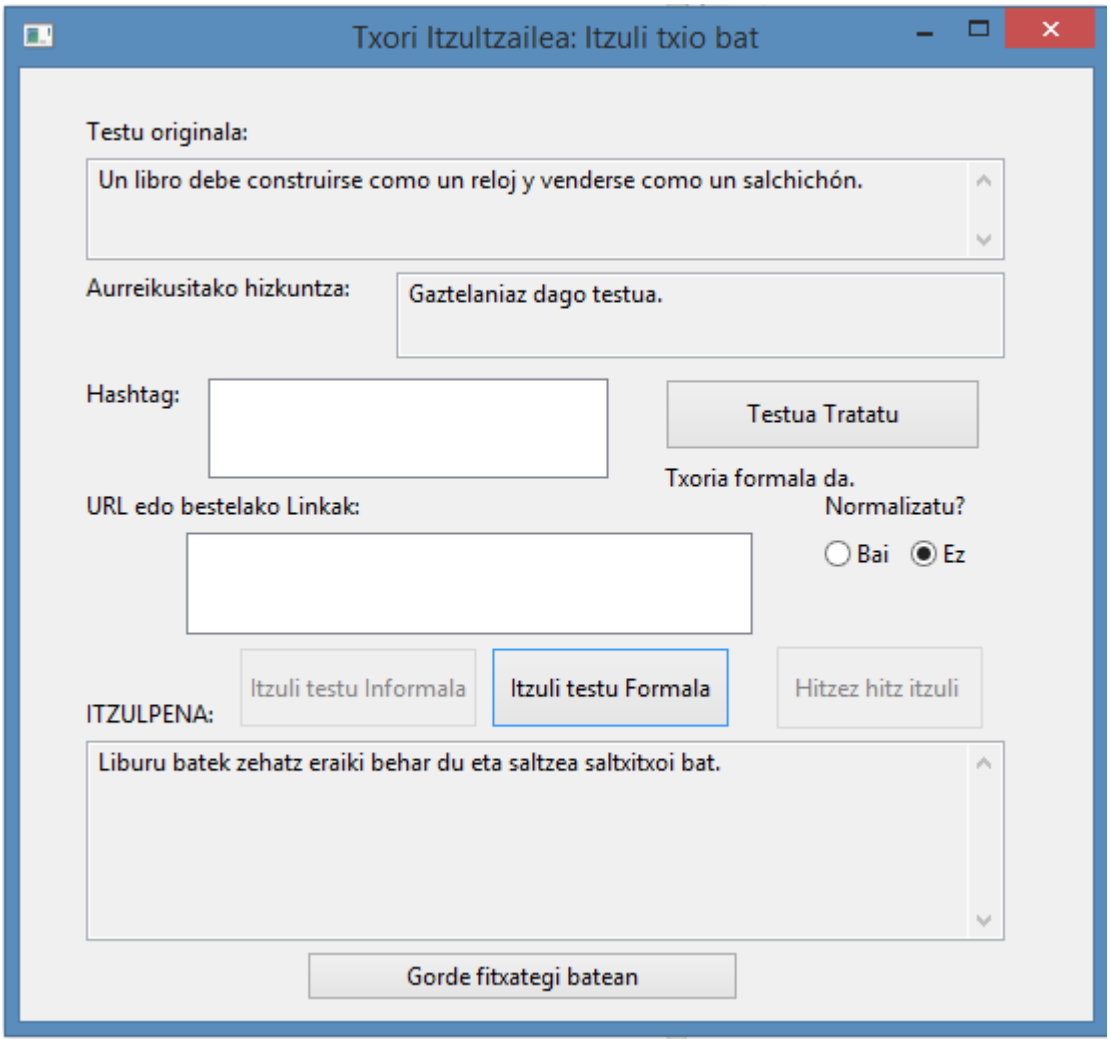
itzultzaileren zain geratzen baita eta txio askoko fitxategietan, edo hitzez hitzeko itzulpenetan, bere denbora hartzen du dei bakoitza egin eta Matxinen erantzuna jaso arte.

Kodean sartuta, Perl fitxategiari dei egiteko erabiliko dugun bidea honakoa da:

```
//Non "KodetuTest" ISO-8859-1 kodeketa eragin ondorengo testua izango den.
//Perl fitxategia proiektuaren karpeten barnean gordeko da.

process = Runtime.getRuntime().exec("perl .\\bin\\matxinEsEu.pl
\""+KodetuTest.toString() +"\"");
process.waitFor();
```

Emaitza jaso ondoren, erabiltzaileari erakutsiko zaio interfaze grafikoan.



Irudia 20 – Itzulpenaren emaitza

4.6.3 Hitzez hitz ala esaldi osoa itzuli

Dokumentu honetan zehar aipatu ditugu testu informalak itzultzeak dituen arazoak, eta nola horiek konpontzeko hitzez hitzeko itzulpenak lagundu gaitzaken. Itzultzaile automatikoek, beraien artean Matxin, itzultzeko bidaltzen diogun esaldiaren azterketa semantiko eta linguistiko bat egiten dute, eta honek sortzen duen zuhaitz formako egitura erabiltzen dute itzuli ondoren berriz ere esaldiari edo hitz solteei esaldi itxura emateko. Testu informaletan ordea esaldiaren %60 gaizki dagoela esan genezake, eta ez beraz azterketa egitean sortzen den zuhaitza suposizioz betea dago, non itzulpena egin ondoren euskarako esaldiari zentzua eman nahian sortzen duen emaitzak kaotikoa atera daiteke.

Honen konponbide bat, eta gogora dezagun testu informaletan helburua ez dela itzulpen egoki bat sortzea baizik eta itzulpen txar bat ez sortzea, hitzez hitzeko itzulpena egitea izango da. Honekin emaitza hobeak lortzen dira orokorrean, baina esan bezala itzulpena ez da kasu gehienetan ona izango.

Kodeari begira, hitzez hitz edo esaldi osoa bidaliz egiteak ez du aldaketa askorik ekartzen. Testu informala ala formaletan, aukera dugu esaldi osoa itzultzeko eta hori, Perl fitxategiari deia egitean bidalitako testua, gure txioaren testu oso eta kasu batzuetan normalizatua izango da.

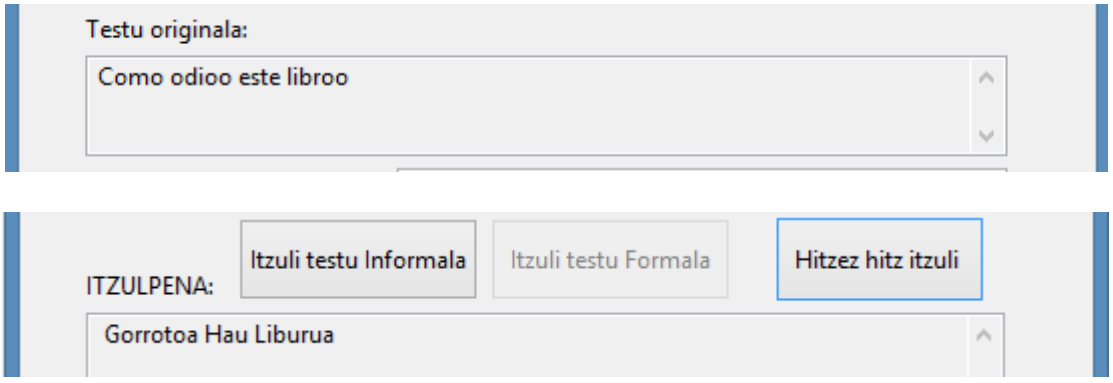
Bestalde, hitzez hitzekoan, testu informaletan bakarrik emango dugun aukera bat izango da, eta hau esaldia banandu ondoren, Matxini bidaliko diogu Perl bidez. Bide hau jotzean, emaitzak hobetuko ditugu, baina txioaren hitz bakoitzeko deia bat egin behar izango dugu Matxinen web-zerbitzura, bere erantzunaren zain denbora itxaronez.

Javan, esaldi bat hitzez hitz banatzeko honako kodea erabiliko dugu, adierazpen erregularren laguntzaz :

```
//Non Testua, itzuli nahi dugun esaldia izango den.

String[] hitzak = testua.split("\\s+|,\\s*|\\.\\s*");
```

Hitzez hitz, ala esaldi osoa itzultzean lortutako emaitzak hurrengo kapituluan aztertuko ditugu.



Irudia 21 – Hitzez hitzeko itzulpena

4.6.4 Itzulitako testua irteerako fitxategietan gorde

Itzulpena aurrean edukita banan banako itzulpenekin erabiltzaileari aukera emango diogu fitxategietan gordetzeko txio formatuan. Hau inplementatzeko garaian, gordeta genituen elementuak berreskuratu behar ditugu, itzulitako testua txio bihurtzeko berriz ere. Bigarren kapituluan ikusi dezakegun bezala, irteerako fitxategian elementu hauek agertzeko ordena bezeroarekin adostua dago.

Jatorrizko txioaren informazioa jakin behar dugu itzulpena berriz ere txio bihurtzerako garaien, ez baita berdina berTxio baten itzulpena edo argazki bat duen txioaren itzulpena. Bezeroarekin adostutako ordena honela gordeko dugu irteerako fitxategietan:

- ✓ berTxioa bada: Itzulpenaren hasieran, erabiltzailea idatzi ondoren "RT" bat idatziko dugu, itzulpena berTxio baten itzulpena dela adierazteko.
- ✓ Aipamenak baditu: Txioan zehar beste erabiltzaile baten aipamena baldin badu itzulpenaren amaieran idatziko ditugu. Baliteke jatorrizko txioan aipamen batek bere zentzua izatea testu erdian agertzerakoan, baina itzuli ondoren ezin dugunez jakin leku hori non den, amaieran jartzeko erabakia hartu zen.
- ✓ URL eta Karaktere bereziak: Aipamenekin bezala, honako elementu hauek itzulpena ez dute aldatzen eta nahiz eta kasu batzuetan zentzua izan testu baten barruen, guk amaieran erakutsiko ditugu.
- ✓ Traolak: Aurreko bi kasuen berdina da, ezberdintasun batekin. Traola batzuk itzultzailera bidali ditugu esaldiaren parte baitziren, beraz orain, jatorrizko traola gordetzeaz aparte itzulitako txioaren amaieran, itzulita agertuko da traola hori testuaren barnean. Ez dugu jarriko itzulitako traola, traola moduan, esanahi oso galtzen baitu.

Bezeroaren eskaera modura, jatorrizko txioarekin gordeko dugu itzulitako txioa. Jatorrizko txio hau UTF-8 ra kodetua egongo da, sarrera fitxategira gehitu nahi bada noizbait. Ondorengo itxura izango du itzulitako txio batek:

```
//Adibide1: berTxio formal bat, itzulpen onarekin.

Originala: @AgusMoraann: RT @JRodrguez7: Quiero playa
Itzulpena: @AgusMoraann:  RT @JRodrguez7:  Hondartza nahi dut
```

```
//Adibidea2: berTxio formal bat, URL eta 2 traolekin, non bat izen berezia den eta bestea hiztegian aurkitu daitekeen. Izen berezia ez du itzuliko eta amaieran agertuko da, baina "#playa" itzuli egingo du.

Originala: @radioeldia: RT @titsa: Recuerda que la parada de la #playa de las #Teresitas estÃ¡ anulada hasta que finalicen los trabajos de limpieza http://ow.ly/DgwN6

Itzulpena: @radioeldia:  RT @titsa:  Gogoratzen du gelditua hondartza baliogabetuta dago garbitasuneko lanak amai ezan arte http://ow.ly/DgwN6 #playa #Teresitas  http://ow.ly/DgwN6
```

5 .

