Universitat Rovira i Virgili (URV) i Universitat Oberta de Catalunya (UOC)

Màster en Enginyeria Computacional i Matemàtica

Treball de Fi de Màster

Ús d'algorismes genètics per desxifrar algorismes de classificació opacs.

Nom Estudiant: Albert Llabrés Darder

Nom Director/a: Dr. Hebert Pérez-Rosés

Data d'entrega: 21 de juny de 2020

El Dr. Hebert Pérez-Rosés, certifica que l'estudiant Albert Llabrés Darder ha elaborat el treball sota la seva direcció i autoritza la presentació d'aquesta memòria per la seva avaluació.

Firma del director/a:

FITXA DEL TREBALL DE FI DE MASTER

Títol del treball:	*Ús d'algorismes genètics per desxifrar algorismes de classificació opacs.*
Nom de l'autor:	Albert Llabrés Darder
Nom del director/a:	Dr. Hebert Pérez-Rosés
Data:	06/2020
Titulació:	Màster en Enginyeria Computacional i Matemàtica
Àrea del Treball:	*Matemàtica Aplicada.*
Idioma del treball:	*Català*
Paraules clau:	Matemàtica Aplicada; Algorismes genètics; Regressió simbòlica.

Resum del Treball:

En els últims anys, els mitjans de comunicació han esdevingut un element clau per tal de transmetre informació d'una forma determinada amb l'objectiu de crear una opinió concreta en el lector.

Tradicionalment, aquest objectiu s'assolia a través de la pròpia redacció de la informació. No obstant, amb l'aparició de productes interactius on tots els usuaris poden intercanviar informació, ha esdevingut clau la forma en la que aquests usuaris transmeten aquesta informació així com el posicionament que obtenen.

Durant aquest treball de fi de màster, s'ha realitzat un seguiment de les notícies publicades a Yahoo! News per tal d'arribar a una aproximació en els seus algorismes de classificació de comentaris.

L'objectiu d'aquest treball és poder comprendre quins paràmetres són més importants a l'hora d'establir l'ordre en què els usuaris finals rebran els comentaris publicats, de tal forma que les entitats interessades en aquest aspecte pugin redactar comentaris de forma eficient amb l'objectiu de transmetre una informació determinada.

Per tal d'assolir aquest objectiu, en aquest treball s'ha fet ús de metodologies estudiades en aquest màster, tals com els algorismes d'optimització meta-heurística, la programació genètica, les regressions simbòliques, etc.

Els resultats d'aquest treball permetran aconseguir una aproximació als algorismes de classificació emprats per Yahoo! News, de tal forma que podrem deduir els criteris seguits amb un determinat error relatiu.

Les conclusions d'aquest treball permetran comprovar que Yahoo! News fa ús de paràmetres de classificació que no es troben a l'abast dels seus usuaris, i que per tant disposa d'una gran opacitat.

Abstract:

In recent years, the media has become a key element in conveying information in a certain way with the aim of creating a concrete opinion in the reader.

Traditionally, this goal has been achieved through the writing of the information itself. However, with the advent of interactive products where all users can exchange information, it has become key how these users transmit this information as well as the positioning they obtain.

During this master's thesis, the news published in Yahoo! News has been tracked in order to arrive at an approximation in their comment classification algorithms.

The aim of this work is to be able to understand which parameters are more important when establishing the order in which end users will receive the published comments, so that the entities interested in this aspect can write comments in a way efficient with the aim of transmitting certain information.

In order to achieve this goal, in this work we have used methodologies studied in this master's degree, such as meta-heuristic optimization algorithms, genetic programming, symbolic regressions, etc.

The results of this work will provide an approximation to the ranking algorithms used by Yahoo! News, so that we can deduce the criteria followed with a certain relative error.

The conclusions of this work will allow us to verify that Yahoo! News makes use of classification parameters that are not available to its users, and therefore has a high opacity.