Itzulpenen Emaitzak

"Kapitulu honetan aplikazioaren aukera ezberdinak erabiliz lortutako emaitzak balioetsiko ditugu non amaierako bertsioan nola eragin duten erabakitzeko eta emaitza partikular eta orokortuen arteko ezberdintasunak lortuz."

5.1 Banakako itzulpenak

Banakako itzulpenak kalitatezko itzulpen baten bermea behar dugunean erabiliko ditugu, edo txio arraroren bat baldin badugu. Hau jakinik, normalizatzeko aukera gehiago eskainiko ditu, eta hori dela eta emaitza oso ezberdinak lortu.

5.1.1 Testu formalak normalizatu gabe

Hauek dira Twitterren aurkitzen ditugun txio arruntenak. Gaztelerazko testu arruntak dira, beraz Matxin itzultzaileari bidalitako testuak ez dute aldaketarik jasango. Lortutako emaitzak Matxinen eta txio originaleko testua modu argian idatzita dagoen ala ez, beraien menpe geratuko da.

- **Adibidea 1:**

Txio originala: ¿Tienes un reloj inteligente? Dar la hora es lo de menos @20m http://t.co/utD9I3MXoJ

Itzulpena: Duzu erloju adimentsu bat? Ordua jotzea gutxieneko da @20m http://t.co/utD9I3MXoJ

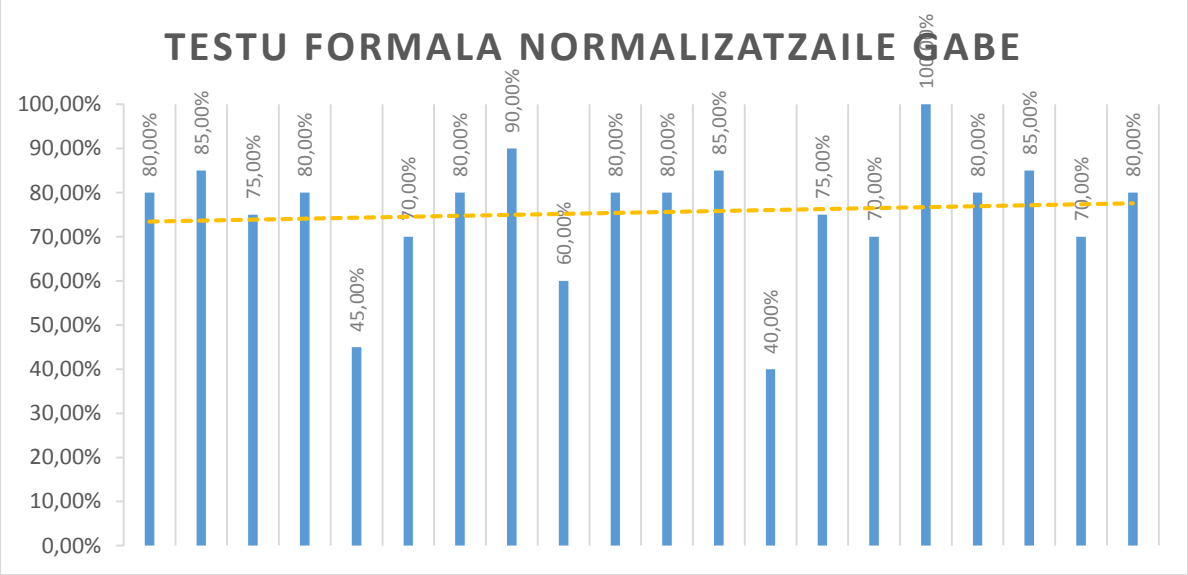
Balorazioa: Itzulpen ona da, non txio originalean esan nahi dena ulertzen den itzulpenean. Lortutako emaitzan gure aplikazioak ez du eraginik, ez baitugu txioko testua ukitu.

- **Adibidea 2:**

Txio originala: Y quien iba a pensar que me iba a terminar gustando tanto este libroo

Itzulpena: Eta pentsatzera zihoan me honenbeste atsegin ukan libroo hau

Balorazioa: Bigarren kasu honetan nahiz eta txioa formaltzat hartuz, baditu akatsak, eta nola ez dugun testua normalizatzen "libroo" hitza izen berezitzat hartuko du Matxinek eta emaitza ez da oso egokia aterako. Hau da, itzulpena ez da ongi geratuko.



Irudia 22 – Testu Formala Normalizatu Gabeko -Emaitzak

Ondorioa: Adibideekin argi utzi dugu testu formalek emaitza ona ekartzen dutela orokorrean baina noizean behin, testu informala izatetik gertu dauden txioen itzulpena ez da batere ona izango. Goiko grafikoan emaitza interesgarri batzuk aurkitu ditzakegu, non emaitza gehienak oso altuak diren eta arazoak dituen txioetan emaitzak ez baitira batere txarrak. Itzulpen bikain bat egin du, %100-a lortuz. Zoriz hartutako 20 adibiderekin emaitza onak lortzen ditugu orokorrean, eta nahiz eta itzulpen batzuk txarrak izan orokorrean %70 tik gorako itzulpen onak lortzen ditugula ikus dezakegu.

5.1.2 Testu formalak normalizatzaile arruntarekin

Normalizatzaile arruntak, lehen azaldu bezala, hitzak banan bana hiztegiarekin parekatzen ditu eta horietako bat ez badago hiztegian, antzekoak diren hitzez baliatzen da. Txioen testua formala denean ez dira honelako kasu asko ematen baina ia beti dago akats ortografikoren bat non ez badugu normalizatzen, itzulpena ez den ongi geratuko. Ikus dezagun ba, testu formalak itzultzean lortzen ditugun emaitzak.

Aurreko adibidearekin (Adibidea 2) hasiko gara, ea normalizatzaile arrunt honek arazoa konpontzen digun eta emaitza ulerkor bat lortzen dugun.

- **Adibidea 3:**

Txio originala: Y quien iba a pensar que me iba a terminar gustando tanto este libroo

Itzulpena: Eta pentsatzera zihoan me honenbeste atsegin ukan liburu hau

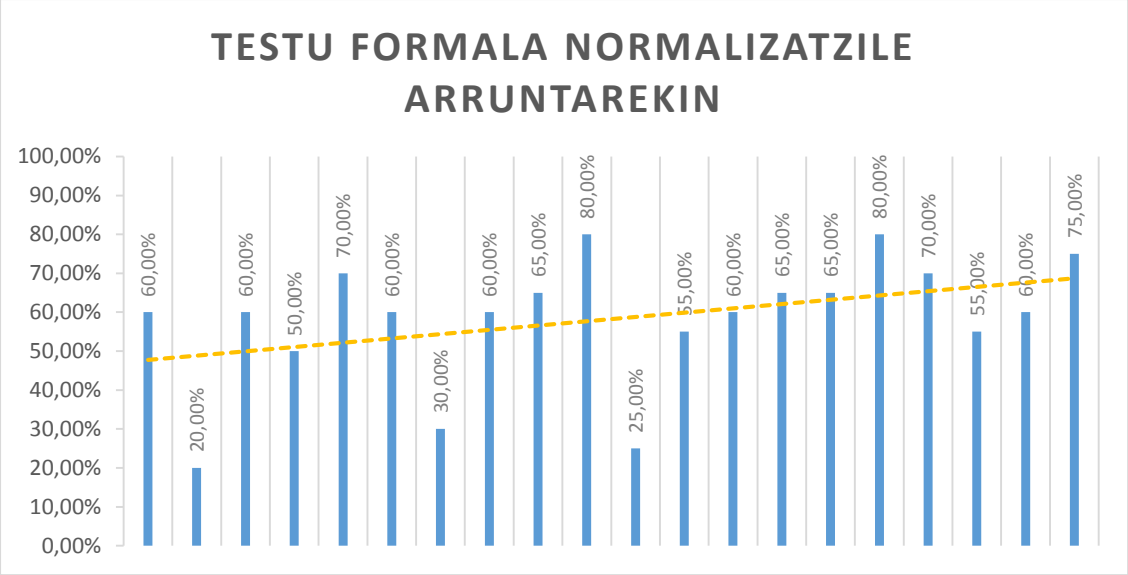
Balorazioa: Arazoa modu txiki batean konpondu du, baina ez da ez lortzea espero genuen hobekuntza handia. “Libroo” hitza bai konpondu du, baina ematen duenez oraindik “me” izenordaina ez du Matxinek ongi itzuli. Itzulpena beraz, zerbait hobea bai, baina ez asko.

- **Adibidea 4:**

Txio originala: Jugando a las cartas con mi abu :3

Itzulpena: Gutunei jokatzean nirea :3 zihoan

Balorazioa: Hasteko, “cartas”- “gutun” bezala itzuli du Matxinek, baina hori alde batera utziz eta “karta” jartzen duela pentsatuko dugu. Kasu honetan “abu” – “Abuela” hitzetik dator eta guk hala ulertzen dugu, baina normalizatzean hiztegian alderatuz, “abu” – “iba” bihurtu dugu eta beraz emandako itzulpena oso kaotikoa bihurtu da. Ez bagenuen normalizatuko, “Gutunei jokatzean nire Aburekin :3” itzuliko zuen Matxinek. Hau da, ulertuko genuen esaldia, non “abu” pertsona bat dela argi uzten baitu.



Irudia 23 – Testu Formala Normalizaile arruntarekin -Emaitzak

Ondorioa: Ikusi dugunez bi kasu nabari horiekin, testu formala denean gure normalizatzailea arruntak ez du ikaragarrizko hobekuntzarik ekartzen, eta beste kasu batzuetan itzulpena okerrago izateraino uzten du. Grafikoa ikusirik, banakako itzulpenetan, testua formala denean, EZ normalizatzea gomendatuko du aplikazioak. Baina, banan banako itzulpenak direnez eta txioak banaka aukeratuko direnez txioak, testua normalizatzeko aukera hor utziko dugu, beharrezkoa ikusten bada, gutxi baldin bada ere hobekuntza bat ematen baitu, eta esandako txio arraro edo kasu partikular batzuetan itzulpen garbiago bat lortzen baitugu.

5.1.3 Testu informalak normalizatzaile arruntarekin

Txio bereziak dira hauek, nahiz eta asko agertu. Txioak informala izateko lehen azaldu bezala, testuaren hitzetatik %60-ak hiztegian agertu behar du, traolarekin datorrena barne. Baina kasu hori ematen ez bada bi aukera ditugu: Esaldi luzeak eta motzak. Esaldi luzeetan, txio informalak, esaldi kaotikoak dira. Ez da ulertzen testu originalean ere zer esan nahi duten eta akats ortografikoz betea dago, beraz derrigor normalizatu behar dugu testua. Emaitzak ez dira inoiz bikainak izango, beraz hori jakinda, ahalik eta gehien hobetzea bilatuko dugu.

Esaldi motzetan berriz, posible da kasualitatez hitz bat gaizki egotea hiru hitzeko esaldi batean eta txio informal bezala hartuko dugu. Hori horrela izanda baliteke ez normalizatu nahi izatea.

Ondoren ikusiko ditugu lortutako emaitzak eta ondorioak.

- **Adibidea 5:**

Txio originala: Qué caloor !!

Itzulpena: Zein bero!!

Balorazioa: Adibide hau esaldi motz batena da, non nahiz eta hitz bakarra egon gaizki eta oso garbia izan zein den akatsa, %60 tik pasatzen da eta informala bezala hartuko dugu. Normalizatu ondoren, hitza asmatu du eta lortutako itzulpena oso ona da kasu honetan. Merezi izan du normalizatzeak.

- **Adibidea 6:**

Txio originala: Gomii no tontee por twitter locooo

Itzulpena: Jan zuen teontzitik ez dezan eraiki eroa

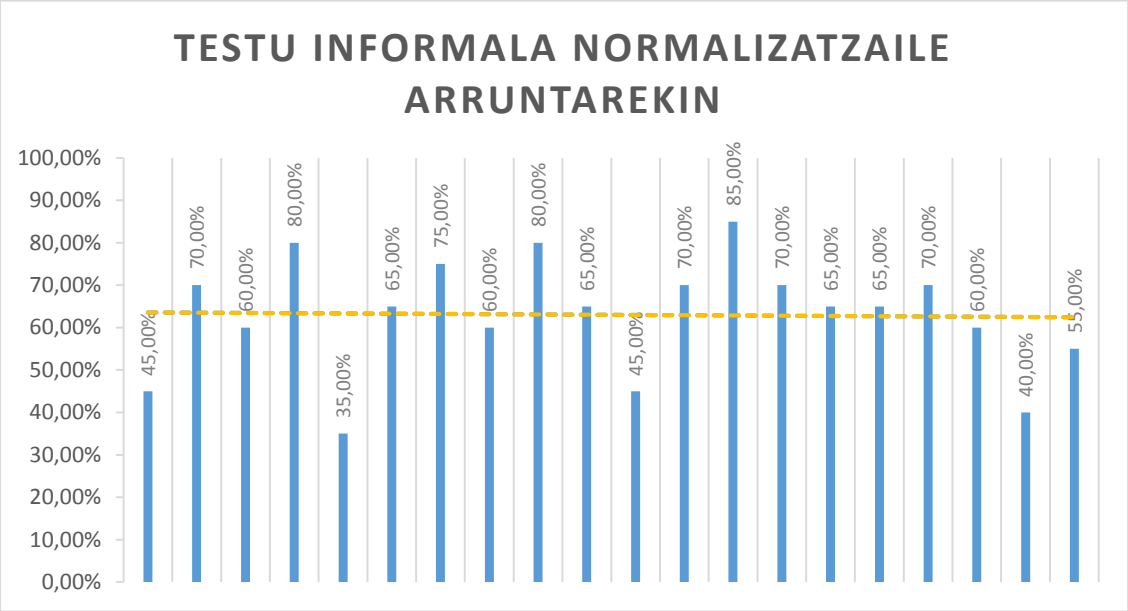
Balorazioa: Hemen adibide on bat dugu nola testu informal bat normalizatzean emaitza kaxkarrak lortzen ditugun. “Gomii” pertsona bat dela pentsatzen dut, beraz ez luke “Comió” hitzaz trukatu behar, hortik “Jan” hitzarekin hastea itzulpena, eta bestalde “twitter” hitza “tetera” bihurtu du, hiztegian begiratzean. Itzulpen txarra lortu dugu, ez baita ulertzen txioak esan nahi zuena. Baina testu berdina ez bagenu normalizatuko, lortutako itzulpena ere oso txarra da beti. Hau da, normalizatu ala ez, emaitzak txarrak dira ia beti.

- **Adibidea 6:**

Txio originala: Como odioo este libroo

Itzulpena: Liburu hau gorrotatzen dudan bezala

Balorazioa: Adibide honetan aldiz, oso emaitza ona lortu dugu normalizatzaile arruntarekin. Ez da beti emango, baina kasu berezi hauetan, non jatorrizko txioaren esaldiaren egitura ongi dagoen, baina esaldiak akats ortografikoak dituenean, itzulpen bikaina egingo du.



Irudia 24 – Testu Informala Normalizatzaile arruntarekin -Emaitzak

Ondorioa: Lortutako emaitza hoberenak %75-80 inguruan daude eta hori emaitza ona dela esan dezakegu testu informaletan, ez baitugu inoiz jakingo testu hauen benetako esanahia. Normalizatzaileak akats ortografikoak zuzentzen ditu batez ere, hau da, ez du lagunduko esaldi kaotikoekin eta traola edo aipamenez josiak daudenekin. Hori jakinik ikusten dugu lortutako emaitzak ez dire batere txarrak. Kasu txarrenean ere %40 inguruan baitaude, non itzulpena ez den ona, baina zerbait ulertzeko gai garen.

5.1.4 Testu informalak hitzez hitz itzuliak

Txio informalen itzulpena hobetzeko premisarekin egindako aukera da hitzez hitzekoa eta teorian hobekuntza bat ekarriko duela pentsatzen bagenu harritu egingo ginenen benetako emaitzez. Emaitza hoberenak esaldi luze eta egitura kaotiko bat izatean lortzen ditugu, aurreko adibidean ikusi dugun bezala normalizatzaileak lagundu ezin duen lekuetan.

Aurreko adibideekin probatuko dugu, nola ongi egituratutako esaldietan emaitza kaxkarrak lortzen dituen, eta ondoren esaldi luze eta kaotiko batekin emaitza hobeak ikusiko ditugu.

- **Adibidea 7:**

Txio originala: Como odioo este libroo

Itzulpena: Gorrotoa Hau Liburua

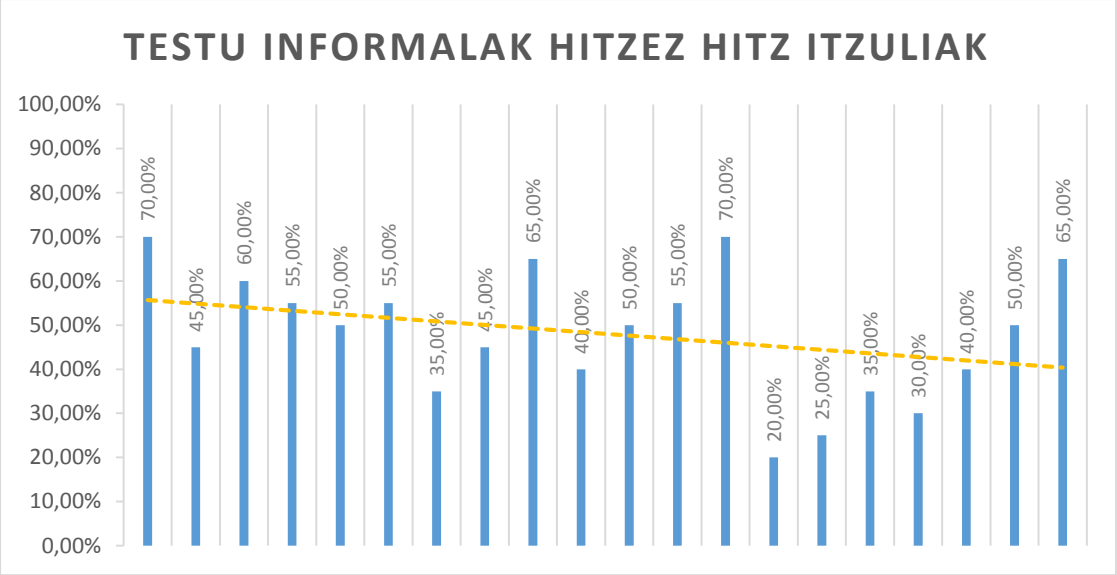
Balorazioa: Adibide honekin kasu txarrena erakusten dugu, non itzulpena hitzez hitz ondo egon arren, esaldiak ez baitu zentzurik. Jatorrizko txioaren esaldiaren egituraren arabera lortuko dugu emaitza ona ala kaxkarra. Kasu honetan "Como" hitza jan egin du itzultzaileak.

- **Adibidea 8:**

Txio originala: Gomii no tontee por twitter locooo

Itzulpena: Jan zuen Ez Eraiki dezan Teontzia Eroa

Balorazioa: 6. Adibidearen antzera, akats nagusiak "Gomii"-"Comió" eta "twitter"-"tetera" izan dira, non "tontee"-"monté" bihurtzean esaldia kaotiko bihurtzen da. Baina normalizatzailearekin bakarrik alderatzen badugu, hitzez hitz egin gabe, gehiago ulertzen da, Matxinek ez baitu esaldirik sortzeko aldaketarik egin, eta itzulpenak linealak diren. Beraz goiko aldaketak ondo egingo balitu honela geratuko litzateke esaldia hitzez hitz itzuli ondoren: "Gomii ez ergelkeriarik egin dezan twiter eroa". Ez da emaitza bikaina baina jatorrizko txioaren esanahia ulertzetik gertuago gaude.



Irudia 25 – Testu Informala Hitzez Hitz itzuliak -Emaitzak

Ondorioa: Gure aplikazioak eskaintzen dituen aukeretatik okerrenean gaudela ikusten dugu hemen. Sarrera datuak ez dira lagungarriak, baina normalizatzaileaz emaitza hobeak lortzen ditu orokorrean esaldiari estruktura bat ematen baitio itzuli ondoren. Hitzez hitzekoan ordea kasu berezietan bakarrik laguntzen digu, hau da, ez gehienetan, baina banan banako itzulpenean gaudenez, aukera hau kontuan izan behar dugu, 8. Adibidean ikusi dugun bezala, akats txikiak zuzendu ezkero emaitza tarteka ez baita txarra izango. Zorizko adibideetan ordea emaitza hoberena %70-ean dago, eta okerrena %20-ian. Ez dira emaitza onak.

5.2 Fitxategi eta txio Multzoen itzulpenak

Banakako itzulpenekin alderaturik, multzokatuta dauden txioekin arreta ez dugu jarri itzulpen bikain bat lortzean, baizik eta esaldiak ulerterrazak izan daitezen. Banan banako itzulpenetako emaitzak ikusirik hartu dugu multzo hauentzat onenak izan daitezkeen aukerak eta hortik lortutako emaitzak aztertuko ditugu hemen, baliozko metodo bat erabiltzen hari garen ala ez ondorioztatzeko.

5.2.1 Normalizatzailerik gabe

Fitxategietatik lortutako txioak multzo handitan datoz eta honek exekuzio denbora areagotzen du, baina onartzea dugun denboraren barnean emaitzak ikus ditzakegu. Txio hauek ez normalizatzailean bi motako testuak aurkituko ditugu, formalak eta informalak. Formalen kasuan ez dago arazorik, ez baitu emaitza asko aldatzen eta kasu batzuetan hobeak lortzen baititu, baina informalen aldetik, aplikazioan beti normalizatzen ditugu esandako arrazoiengatik, orain ordea gabe egingo dugu ondorengo emaitzak lortuz:

- **Adibidea 8:**

Txio originala: El que se rie del mal del vecino ... el suyo viene de camino ...

Itzulpena: Se gaitzeko auzokoaren riea ... harena bidenabar dator ...

Balorazioa: Emaitza kaxkarra, adibideetako batean, baina gehienak itxura berdina dute. Esan dugu Twitterren, gehienbat txio formalak daudela, eta gure bakarkako emaitzetan ikusi dugun bezala, txio formaletan normalizatze arruntak hobekuntza nabarmen bat eman digu. Hori dela eta fitxategiko datuak itzultzean eta gehiena formala dela uzte izatean, mota honetako itzulpenak bigaren lekuan jarriko ditugu.