Índex

Llista de figures

1. Introducció

1.1 Context i justificació del Treball.

Avui en dia, la difusió d'informació es pot fer a través de nombrosos mitjans de distribució, per una banda existeixen els clàssics mitjans de comunicació tals com la televisió, la radio, i la premsa escrita, mentre que en les darreres dècades s'ha fet extensiu l'ús d'internet com a principal mitjà de comunicació. Dintre d'internet podem trobar nombroses eines per tal de poder fer difusió d'aquesta informació. Entre altres tenim els diaris digitals, les xarxes socials, els blogs, etc.

Les diferents eines de què disposem a internet, han fet possible que la població pugui interactuar entre si a l'hora de obtenir o distribuir informació. Per una banda, això ha possibilitat que els usuaris tinguin nombroses fonts d'informació de les quals beneficiar-se, poder comparar la mateixa informació a través de diferents canals, així com poder-la contrastar a través de les pròpies fonts. No obstant, a l'altra cara de la moneda hi trobem informació poc fiable, sense citar-ne les fonts, o directament informació no veraç com és el cas de les anomenades "fake news" o notícies falses.

Molts països han legislat sobre internet al llarg dels anys per tal de poder fer-ne un lloc segur i fiable, no obstant el fet que tots els usuaris en puguin fer ús per consultar i difondre informació fa que sigui difícil de controlar per part dels diferents governs. Una de les lleis més importants sobre la que es sustenta internet és la Communications Decency Act (CDA), on podem trobar la secció 230 que també s'anomena popularment "les 26 paraules que van crear internet" tal i com el coneixem a dia d'avui.

Aquesta secció cita textualment:

Cap proveïdor o usuari d'un servei informàtic interactiu no serà considerat com l'editor o altaveu de cap informació proporcionada per un altre proveïdor de contingut d'informació.

Aquesta secció ha provocat que internet es trobi ple d'informacions, comentaris o opinions de dubtosa credibilitat i fins i tot falsos.

Aquest fet ha generat una finestra d'oportunitat per totes aquelles persones o entitats que volen difondre informació, ja sigui o no veraç, amb l'objectiu de crear una determinada opinió sobre el lector.

Tradicionalment, la forma de crear opinió a través de la difusió d'una notícia per part dels diferents diaris digitals que podem trobar, es basava en el redactat de la pròpia notícia, així com en el seu titular. En algunes ocasions els mitjans fan ús d'allò que s'anomena sensacionalisme, que és la forma en que es presenta una determinada informació per tal d'apel·lar

a determinats sentiments del lector, posant èmfasi en certs aspectes de la informació publicada.

No obstant, durant els darrers anys, amb el sorgiment de la interactivitat dintre dels propis diaris digitals, els usuaris han pogut fer ús de la secció de comentaris, amb l'objectiu de poder expressar la seva opinió sobre una informació determinada. Aquest fet ha provocat que juntament amb la informació que es presenta en una notícia, també es pot obtenir una gran quantitat d'informació a través dels comentaris existents a la mateixa. És així com s'ha creat una nova forma de generar opinió, molt més subtil que la pròpia redacció de la notícia i el seu possible sensacionalisme.

Són moltes les persones i entitats que fan ús d'aquestes eines interactives, on tots els usuaris poden publicar els seus comentaris, per tal de poder fer difusió d'una informació determinada amb l'objectiu que arribi a un determinat tipus de lector, de tal forma que l'objectiu final d'aquestes persones o entitats no és altre que crear una opinió determinada en els lectors que vagi més enllà del redactat de la notícia.

Aquest treball no pretén aprofundir en la veracitat de la informació ni en els objectius que pugui tenir la persona o entitat encarregada de la seva difusió, com tampoc pretén obrir un debat sobre la moralitat de crear una determinada opinió sobre un determinat tema, sinó que es centrarà exclusivament en les tecnologies existents i en la forma en la que es pot presentar una informació per tal que pugui arribar a la major quantitat possible de lectors creant així una opinió determinada.

1.2 Objectius del Treball.

Aquest treball es centrarà exclusivament en Yahoo! News. Yahoo! News és un agregador de notícies que s'encarrega de distribuir notícies d'altres fonts d'informació, generalment diaris digitals, per tal de facilitar-ne la seva lectura i la seva distribució.