5.2.2 Normalizatzaile arruntarekin

Fitxategietatik lortutako txioak multzo handitan datoz eta honek exekuzio denbora areagotzen du, baina onartzea dugun denboraren barnean emaitzak ikus ditzakegu. Txio hauek ez normalizatzailean bi motako testuak aurkituko ditugu, formalak eta informalak. Formalen kasuan ez dago arazorik, ez baitu emaitza asko aldatzen eta kasu batzuetan hobeak lortzen baititu, baina informalen aldetik, aplikazioan beti normalizatzen ditugu esandako arrazoiengatik, orain ordea gabe egingo dugu ondorengo emaitzak lortuz:

- **Adibidea 8:**

Txio originala: Minuto 25. Neymar, con un disparo cruzado, a punto de superar a Cillessen

Itzulpena: 25 minutua. Zenbatzen du, tiro bat gurutzatuarekin, gainditzeko zorian atera zitezen

Balorazioa: Itzulpena ez dago batere gaizki baina arazo batzuk ditu, Neymar eta Cillessen itzultzen saiatu da eta honek asko aldatu du txioaren esanahia. Baina arazo hau kenduta emaitza ontzat hartu dezake, beste kasu asko oso itzulpen txarra jaso baitute

5.2.3 Hitzez hitzeko itzulpena

Aukera hau inplementatua dago kodean, baina exekutatzean txio bakoitzaren testuko hitz bakoitzeko egiten du deia Matxin itzultzailera, eta honek denbora asko ematen du. Aukera honen emaitzak ikusitakoa ikusita testu informaletan bakarrik du onura eta kontuan izanda fitxategietako testuak hasiera batean bi modutako testuak izan daitezkeela, testu formal guztietan atzera pausu bat izango zen eta honelako txio multzo handietan ez du merezi denbora eskaintzea ondoren emaitzak okerragoak badira.

Ondorioa: Lortzen diren onura urriak direla eta eramaten dion exekuzio denbora kontuan izanik, metodo hau erabili merezi izateko bi hobekuntzak beharko lirateke, denbora aldetik azkartzea (seguruena APIan aldaketak eskatzen) eta hitzez hitz itzuli beharrean sintagmaz sintagma itzulpena lortzen.

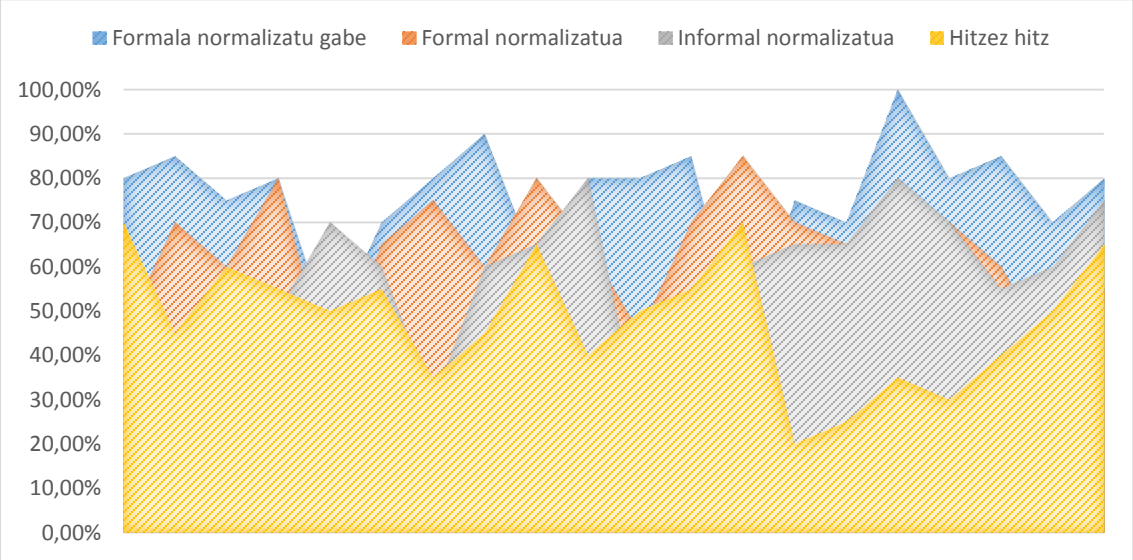
5.3 Ondorio orokorra

Itzultzaileak ematen dituen lau aukeren artean, formal ala informal, emaitza onak lortzeko gai gara, sarrera txioa edozein dela medio.

Testu formaletan txio gehienak ondo itzultzen ditu normalizatzailerik gabe, baina gaizki itzulitakoetan normalizatzaileaz itzultzeak lagundu dezake hobekuntza txiki baten moduan.

Bestalde testu informaletan normalizatzailearekin emaitza onak lortzen ditugu, baina arazoak sortu daitezkeen txioekin hitzez hitzeko itzulpena proba dezakegu, non ikusi dugun, emaitza batzuekin lagundu dezakeela.

Orokorrean, proiektuan zehar egindako probekin emaitza onak lortu ditugu, bai Matxinen emaitza onagatik edo bai gure aplikazioak ematen dituen aukeragatik, non ondoriozta dezakegu aukera denak erabilita emaitza on bat lortzeko aukerak handiak direla.



Irudia 26 – Itzultzailearen aukera denen emaitzak

6 .

Ondorioak

"Azken kapitulu honetan proiektu hasieran aurreikusitako eta amaieran lortutako on"

6.1 Helburuak berrikusten

6.1.1 Lorpen maila

Proiektua amaiturik egonda, hasiera batean bezeroarekin adostutako helburuak beti ditugun eta ze mailetan ikusi behar dugu.

- Twitter API-a erabiliz: ✓
  - Txio bilatzaile bat ondorengo kategorietan bilatzeko aukerekin:
    - Aipamenak. ✓
    - Traola edo Hashtag-ak. ✓
    - Erabiltzaile jakin baten txioak. ✓
    - Hitz edo testu jakin bat. ✓
  - Txioei buruzko informazioa atera:
    - BerTxioa den kasuetan testu originala lortu. ✓
  - Txioaren hizkuntza bereiztu ✓
  - Itzuli nahi den hizkuntza eta nora itzuli nahi duen aukerak eman. ✓
  - Txioak banaka edo bilaketaren emaitza denak itzultzeko aukera eman. ✓

- Fitxategiak erabiliz:
  - Fitxategietako testua katalogatu Twitter aplikazioko txio bat izango balitz bezala. ✓
  - Txioen buruzko informazioa atera. ✓
  - Itzuli nahi den hizkuntzen kontrola eraman. ✓
  - Emaitzen fitxategian gorde itzulitako testuak. ✓
- Txioen itzulpena erderatik euskarara:

- Txioan dauden elementu desberdinak bereiztu itzuli nahi den testutik eta gorde itzuli ondoren berriz idazteko. ✓
- Testu gordina tratatu eta formala ala informala den erabaki.
  - Txio formaletan normalizatzeko aukera eman, hau da, erderako hiztegian ez dauden hitzak ordezkatu daudenekin. ✓
  - Txio informaletan hitzez-hitz begiratu hiztegian agertzen den ala ez eta proposamenetatik lehenengoaz ordezkatu. ✓
- Testua sintagmetan zatitu eta itzultzailera zatika pasatzeko aukera. ✖
- Itzulpenen emaitzak fitxategietan gordetzeko aukera inplementatu. ✓

Bezeroak eskatutako dena egin da bigarren normalizatzailea izan ezik. Hobeto azalduta, Mashup-aren hiru helburu nagusietatik sarrera eta irteera datuen kontrola inplementatu dugu behar bezala, erabiltzailearen interfaze grafikoa ere erabilerarako garatua izan da eta API eta web-zerbitzuak maneiatzen dituen aplikazioa ere inplementatu dugu. Azken honen barnean bezeroak testua normalizatzeko aukera gehiago garatzeko eskaera egin zuen baina ezin izan dugu lortu bi arazo nagusi gatik.

6.1.1.1 – Normalizatzaile berriak ez sortzearen arrazoia

Bezeroarekin bilerak egin ahala eta aplikazioaren nondik norakoak erakustean, normalizatzaile arruntaz gain, beste bi normalizatzaile garatzeko eskaera egin zuen. Normalizatzaile hauek inplementatzen saiatu ginen proiektuan, baina arazoak izan genituen inplementazioan eta denbora urria zela eta atzera uzteko erabakia hartu zen.

Bi normalizatzaileak honakoak ziren:

- **Sintagmez banandu eta itzuli**: Freeling [27]izan zen honetarako erabili nahi izan genuen API-a, non berak azterketa morfologiko bat egingo zuen eta ondoren guk itzultzaileari zatika bidaliko genion, lana errazteko eta emaitza hobeak lortzeko.
- **Normalizatzaile hobeagoa**: Bezero berak gomendatutako normalizatzaile bat zen hau, **Foma** izenekoa. Interneten iritzi onak lortzen dabil.

Probak egin ziren biekin, baina inplementazioan arazoak eman zituzten. Lehenak, Freeling-ek, erabileran moztu gintuen, non akatsak agertzen ziren eta dokumentazio falta dela eta ezin izan genuen jarraitu erabiltzen. Foma-ri dagokionez, konpatibilitate arazoa agertu zen, arazoak ematen baitzituen win32 –rako konpilatzerako garaian eta honen dokumentazio urriak ez zuen laguntzen arazoa konpontzerako garaian.

Arazo hauek zirela medio, eta kontuan izanik proiektuak sartzen hari ginen ordu kopuruak maximotik gertu zeudela, ez inplementatzeko erabakia hartu zen, eta etorkizunari begira, aurrerago inplementatzeko aukera bezala utzi ditugu.

[27] FreeLing 3.1 analizatzailearen Demoa [Link]

6.1.1.2 – Gehitutako hobekuntzak

Aplikazioren garapena aurrera joan heinean Mashup metodologia jarraituz, ataza bat egin bitartean beste bat teilakatu dezakegu eta horrek ideia berriak sortzea dakar. Ataza bat bukatu aurretik ideia hauek probatzen joan gara, inplementazioan duten eragina eta lortzen ditugun emaitzak ikusirik beraiekin jarraitu ala ez erabakitzeko. Batzuk bide ertzean utzi ditugu baina beste batzuk aurrera atera dira.

Orokorrean hobekuntza txikiak dira, programa bizkorrago joan dadin, baina orokorrean hobekuntza handienak txio bilatzailean daude, eta bertako txioak itzultzeko ematen dituen aukerak.

Bezeroaren ideia proiektu hasieratik txio itzultzaile bat bakarrik ez, txio bat modu anitzetan itzultzeko erraminta bat sortzea zen, baina proiektua aurrera joanda txio bilatzailea ere bere helburuetan sartu zuen, guk demoan erakutsi ondoren. Gure asmoa txio bilatzailearekin, fitxategietako prozesua arintzea zen, non bertan txioak banaka itzuli ahal ditzakegun. Bilaketa txio denak itzultzeko aukera ere eskaintzen dugu, bilaketako emaitza guztiak interesatzen baldin bazaizkio.

Txio bilatzaileak Twitter API erabiltzen du, baina kodean antolatu dugun moduan, konexioa behin ireki eta bilaketak azkar egiten ditu. Proba bezala, bezeroari bileran erakutsitako lehen demoarekin alderatuz ia bi segundo azkarrago dabil orain.

Azkartasuna dela bide, txio identifikatzaile asko dituen fitxategiak hitzez hitz itzultzeko aukera ere inplementatu dugu, baina nahiz eta programan idatzia egon, ez diogu aukera ematen erabiltzaile arruntari hau exekutatzeko, exekuzio denbora gehiegi eramaten baitu eta momentu batzuetan, web-zerbitzuaren eskarietatik.

6.2 Proiektuan emandako denboraren

Proiektuaren **ordu kopuru maximoa 450 ordutan** jartzea planteatu zen eta egindako estimazioa, atazen denborak kontuan hartuz, denera 425 ordukoa [28]zen.