El fet de ser un agregador de notícies, provoca que la redacció de la notícia així com el seu titular no depengui de Yahoo! News, sinó que són les pròpies fonts de la informació les encarregades de la redacció i el titular de les notícies que es presenten.

Tot i que Yahoo! News només és un agregador de notícies, la facilitat d'ús que presenta i el gran contingut d'informació que es pot trobar provinent de moltes i diverses fonts d'informació provoca que sigui un dels canals de difusió d'informació més importants a nivell mundial, i per tant, és d'un gran interès a l'hora de difondre informació i crear opinió per part de les entitats o persones interessades en aquest sentit.

Com ja s'ha comentat amb anterioritat, Yahoo! News no s'encarrega ni dels titulars ni de les redaccions de les notícies que es poden trobar a la pròpia aplicació. No obstant, una de les eines més interessants que es poden trobar dintre de Yahoo! News, és la secció de comentaris. La secció

de comentaris de Yahoo! News és l'eina que permet als usuaris lectors poder interactuar entre si i poder escriure la seva opinió o poder compartir informació sobre un determinat tema.

Les notícies que podem trobar a Yahoo! News poden arribar a tenir una gran quantitat de comentaris, de l'ordre de desenes de milers, i en alguna ocasió fins i tot de centenars de milers. Aquest fet deixa entreveure l'enorme repercussió que arriba a tenir Yahoo! News i la importància que poden tenir els comentaris d'una notícia per tal de crear opinió entre els lectors.

Els objectius específics d'aquest treball es poden desglossar de la següent forma:

- Extreure la quantitat d'informació suficient de les noticies de Yahoo! News així com dels seus comentaris per tal de poder obtenir resultats concloents.

- Desenvolupar en codi obert, una eina capaç de realitzar una aproximació a les tècniques i mètodes de classificació de Yahoo! News mitjançant l'ús d'algorismes meta-heurístics, programació genètica i regressions simbòliques.

- Assolir un marge d'error en la nostra funció objectiu inferior al 20%.

- Demostrar l'opacitat existent en alguns mitjans de comunicació a l'hora de classificar i ordenar les interaccions dels seus usuaris

1.3 Enfocament i mètode seguit.

Durant aquest treball, serà necessari obtenir una gran quantitat d'informació, ja que per poder aproximar-nos als algorismes de classificació de comentaris emprats per Yahoo! News és necessari que disposem d'una gran quantitat d'informació tant de les notícies com dels propis comentaris.

Per tal d'obtenir tota la informació que necessitem, farem ús de diferents tecnologies, tals com RSS, APIs, Web scrappers, etc.

Yahoo! news disposa de RSS. Aquesta eina ens permet obtenir informació sobre les notícies que podem trobar a Yahoo! News amb tot un seguit de paràmetres, tals com el seu identificador, el seu enllaç, el títol, la descripció, la data de publicació, etc. Gràcies a aquesta tecnologia ens resultarà molt fàcil poder fer un seguiment de les notícies.

No obstant això, hi ha molts altres paràmetres que no podrem trobar dintre de RSS, tals com el context de la notícia, el nombre de comentaris, o el contingut dels mateixos. El context de la notícia és una cadena alfanumèrica que serveix per identificar la notícia amb l'objectiu de poder

obtenir diferents continguts a través de la pròpia API de Yahoo! News, tals com els comentaris d'una notícia.

En aquest treball ens interessa obtenir informació sobre les notícies que disposen d'una gran quantitat de comentaris, així com dels comentaris en si, i per tant es necessitarà obtenir certa informació que és de més difícil accés per part dels algorismes emprats, ja que no es disposa d'una API ni de RSS per poder accedir a certs continguts.

És aquí on entra en joc el web scrapping, una tècnica que consisteix en emular les interaccions d'un usuari a través d'un explorador d'internet, emprant eines pròpies de programació i llibreries externes es desenvoluparà un web scrapper, que servirà tant per obtenir informació dels resultats de la API i de RSS com per obtenir aquella informació més inaccessible que només es pot aconseguir a nivell visual a través de la interfície d'usuari.

Un cop haguem obtingut tota la informació que necessitem, haurem de desenvolupar algorismes meta-heurístics per tal de crear una regressió simbòlica a través dels diferents atributs de què disposem, amb l'objectiu de poder aproximar de la forma més òptima possible quin és l'algorisme de classificació emprat per Yahoo! News.

Quan s'hagin obtingut els primers resultats, serà necessari fer un anàlisi dels mateixos, veure quin ha sigut l'error absolut i l'error relatiu de l'execució del nostre algorisme meta-heurístic, i fer-ne les modificacions pertinents per tal de millorar els nostres resultats.

1.4 Planificació del Treball.

Aquest treball implica el desenvolupament de diverses eines, en primer lloc necessitarem un conjunt d'eines per tal d'obtenir la informació que volem. En segon lloc, serà necessària una eina per tal d'emmagatzemar aquesta informació. En tercer lloc, necessitarem una eina capaç de treballar amb tot aquest conjunt d'informació per tal d'arribar a resultats d'una regressió lineal. I per acabar, necessitarem eines per interpretar i avaluar aquests resultats amb l'objectiu de fer les modificacions necessàries de cara a una millora.

Durant aquest treball serà necessària una planificació en diferents fases que alhora es trobaran dividides en diferents tasques, en les quals es desenvoluparan les diferents eines així com s'interpretaran els resultats de cara a realitzar les modificacions oportunes.

A continuació es proposen les diferents fases en que es dividirà el projecte, així com els recursos necessaris en cadascuna d'elles.

- **Fase de planificació:** durant aquesta fase es durà a terme un primer contacte amb Yahoo! News, es valoraran les eines de que disposa (API, RSS, WS, UI, etc.), i es prendran decisions sobre els mètodes a

seguir, els objectius als quals es vol arribar, els nivells d'abstracció inicial de la informació que volem obtenir, i la forma en la que volem obtenir aquesta informació.

- **Fase d'anàlisi:** durant aquesta fase es decidiran les metodologies de desenvolupament a seguir per tal de poder assolir els nostres objectius. Aquestes metodologies es decidiran en funció de les eines de les que disposem així com també de les tecnologies que podem trobar a Yahoo! News i que ens marcaran de forma clara que és allò que podem fer i que és allò que no.

- **Fase de disseny**: durant aquesta fase és necessari que es defineixi com el nostre producte assolirà els objectius plantejats, i per tant es durà a terme una selecció de recursos, es decidirà quines tecnologies són les més apropiades tant a nivell de programari com a nivell de maquinari.

- **Fase de desenvolupament**: es tracta de la fase més important del projecte i la que implicarà més temps. Durant aquesta fase es dura a terme el codi de les diferents eines necessàries, que entre d'altres seran scrappers de notícies, scrappers de comentaris, definició de les diferents entitats, creació de la base de dades, programació genètica, codificació d'analitzadors dels resultats obtinguts, etc.

- **Fase de proves**: un cop finalitzada la fase de desenvolupament, serà necessari realitzar les primeres proves, en les quals podrem obtenir els resultats de la nostra eina i podrem valorar-ne la idoneïtat. Es valorarà el marge d'error obtingut, així com el nivell d'abstracció dels paràmetres seleccionats com a part de les proves.

- **Fase de millora**: Com que es tracta d'un projecte d'optimització meta-heurística, seran constants les millores que haurem de realitzar al projecte amb l'objectiu d'obtenir resultats cada cop més òptims. En aquesta fase es decidiran les millores a realitzar i es tornarà novament a la fase de desenvolupament per tal que siguin implementades per fer les seves posteriors proves i anàlisi de resultats.

1.5 Breu sumari dels productes obtinguts.

En aquest treball es desenvoluparan les eines necessàries per crear un seguiment de comentaris de Yahoo! News, aquests productes seran els següents:

- **Aplicació de consola de Windows programada en c#:** Aquesta aplicació s'encarregarà tant de l'obtenció de les dades com del seu posterior tractament mitjançant programació genètica. Així mateix s'encarregarà d'exportar els resultats tant a la base de dades com a fitxers CSV pel seu tractament en matlab.