Ataza	**Aurreikusitako iraupena** *(orduak)*	**Benetan emandako iraupena** *(orduak)*
Plangintza	25	20
Prestakuntza	40	35
Diseinua	50	55
Inplementazioa	215	255
Probak	15	25
Dokumentazioa	80	85
Denera	425	**475**

Goiko taulak adierazten duen bezala egindako estimazio gehienak ez genituen ondo aurreikusi, eta denbora gehiago eman digu hauek betetzen. Ondoren azalpena:

- ✓ **Plangintza**: Aurreikusitakoa baina denbora gutxiago eman digu, honek ez du esan nahi okerrago planifikatu dugunik. Arrazoia da, hasieratik ikusi genuela proiektuak eboluzionatu egingo zuela, Mashup metodologia medio. Plangintza beraz idazterakoan helburua ez zen izan dena perfektu antolatzea, baizik eta lanean hasteko oinarri bat izatea.
- ✓ **Prestakuntza**: Bai Twitter API-a eta bai Matxin itzultzailearen erabilera prestatzeak bere denbora eman zigun, baina denera 15 ordu besterik ez ziren izan. Beraz aurreikusitako denbora oso gaizki kalkulatu zen, bi elementu horietan pentsatu baitzen bakarrik. Amaieran, ordu kopurua 35 orduraino igo ziren, bai programazio, bai itzultzaile automatikoei buruzko atalei buruzko informazioa bilatzen eta amaieran proiektuan batu ezin izan diren bestelako normalizatzaile eta API ezberdinen erabilerak aztertzen.
- ✓ **Diseinua**: Honelako dimentsioko eta ordu kopuruko proiektu bat diseinatze ez da gauza erraza izan eta espero ez genituen ordu kopuru handiago bat sartu behar izan dugu dena antolatzen. Horretako arrazoia Mashup metodologia aztertu aurretik beste bide batzuetatik joan bait ginen baina Mashup-era pasatzeko erabakia azkar hartu zen.
- ✓ **Inplementazioa**: Proiektuaren garapenean aurre ikusten zen emango zela denbora gehien eta ondoren, errealitatean ala izan da. Ordu kopurua estimatua baino gehiago izan da, baina egia esan honelako ordu kopuru handi batean, 40 ordu gehiago sartzea ez da

[28] A Eranskina: Denboren planifikazioa

izugarrizko aldaketa. Pozik gaude proportzioan, asmatu bait dugu lan kopuru gehiena bertan emango genuela ikustean.

- ✓ **Probak**: Berriz ere espero zena baino denbora gehiago eman dugu. Ez zen aurreikusi itzultzailearen emaitza ezberdinak ikusteko egin behar izango genituen proba guztiak, eta beraiekin Twitterren aurki ditzakegun txio motak bilatu eta probatzeak ere bere denbora eskatu digu. Ordu kopurua ez da handia izan, baina beharrezkoa ikusi ditugu, bezeroari kalitate maila bat bermatzeko eta aplikazio grafikoak, besteak beste, arazorik ez emateko aurki ditzakegun etorkizuneko kasu arraro batekin.
- ✓ **Dokumentazioa**: Azkeneko atal hau aurre ikusten zen, ordu asko emango zituela, eta horregatik jarri genuen mugarri bezala denborarekin idazten hastea. Mugarriak betetzea erraza izan ez den bezala, eskainitako orduak beharrezkoak ikusten dira produktuaren eta hiru emangarrien kalitate maila igotzeko.

Orokorrean, egin genuen orduen estimazioa altua izan zenez, benetakoarekin alderatuz ez daude ordu askoren diferentzia (ordu kopuruak borobilduak baitaude) eta azken finean ia amaiera arte ezin izan baita ataza ezberdinetako denbora murriztu. Denbora murriztu genezakeen ataza bakarra inplementazioa zela erabaki genuen, beti ere proiektuari dedikatutako ordu kopuru maximoa ez pasatzeko. Ez dugu lortu hau betetzea, zeren bezeroaren eskaerak amaitu nahi izan baititugu baina ordu maximo hau pasa ondoren garapenean falta ziren atazak ezin izan ditugu aurrera eraman. Hau dela eta ezin izan da bigarren normalizatzaile bat inplementatu besteak beste.

Ondorio bezala, konturatu behar gara proiektua hasi eta lehenengo egindako atazak izan direla estimatutakoa baino denbora gutxiago eman digutenak, proiektuaren amaiera urruti ikusten baitzen eta oraindik hobetzeko denbora soberan izango baitzen, baina proiektua aurrera joan ahala eta inplementazioan sartutako ordu kopuru altuarekin amaierako atazei lehentasuna eman behar izan diogu. Hau lortzeko esan beharra dago, emandako orduen jarraipen on batek lagundu duela, beti ere ordu maximoa kontuan izanda eta garapen erdira iristean konturatzen baitgara ez digula denbora emango dena egiteko. Erabakia hauek proiektuaren bizitza lerroaren erditik aurrera hartu ezkero emaitza txarrak lortzeko bakarrik balio dute. Gure kasuan emaitza hoberik lortu ezean, emaitza txarra lortzea ekiditeko lagundu digu.

6.3 Ikasitako lezioak

Gradu amaierako proiektu bat bezala aurkeztu zidaten honako lan hau, eta bertan premisa bezala Informatikako Graduan emandako lau urteetan ikasitakoa praktikan jartzeko aukera izango zela esan zidaten. Esperientzia gutxiko pertsona izanda proiektuetan, honelako batek begiak ireki dizkit. Lan gogorra eta dedikazioa izango dira ikasitako lezioetan aipatuko nukeen lehenengoa.

Edozer lortzeko gai garela erakusten digu honek, eta inor ez dela jakinda jaio. Proiektuaren atal eta ataza bakoitzeko informazio bila ikusten nintzen, Graduan zehar ikasitakoarekin baliatuz eta gehiago landuz atal horri buruz nuen ideia guztiak. Bai eguneroko gaitan, non programatzeko izan ditudan zalantzekin, bai inoiz jakin ez ditudan gauzetan, itzultzaile automatikoen misterioak adibidez, Internet eta liburuetara jo behar izan dut.

Beti jakin izan dudan arren, honelako ordu pila gai edo materia batean pasatzean, konturatu nahiz benetan inguruko pertsonek eskaintzen duten laguntzaz. Eta hau ez doa esperientzia edo jakintza gehiago dutenei buruz bakarrik. Gradu amaierako proiektu hau nire lana izan da, baina azken lau urte hauetan izan ditudan ikaskide eta maisuen esperientziak Internet eta liburuak baina laguntza handiago izan dira, txikienetik handienera. Beti eman izan diot garrantzia talde lanari, bai lan munduan, bai bizitzan zehar, eta nahiz eta bakarkako proiektu bat izan, eta lan guztia azken finean norberak egin, ingurukoekin kontatzeak bai ideiak aipatzeko edo izandako buruhausteez hitz egiteko.

Proiektu honi dagokionez, ikasitako lezioak, esperientzia gutxiko pertsona gehienek ikasitakoak dira. Plangintza on baten garrantzia, denboren antolaketa eta jarraipen ona, aldez aurreko diseinuen azterketa eta hartutako lehen erabaki horiek proiektu osoan duten eragina. Baina gauza hauek, Graduan zehar egindako proiektu txiki guztietan ikusi ditugu eta aurre ikusten nuen berriz ere gertatuko zirela. Honek ez du esan nahi gai naizenik hauek ebitatu eta nire alde jartzeko. Esandako puntuetan egindako akatsak, onak dira, azken finean oraindik ikasten gaudelako eta etorkizunean, aurrean tokatzen zaigun proiektu orotan kontuan izan beharko baititugu. Honek ondoren leziora eramaten nau.

Akats, huts, edo erroreak orokorrean maneiatzeko gaitasuna. Esan bezala, proiektu hasieratik gaude jakinaren gainean, gertatuko direla, baina plangintza duin batekin aurreikusi ditzakegu arazo handienak, eta batez ere ondoren datozen ataza eta mugarrietan ahalik eta eragin gutxien lortzea da helburua. Talde lanetan ere ikasten da hau, baina bakarkako proiektu batean, arazoari nola aurre egin norberaren esku dago.

Txio itzultzaileari dagokionez, pozik esan dezaket bilatzen nituen gaitasunak lortu ditudala eta asko ikasi dudala bai Mashup metodologia duen proiektu bat erabiltzen eta API ezberdinak, gure kasuan batez ere Twitterren API-a, erabiltzen. Amaierako produktuan integraturik ez egon arren API ezberdin asko erabili eta probatu ditugu, eta honi buruz esan dezakedan lezioa da, non bi lezio ezberdin dira egia esan, dokumentazio duin baten beharra. API hauek denak dokumentazio urri bategatik utzi ditugu kanpoan gehien bat, ez baitute laguntzen izan ditzakegun arazoekin martxan jartzean. Aldi berean esan dezaket, honek erakutsi didala kontuan izaten beharrezkoa den heinean kodea komentatzen eta dokumentazio duin bat idazteko motibatzen.

Azkeneko lezioa, norberaren lanarekin gustura egotea da. Bezeroak beti jarraituko du eskaerak egiten, batez ere zuk ala uzten badiozu (Mashup metodologiaren ahulezietako bat) eta nahiz eta zure helburu nagusi bat bezeroaren onespena lortzea izan behar duen, azken produktuak norberaren seilua eramango duenez, egindako lanarekin gustura egon behar garela ikasi dut.

Proiektua beti amaituko da, eta ez dago bide egoki bakarra horra iristeko emaitza onekin. Bide asko daude eta ez dugu beti ondo aukeratuko bide hori, baina kontuan izan behar dugu hurrengo baterako ez jarraitzen berriz ere bide horiek eta beraiengandik ikasi dugunaz baliatzen etorkizunean bide berri batzuk aukeratzean. Emaitza da garrantzitsua eta nahiz eta bide egokiak aukeratzeak laguntzen duen, txarretatik ere asko ikasi daiteke Gradu Amaierako Proiektu batean.

6.4 Itzultzailea partekatzen

Proiektu hau eta bere emangarri guztiak Interneten egongo dira atzigarri aurkezpena egiten den egunerako. Kodea eta bere interfaze grafikoa edonork aldatu ahal izango du, normalizatzaile hobeak edo hizkuntza gehiago gehitzeko.

https://www.dropbox.com/sh/hx58txjb5b79mq9/AADaqW2PHkKZg-prIygqP4PRa?dl=0

6.5 Etorkizunera begira

Bezeroak ezarritako helburuak betetzen ditu aplikazioak, baina hori ez da bere benetako ahalmena. Aplikazioa etorkizunari begira diseinatu da, non kodean aldaketa handirik gabe gehitu ahal ditzakegun bestelako eginkizunak: hizkuntza ezberdinak gehitu, normalizatzaile ahaltsuagoak erantsi eta fitxategi mota berriak irakurtzeko aukerak gehitu.

6.5.1 Hizkuntza berriak gehitu

Aukera berri hauekin aplikazioa asko handitu daiteke, beti ere norberaren beharretarako egokitua eta aplikazioak bere lana egingo du txioak itzultzen.

Hizkuntza berri bat gehitzean, kontuan izan behar dugu itzultzaile bat aurkitu behar dugula lehenbizi eta ondoren hiztegi bat berarekin parekatzeko. Itzultzaileak web-zerbitzu bat izateak asko lagunduko luke non aplikazio barruan inplementatu beharrean itzultzailearen pisu guztia kanporatu egingo baitu.