- **Base de dades en MySql:** Aquesta base de dades s'encarregarà d'emmagatzemar tota la informació. Aquí és on es desaran les dades de: Notícies, comentaris, resultats d'execució i errors sobrevinguts de l'aplicació.

- **Fitxers CSV:** Un cop executat de forma satisfactòria l'eina de web scrapper sobre una notícia, ens pot resultar d'un gran interès poder exportar aquesta informació en una forma que ens faciliti el seu tractament en matlab per la seva posterior interpretació de cara a futures millores. Aquest format seran els fitxers CSV.

- **Script Matlab:** Aquest script s'encarregarà de mostrar de forma més intuïtiva tots els resultats del seguiment d'una notícia, de tal forma que podrem analitzar millor els resultats i implementar millores en el nostre algorisme genètic.

1.6 Breu descripció dels altres capítols de la memòria.

Part I: Marc Teòric.

- **Capítol 2.** *Algorismes meta-heurístics*: s'explica el marc teòric sobre el que es basarà aquest treball.

Part II: Fase de planificació.

- **Capítol 3.** *Yahoo! News: Un primer contacte*: s'explica el funcionament de Yahoo! News, com funcionen les notícies i els comentaris, es planifica quina serà la forma més adient d'obtenir informació sobre les notícies i els seus comentaris.

Part III: Fase de disseny

- **Capítol 4.** *Preparació de l'entorn de desenvolupament:* es durà a terme la selecció de tecnologies més apropiades que s'empraran pel desenvolupament de l'aplicació així com les eines necessàries per treballar amb aquestes tecnologies.

Part IV: Fase de desenvolupament

- **Capítol 5.** *Obtenció de dades:* es desenvoluparan les eines necessàries per tal d'obtenir les dades suficients dels comentaris i les notícies per ser tractats amb posterioritat per l'algorisme genètic.

- **Capítol 6.** *Creació de gens i cromosomes:* es desenvoluparan totes les estructures de dades necessàries per tal de representar la informació i poder ser tractades de forma eficient per part de l'algorisme genètic.

- **Capítol 7.** *Implementació de l'algorisme genètic:* es desenvoluparà l'algorisme genètic i es prendran les decisions adients en quan a selecció de gens, exploració, mutació, etc. Per tal de millorar els resultats de la funció objectiu.

Part V: Fase de proves

- **Capítol 8.** *Execució i anàlisi dels primers resultats:* s'executarà l'algorisme genètic emprant les dades obtingudes i es valoraran els resultats de la funció objectiu i el seu marge d'error.

- **Capítol 10:** *Execució i anàlisi final dels resultats:* s'executarà la versió de l'algorisme que conté les millores dutes a terme i es valoraran els resultats obtinguts de forma crítica tenint en compte els objectius.

Part VI: Fase de millora

- **Capítol 9:** *Presa de decisions i implementació de millores:* es prendran les decisions necessàries per tal de millorar els resultats obtinguts durant l'execució de la primera versió del nostre algorisme genètic.

Part VII: Conclusions

- **Capítol 11.** *Conclusions:* es valorarà la consecució dels objectius del treball, el mètode seguit, la planificació, i les línies de treball futures.

2. Algorismes meta-heurístics

Al llarg de la història, a l'hora de trobar la solució a certs problemes computacionals, es recorre als algorismes heurístics, que s'encarreguen de trobar la solució al problema amb un cos computacional determinat. No obstant, en algunes ocasions no existeix un mètode heurístic per trobar una solució a un problema computacional determinat, o aquets mètodes són computacionalment massa costosos. Podem trobar exemples en problemes de combinatòria tals com el problema d'enrutament de vehicles (VRP), en el qual en moltes ocasions es tracta de trobar una solució el més optima possible tot i que no sigui la millor de totes elles.