Hizkuntza berri bat gehitzeko 2 elementu nagusi behar ditugu: Hiztegi bat eta itzultzaile bat, guk nahi dugun hizkuntzatik, guk itzuli nahi dugun hizkuntzara itzuliko duena. Kodean hizkuntza gehitzerako orduan aldaketa gutxi egin behar dira. Hizkuntzen kontrola egina dago, beraz hiztegia gehitu proiektuaren barneko karpetetan, eta kodean jarri zein den bere path-a. Ondoren itzultzailea integratu beharko da, non gomendio bezala, web-zerbitzu bat badu honetaz baliatzea gomendatuko genuke, aplikazioari pisua murrizten baitio.

Kontuan izan, beti ere , hizkuntza asmatzeko txioetatik gure aplikazioak Twitter API-aren baliabideak erabiltzen dituela. Gehitu nahi den hizkuntza Twitterrek bereizi dezakeen bat dela begiratu behar da, eta ezezkoa bada, hizkuntzak asmatzeko gai den API bat instalatu beharko da.

6.5.2 Normalizatzaile berria gehitu

Gure aplikazioak normalizatzaile ezberdinak gehitzeko aukera ere errazten du, kodea egokitua baitago honekin laguntzeko. Kodean, txio baten testua lortu eta hau tratatu ondoren, itzultzailera bidali aurreko prozesua normalizatzailea izanik, aski da funtzio bat sortzea non sarrera bezala tratatutako testua den eta normalizatu ondorengo testua irteera datu izatea. Honek bai esaldi osoa bai hitzez hitz itzultzerakoan lagunduko digu.

Normalizatzaile berri bat gehitzeak kodearen zati handi bat aldatzea eskatzen du, beraz etorkizunean hau egitera bagoaz kontuan izan beharreko datuak honakoak dira.

- ✓ Testua tratatua egon behar du.
- ✓ Normalizatzaileak Internetetik zerbait atzitu behar badu aplikazioaren exekuzio denbora handituko du.
- ✓ Normalizatutako testua kodetu egin behar da Matxin itzultzailera bidali aurretik.

6.5.3 Burutu ezin izan diren ideiak

Bi ideia nagusi sortu dira aplikazioaren garapenean. Ideia hauek ez ziren bezeroarekin hitz egin, beraz ez zen ikusi beraien inplementatzeko beharrik, baina etorkizun baterako oso interesgarriak izan lirateke ondoren aurkeztutako ideiak:

- **Txio itzultzaile automatikoa Twitter bidez**: API-ak txioak idazteko aukera ematen du, hori dela eta bururatutako ideia da, bost minutuko tarteetan, Twitterren traola bat -adibidez "#TxioIt" itxurako bat- bilatzea eta hau duten txioak itzuli sortu dugun aplikazioaren, ondoren, fitxategietan gorde beharrean berriz ere Twitter kontu baten bitartez, txio hauek argitaratzea Twitter bertan. Honela beharra denean, eta Twitter aplikazioa bakarrik eskura dutenek, bost minutuko tarte batean itzultzaile automatiko bat izango zuten.

- **Aplikazioa hodeian jarri**: Guk Matxin API-aren web zerbitzua atzitzen dugun bezala, gure aplikazioa ere Internetetik atzigarri jartzea da ideia. Honela ez genuke erabiltzailearen sistema eragilearen bateragarritasunaz arduratu behar. Aplikazioaren instalazioa eta Perl bezalako bestelako aplikazioak ere ez lirateke instalatu beharko. Ideia hau aurrera eramateko proiektuaren dokumentazioak eta bertan agertzen diren ideia eta diseinuak lagunduko lukete baina kodearen zati gehiena aldatu beharko litzateke eta hasieratik berriz ere erabaki batzuk hartu.

A. Eranskina

Proiektuaren Plangintza

"Eranskin honetan proiektua modu antolatuan eramateko prestatutako plangintza agertuko da. Hasiera batean genituen aurreiritziak eta ondoren hartutako erabakiak. Zati honetan proiektuaren helburuak, atazak, mugarriak… aurkitu ahal ditzakezu."

A.1 Proiektuaren helburuak

Proiektuaren helburua, itzultzaile bat izateaz aparte, itzultzeko eman daitezkeen aurre pausu ezberdinekin probak egiteko erremienta bat da. Horretarako aplikazioak ahalik eta aukera gehienak eskaintzea nahi izan dugu ondoren emaitzak ebaluatzeko.

Bere osagaiak honakoak dira:

- Twitter API-a erabiliz:
  - Txio bilatzaile bat ondorengo kategorietan bilatzeko aukerekin:
    - Aipamenak.
    - Traola edo Hashtag-ak.
    - Erabiltzaile jakin baten txioak.
    - Hitz edo testu jakin bat.
  - Txioei buruzko informazioa atera:
    - BerTxioa den kasuetan testu originala lortu.
  - Txioaren hizkuntza bereiztu
  - Itzuli nahi den hizkuntza eta nora itzuli nahi duen aukerak eman.
  - Txioak banaka edo bilaketaren emaitza denak itzultzeko aukera eman.

- Fitxategiak erabiliz:
  - Fitxategietako testua katalogatu Twitter aplikazioko txio bat izango balitz bezala.
  - Txioen buruzko informazioa atera.
  - Itzuli nahi den hizkuntzen kontrola eraman.
  - Emaitzen fitxategian gorde itzulitako testuak.
- Txioen itzulpena erderatik euskarara:
  - Txioan dauden elementu desberdinak bereiztu itzuli nahi den testutik eta gorde itzuli ondoren berriz idazteko.
  - Testu gordina tratatu eta formala ala informala den erabaki.
    - Txio formaletan normalizatzeko aukera eman, hau da, erderako hiztegian ez dauden hitzak ordezkatu daudenekin.

    - Txio informaletan hitzez-hitz begiratu hiztegian agertzen den ala ez eta proposamenetatik lehenengoaz ordezkatu.
- Testua sintagmetan zatitu eta itzultzailera zatika pasatzeko aukera.
- Itzulpenen emaitzak fitxategietan gordetzeko aukera inplementatu.

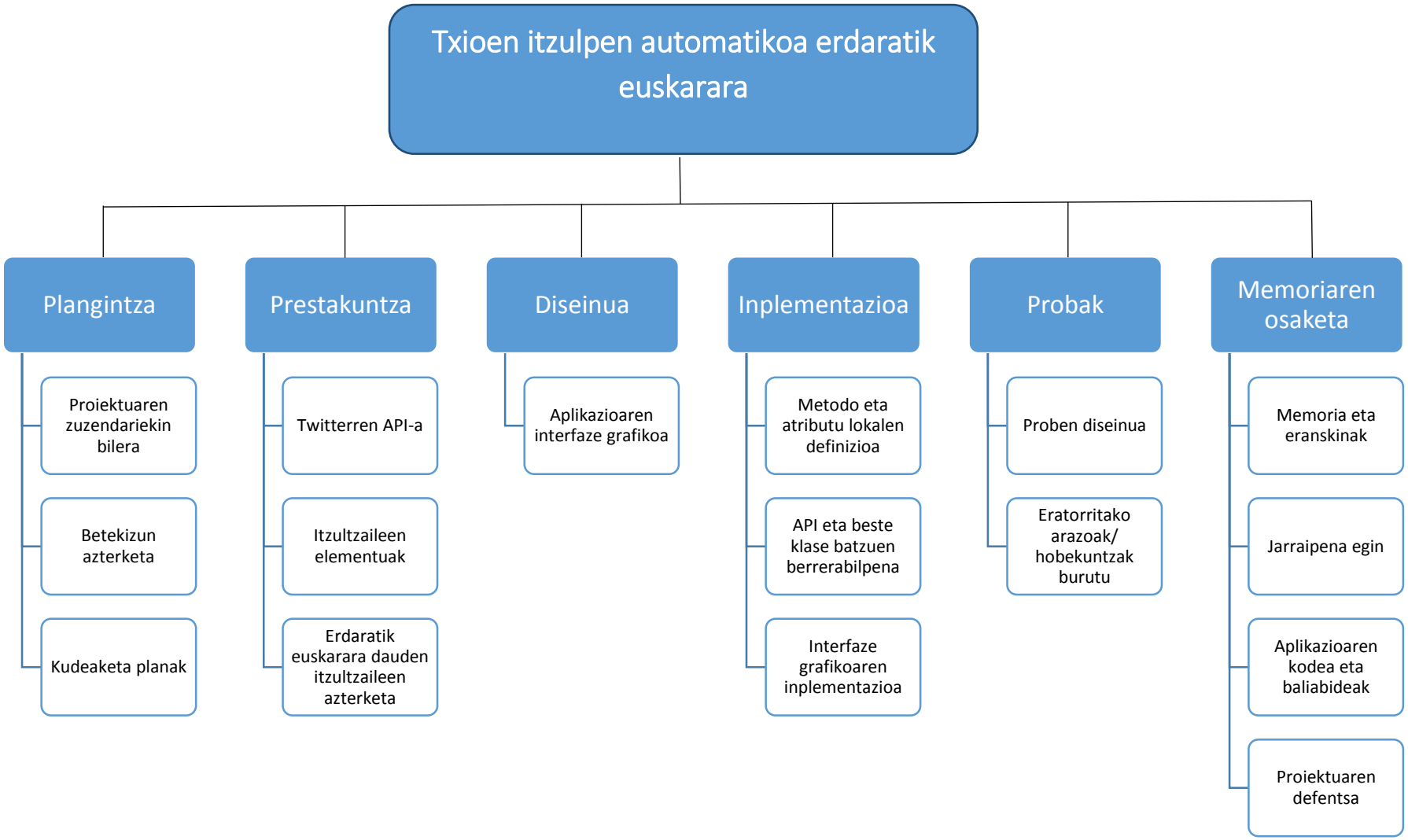
A.2 Emangarriak

Proiektu honen parte diren hiru emangarriak ondorengoak izango dira:

- ✓ **Txio itzultzailea**: Edo sortutako aplikazioa. Aplikazio hau bezeroaren esku utziko da eta baita sortzeko erabili den kodea ere. Proiektuaren zati hau Interneten jarriko da baita ere eskuragarri aplikazioa hobetu nahi duen guztientzat.
- ✓ **Dokumentazioa**: Proiektuko informazio guztia agertzen da bertan, plangintzatik, erabili diren erremientak, hartu diren erabakiak , diseinua eta lortutako emaitzak besteak beste.
- ✓ **Erabiltzailearen eskuliburua**: Aplikazioa nola erabili eta ematen dituen aukeren gida bat izango da emangarri hau, non bere instalaziorako behar diren programak eta nola exekutatu agertuko diren besteak beste.

Emangarriak 2015-ko Azaroak 3-an egon beharko dira eskuragarri bezeroarentzat, eta bi aste beranduago igoko dira Internetera.

A.3 LDE diagrama



A.4 Atazak

Proiektua aurrera eramateko honako atazak eman dira aurrera:

- Plangintza:
  - Proiektuaren zuzendariarekin bilera egin eta betekizunen azterketa egin. Bilera bat baino gehiago behar izan ziren proiektuaren behar ezberdinak finkatzeko.
  - Ataza bakoitza aurrera eramateko beharko den denbora eta material estimazioa egin.
- Prestakuntza:
  - *Twitterren API-a* : Informatu Twitterrek denon eskura uzten duen API-ari buruz jakin beharrekoa; bere politika eta bete behar diren baldintzak, kode mailan dituen baliabideak eta honen integrazio eta erabilera ikasi.
  - *Itzultzaileen elementuak*: Itzultzaile bat egiterako orduan jakin beharrezkoak diren elementuak; testuak normalizatu, formal eta informalen tratamendu bereziak eta hitzez-hitz edo sintagmez itzultzeak ematen dituen abantailen arteko diferentzia ikasi.
  - *Erdaratik euskarara dauden itzultzaileen azterketa*: Erdaratik euskarara dauden itzultzaile desberdinak aztertu eta proiekturako onena dena aukeratu. Gure kasuan Matxin erabiliko dugu, beraz honi buruzko erabilera eta instalazioa ikasi.
  - Erabiliko diren bestelako API-ak erabiltzen eta integratzen ikasi.
- Diseinua:
  - Proiektua hasi aurretik erabiliko diren metodoen eta datu egituren azterketa egin, eta erabaki nagusiak hartu.
  - Erabiliko diren tresna nagusiak aukeratu.
  - Proiektuari bat datorkion metodologia aukeratu.
  - Aplikazioak izango duen interfazearen diseinua egin bai proiektuan zehar erabiliko diren demo eta azken produkturako.
- Inplementazioa:
  - *Metodo eta atributu lokalen definizioa*: Aplikazioa inplementatzerako garaian aurreikusitako metodo eta atributuak definitu, itzultzailearen helburuak kontuan izanda.
  - *API eta beste klase batzuen berrerabilpena*: Interneten kodigo askeko API eta klaseak bilatu eta aztertu zein izan daitezkeen proiektuan zehar erabili ahal izango ditugunak, beti ere bakoitzaren politika eta betebeharrak kontuan izanik.
  - Diseinua jarraituz, interfaze grafikoa inplementatzen joan.
- Probak:
  - *Proben diseinua*: Proiektuaren zuzendariekin adostu beharrezkoak izango diren bertsioak eta erakusketarako erabiliko diren demoak.
  - Probetan emandako arazoak konpondu eta bururatutako hobekuntzak aplikatu azkeneko bertsioari.

- Memoriaren osaketa:
  - Memoriako dokumentua osatzen joan proiektua aurrera doan heinean.
  - Jarraipena egin egindako plangintzarenak, hau modu egokian garatu dela ziurtatzeko.
  - Beharrezkoak izan diren kode zatiak eta baliabideak jaso eta beharrezkoa ikusten bada eranskin batean gehitu memoriari.
  - Proiektuaren defentsa prestatu proiektuak eman duen denboraren eta atazen gainbegiraketa bat eginez.

A.5 Denboren planifikazioa

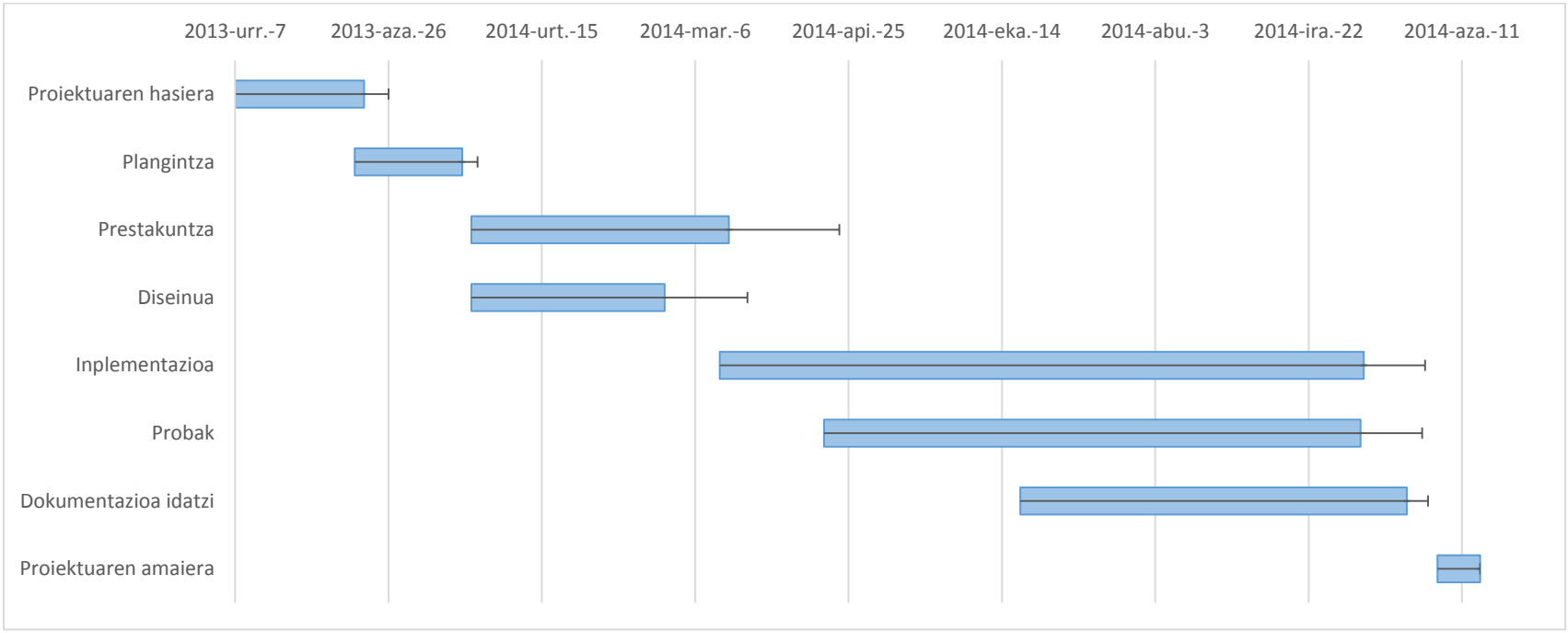
Atal honetan proiektua aurrera eramateko kalkulatzen diren ordu kopuruak ataza bakoitzeko aurkeztuko ditugu. Proiektuaren luzapena eta lan maila kontuan izanik eta emangarrien garrantzia ikusirik, egileak proiektuak lantzerakoan duen esperientzia gutxiak orduen kalkulu zabalago bat egitera eraman digu. Proiektuaren **ordu kopuru maximoa 450 ordutan** jartzea erabaki da beraz, ondorengo atazetan banaturik:

Ataza	**Iraupena** *(orduak)*
Plangintza	25
Prestakuntza	40
Diseinua	50
Inplementazioa	215
Probak	15
Dokumentazioa	80
Denera	425

Proiektuaren amaiera datu aldatu egin genuen denbora falta zela eta, honekin beste 2 hilabete lortuz lana amaitzeko. Ordu kopuru hauek, proiektuaren hasiera eta amaiera datak ikusirik honela banatu ditugu, lan karga handiena 2014-ko udararako utziz.

ID	Ataza	Hasiera data	Aurreikusitako denbora *(Egun kopuruetan)*	Amaierako data	Errorea *(Egun kopuruetan)*
GAP-1	Proiektuaren hasiera	2013-urr.-7	42	2013-aza.-18	50
GAP-2	Plangintza	2013-aza.-15	35	2013-abe.-20	40
GAP-3	Prestakuntza	2013-abe.-23	84	2014-mar.-4	120
GAP-4	Diseinua	2013-abe.-23	63	2014-ots.-21	90
GAP-5	Inplementazioa	2014-mar.-14	210	2014-urr.-11	230
GAP-6	Probak	2014-api.-17	175	2014-urr.-20	195
GAP-7	Dokumentazioa idatzi	2014-eka.-20	126	2014-urr.-26	133
GAP-8	Proiektuaren amaiera	2014-aza.-3	14	2014-aza.-17	14

A.5.1 Proiektuaren Kronograma



Irudia 27 - Proiektuko kronograma

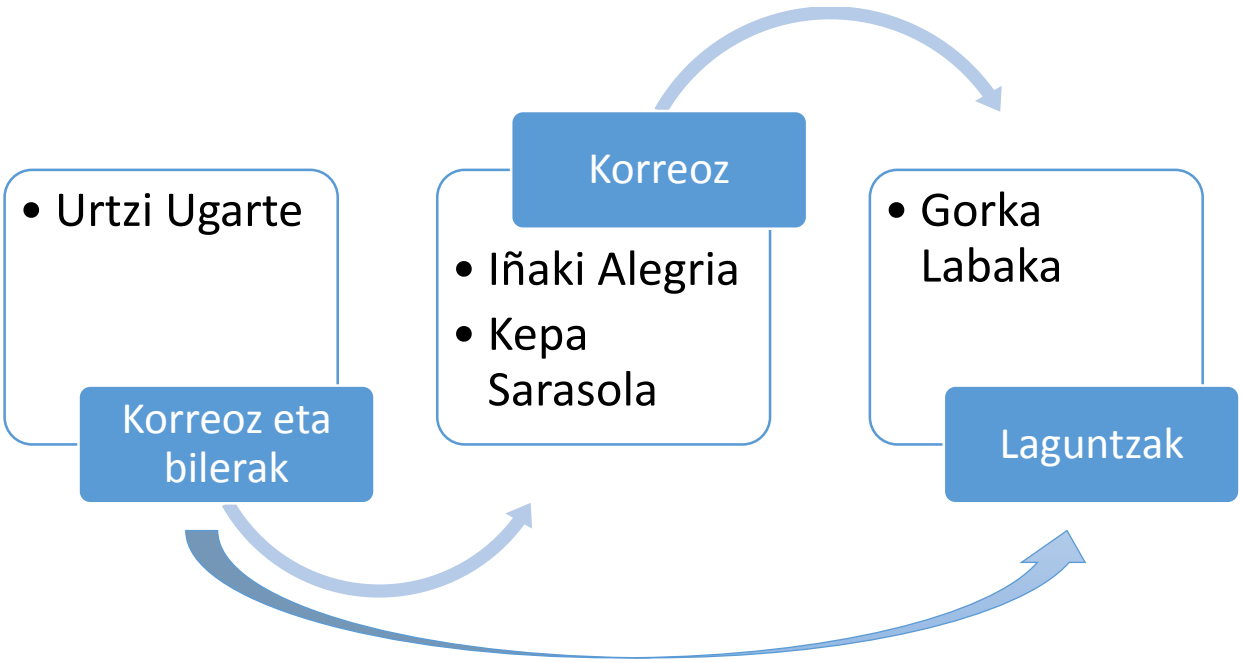
A.6 Komunikazio plana

Proiektua indibiduala izanik, barne komunikaziorik ez dago.

Kanpo komunikazioei dagokionez, kontaktu nagusiak bi izango dira: Iñaki Alegria (*i.alegria@ehu.es*) eta Kepa Sarasola (*kepa.sarasola@ehu.es*), hau da, proiektu zuzendariak. Hauekin erabiliko den komunikazio metodoak *Email*-a izango da.

Proiektuan aurrera pausu garrantzitsuren bat ematen denean bilera bat egingo da bi irakasleekin eta bertan proiektuaren jarraipena egiteaz gain, falta den eta hobetzea dagoen atalak azalduko dira.

Bestalde badira beste kontaktu batzuk non proiektuaren atal desberdinetan beharrezkoak izan diren hauen laguntza, baina proiektuko zuzendariak bitartekari bezala aritu dira. Kontaktu hauek honakoak dira: Gorka Labaka (*gorka.labaka@ehu.es*) non Matxin itzultzailearen web-zerbitzuari buruzko azalpenak eman zizkigun.



A.7 Kalitate plana

Proiektu honen kalitateari buruzko atala 3 zatitan banatu da: **Proiektua, Produktua eta Dokumentazioa.**

Atal bakoitzak bere kalitate mota desberdina du, eta neurtzeko era desberdinak. Proiektuaren kalitate totala bere zatien (eta bakoitzak duen azpi atal guztien) kalitate mailen batura izango da.

A.7.1 Proiektuaren kalitatea

Hemen proiektuaren kudeaketan jakin behar diren kalitate arauak agertzen dira. Arauak kalitate optimo bat izango litzake, baina onargarriak diren

- Eramangarriak
  - **KE1**- Emangarri guztiak bere identifikadorea izango dute, plangintzan azalduko dira bakoitzaren identifikazioa.
  - **KE2**- Emangarri bakoitza bere mugarria baino lehen bidali behar da.
  - Emangarrien kalitate ona izateko:
    - Emangarrien ehuneko %80 identifikadorea izango dute.
    - Emangarri guztiak bere mugarria betetzen denerako bukatua egon behar da.

Erroreen antzematea:

Arau hauek eramangarri bakoitzak bukatu baino lehen atzeman behar da, horrela arazoak konpontzeko denbora izateko.

A.7.2 Produktuaren kalitatea

- Diseinua
  - **KPD1**- Aplikazioan ortografia akatsik ezingo da egon.
  - **KPD2**- Adostutako koloreak izango dira aplikazioaren kolore nagusienak.
  - **KPD3**- Adostutako letra formatua erabili behar dira.
  - **KPD4**- Aplikazioa Windows Sistema eragilearekin, gutxienez, erabili behar da.

- Diseinuaren kalitatea ontzat emateko:
  - Ortografia akatsik ez (KPD1 betebehar bat da).
  - Adostutako koloreak %80an agertzea.
  - Adostutako letra formatua %80an agertzea.
  - Aplikazioa gutxienez Linux eta Windows sistema eragileetan funtzionala izan behar da.

Erroreen antzematea:

Mugarria heldu baino lehen arau guztiak hartu eta ikusi ondo dauden, bestela zuzendu. KDP4-ren kasuan hasieratik pentsatua egon behar da denbora eta lan asko izan daitekeelako azken orduko aldaketak.

- Ulerterraza
  - **KU1**- Erabilitako hizkuntza aplikazioan Euskara izango da.
  - **KU2**- Testuak ezin dira 20 lerrotik gora pasa.
  - **KU3**- Dokumentuen kalitate ataleko kalitate on bat izan behar du.
  - Ulerterrazaren kalitatea ontzat emateko:
    - Erabilitako hizkuntza Euskara (KU1 betebehar bat da).
    - Testuak 30 lerrotik gorakoa ez izatea.
    - Dokumentuen kalitate ona izan behar du

Erroreen antzematea:

Zuzendariekin irakurri eta ulertze arazoak badaude aldatu.

A.7.3 Dokumentuaren kalitatea

- **KD1**- Dokumentu bakoitzak bere identifikazioa izan behar du.
- **KD20**- Dokumentu guztiak azal bat izan behar dute.
- **KD21**- Dokumentuaren azalean Izen eta bi abizen agertu behar dira.
- **KD22**- Dokumentuaren azalean data agertu behar da.
- **KD3**- Dokumentuan ezin dira egon akats ortografikorik.
- **KD4**- Dokumentuan Euskara erabili behar da.
- **KD5**- Dokumentuaren orri bakoitzean orri oina agertu behar da.
- **KD6**- Dokumentuak "Justifikatuak" egon behar dira.

Erroreen antzematea:

Erroreak atzemateko arau bakoitza hartu eta dokumentu guztitik arazo hau gainbegiratu behar da.

A.8 Arriskuen plana

A.8.1 Identifikaturiko arriskuak

Proiektuaren planifikazioa betetzeko gai ez izatea denbora arazoengatik, irakasgaien lan karga edo proiektuaren tamaina dela medio. Hau gertatzen bada, kronograma egokitu beharko da eta horrekin atazak gainbegiratuko dira soluzio bat topatzeko.

Proiektuan erabiliko diren datu guztien edo zatiren baten galera.

A.8.2 Kontingentzia plana

Arazo hauek ekiditeko planifikazio eta barne mugarri malguak diseinatu dira, non irakasgaiek lan karga handia hartzen duten momentuan proiektuaren lan karga atzeratzeko aukera izango den. Barne mugarriak aldatzeko aukera emateak esan nahi du, kanpo mugarriekin kontuz ibili behar garela plangintza egitean, bertan arriskuek sortu ditzaketen arazoentzako denbora utziz. Denbora hau bi hilabeteraino iritsi daiteke.

Datuak ez galtzearen arazoa ekiditeko, astero proiektu osoaren segurtasun kopiak gordeko dira bai lainoan bai disko gogor batean. Hilabetean behin azken kopia hauek martxan jarriko ditut gordetako kopiak ondo dabiltzala jakiteko. Bestalde, kodearen bertsioak kudeatzeko aplikazio bat erabiliko dut laguntza bezala: **Git[29].**

[29] Git –distributed-is-the-new-centralized [Link]

A.9 Mugarriak

Kalitatezko proiektu bat lortzeko intentziotan barne mugarri batzuk aurreikusi ditugu, beti ere kanpo mugarriak garaiz eta arriskuen planean ikusi bezala denbora soberan utziz.

A.9.1 Barne Mugarriak

- 2014/01/27 – Oinarrizko zatia egin
  - ✓ Twitterren API-a.
  - ✓ Normalizatzaile arrunta.
  - ✓ Matxin Web-Zerbitzua.
  - ✓ Memoriaren plangintza zatia amaitua egon behar du.
- 2014/04/24 – Lehengo bertsioa + Demoa
  - ✓ Jadanik itzultzailearen lehen bertsioak amaitua egon behar du eta demo bat landua bileretan erakusteko.
- 2014/06/20 – Dokumentazioarekin hasteko eguna.
- 2014/07/05 – Bigarren bertsioa + 2. Demoa
  - ✓ Fitxategietatik itzultzeko aukera prest egon behar du
- 2014/10/05 – Hirugarren bertsioa. + Aplikazioaren azken Demoa
  - ✓ Sintagmaz banatu eta normalizatzaile hobeago batekin itzultzeko prest egon beharko luke.
- 2014/10/15 – Jendaurrean erakusteko egin beharreko prestakuntzak amaitu.
- 2014/10/20 – Proiektua bukatzeko azken eguna.

A.9.2 Kanpo Mugarriak

- 2014/11/03 – Proiektua ADDI-ra igo
  - ✓ Proiektuko ataza guztiak amaituak egon behar dute proiektuaren defentsa prestaketa izan ezin.
- 2014/11/17-19 – Proiektuaren defentsa
  - ✓ Egun honetarako proiektuaren defentsaren aurkezpenak amaitua egon behar du.

Bibliografia

[1] Twitter API-aren web-gunea. URL: https://dev.twitter.com/

[2] Matxin API-aren web-gunea. URL: http://matxin.sourceforge.net/

[3] Matxin itzultzailearen Ixako web-gunea. URL: http://ixa.si.ehu.es/Ixa/Produktuak/1273221240

[4] Jazzy SpellChecker web-gunea. URL: http://jazzy.sourceforge.net/

[5] Twitter4j web-gunea. URL: http://twitter4j.org/en/index.html

[6] TxioItzultzailearen Twitter Apps web-gunea. URL: https://apps.twitter.com/app/5372007/show

[7] (2009) Philipp **Koehn**: *Statistical machine translation*. Cambridge University Press. xii, 433 horri. [ISBN: 978-0-521-87415-1]

[8] (2006) C.K.**Quah**: *Translation and technology*. Basingstoke (UK): Palgrave Macmillan. xix, 221 horri. [ISBN: 978-1-4039-1831-4 (hardback), 978-1-4-39-1832-1 (paperback)]

[9] (2009) Kevin **Makice**: *Twitter API: Up and Running: Learn how to build applications with the Twitter API*. i, 416 horri. [ISBN: 978-0596154615]

[10] (2002) **Chan** Sin-wai (ed.) *Translation and information technology*. Hong Kong: Chinese University Press. xii,215 horri.. [ISBN: 962-996-077-x]

[11] (2003) **Reifer**, D.J., **Maurer**, F., **Erdogmus**, H: *Scaling agile methods*, IEEE Software.

[12] (2008) **IBM**: *Jibes creates dynamic demand planning for semiconductor industry using enterprise mashup from IBM*. IBM Corporation.

[13] (1995) **Gamma**, E., R. **Helm**, R. **Johnson**, J. **Vlissades**. *Design patterns elements ofreusable object-oriented software*. Addison-Wesley professional computing series.

[14] (2009) W. **Ketter**, M. **Banjamin**, R.**Guikers**, Alfred **Kayser**: *Introducing an Agile Method for Enterprise Mash-Up component Development*. URL: http://www-users.cs.umn.edu/~ketter/research/publications/Ketter09CEC.pdf

[15] (2008) J. Wong, J. Hong: What do we mashup when we make mashups?. URL: http://mashup.pubs.dbs.uni-leipzig.de/files/Wong2008Whatdowemashupwhenwemakemashups.pdf

[16] (2001) Iñaki **Alegria**, Xabier **Artola** eta Kepa **Sarasola**. *Hizkuntzaren tratamendu automatikoa: aplikazioak, tresnak, baliabideak eta oinarriak.* URL: http://www.euskonews.com/0110zbk/gaia11004eu.html#1

[17] Itzulpen automatikoaren magia eta mugak. URL: http://zientzia.net/artikuluak/itzulpen-automatikoaren-magia-eta-mugak/

[18] (2002) K. **Schwaber**, M. **Beedle**: *Agile Software Development with Scrum*. Pearson International edition. [ISBN: 978-0130676344]

[19] (2009) Mike **Cohn**: *Succeding with Agile: Software development using Scrum*. Pearson Education. [ISBN: 9780321660565]

[20] (2007) J **Sutherland**, A **Viktorov**, J **Blount**: *Distributed Scrum: Agile Project Management with Outsourced Development Teams*. URL: http://jeffsutherland.com/SutherlandDistributedScrumHICCS2007.pdf

[21] (2010) Max **Kaufmann**: *Syntactic Normalization of Twitter Messages*. URL: http://cs.uccs.edu/~jkalita/work/reu/REUFinalPapers2010/Kaufmann.pdf

[22] (2010) *The Edinburgh Twitter Corpus.* Computational Linguistics in a World of Social Media

[23] (2012) U. **Cabezón**. *Itzulpen automatikorako tresnen egokitzapena euskararako: post-edizioa, ebaluazioa eta aurre-edizioa*. URL: *https://addi.ehu.es/bitstream/10810/9041/4/Unai%20Cabez%C3%B3n%20-%20Karrera%20Bukaerako%20Proiektua.pdf*

[24] (2011) **Alegria** I., **Cabezón** U., **Labaka** G., **Mayor** A., **Sarasola** K.: *Matxin-Informatika: versión del traductor Matxin adaptada al dominio de la informática.* URL: http://journal.sepln.org/sepln/ojs/ojs/index.php/pln/article/view/992/745

[25] (2009) K. **Sarasola** : *Matxin: developing sustainable machine translation for a less-resourced language.* URL: http://ixa.si.ehu.es/Ixa/Argitalpenak/Artikuluak/1257264733/publikoak/FreeRBMT_Alacant09.pdf

[26] (2006) **Crystal**, D: *Language and the Internet*. Cambridge University Press, 2 edition.

[27] (2010) University of Edinburgh : *Machine Translation for Twitter*. Examination Number: 6898847. URL: http://homepages.inf.ed.ac.uk/miles/msc-projects/jehl.pdf