Per donar solució a aquest tipus de problemes, ha sorgit l'optimització meta-heurística, així com els seus diferents tipus d'algorismes que en podem trobar. La meta-heurística no és un camp que pugui trobar una solució determinada a un problema tal com si ho fan els algorismes heurístics, sinó que prova de trobar una solució òptima, dintre d'un temps determinat, i amb un cost computacional concret, és per això que tracten d'anar més enllà del que poden arribar les heurístiques, d'aquí el prefix grec meta (més enllà).

Els algorismes heurístics acostumen a ser més eficients en els problemes que accepten aquest tipus d'algorismes, no obstant, quan no parlem de problemes purament heurístics, en moltes ocasions els algorismes heurístics no poden trobar una solució òptima, o el cost de trobar-la és molt més elevat que no pas el cost que suposa per un algorisme meta-heurístic.

D'algorismes meta-heurístics en podem trobar de molts de tipus, entre els quals en destacarem els algorismes de selecció genètica, que són els que implementarem al llarg d'aquest treball.

2.1. Algorismes genètics.

Dintre dels algorismes meta-heurístics, podem trobar els anomenats algorismes evolutius, i dintre d'aquests últims en podem destacar els algorismes genètics.

Els algorismes genètics són un tipus de meta-heurística que es basa en la teoria evolutiva que podem trobar dintre de la naturalesa dels éssers vius. Aquests algorismes parteixen d'una població, la qual podran anar modificant a través de la creació de nous individus, la mutació dels ja existents o el creuament entre diversos individus.

Aquests algorismes són realment útils quan ens enfrontem a un problema que és massa complex per plasmar-lo en llenguatge de programació, o bé quan la forma de trobar una solució no és prou clara, o bé quan no és possible explorar tot l'espai de solucions.

Cal deixar clar que aquest tipus d'algorismes no acostumen a arribar a una solució òptima, i que el seu marge d'error sempre serà considerable, aquesta és la diferència entre aquest tipus d'algorismes i els algorismes heurístics tradicionals, ja que aquests últims s'empren per poder trobar una solució determinada sense marge d'error o amb un marge d'error molt petit.

Un dels problemes on aquests algorismes són recomanables és en una regressió simbòlica d'una quantitat considerable de variables, doncs la funció a la que volem arribar no és pas coneguda com tampoc ho és la forma de trobar la solució.

El funcionament d'aquest tipus d'algorismes es podria definir de la següent forma:

Inicialització: Es genera una població inicial de cromosomes amb valors aleatoris dintre d'uns marges definits.

Avaluació: S'avalua la idoneïtat de cada un dels cromosomes creats amb el resultat final que desitgem obtenir i se'n estableix una variable de fitness.

Selecció: Es seleccionaran els millors cromosomes a través dels majors fitness obtinguts.

Creuament / mutació: Es definirà quin és el mètode a seguir per tal de crear nous cromosomes a partir dels existents. En el cas del creuament s'entrecreuaran valors de varis cromosomes seleccionats, mentre que en el cas de la mutació es canviaran un determinat nombre de valors d'un sol cromosoma per tal de crear-ne un de nou.

Substitució: Un cop acabat el creuament o la mutació, es compararan els cromosomes originals amb els cromosomes generats amb l'objectiu de substituir-los en cas que els que s'han generat tinguin un millor resultat en la seva avaluació.

Terminació: Un cop realitzada la substitució, es retornarà al pas de selecció i es seguiran executant tots els passos de forma iterativa fins que es compleixi una condició de terminació. Aquesta condició pot venir donada per un resultat determinat en la fase d'avaluació, per un temps determinat en l'execució, o per un nombre d'iteracions determinat.

A continuació es mostra el funcionament d'un algorisme genètic simple mitjançant pseudo-codi, així com a l'annex 15.1 d'aquest document es pot veure un diagrama de flux del funcionament d'un algorisme genètic simple.

```
1. BEGIN /* Algorisme Genètic Simple */
2.  Generació una població inicial.
3.  Computació d'avaluació (fitness).
4.  WHILE NOT Acabat DO
5.  BEGIN /*Cicle repetitiu */
6.        Selecció de cromosomes.
7.        Clonació.
8.        Mutació de l'algorisme clonat.
9.        Computació d'avaluació del cromosoma clonat.
10.       END
11.       IF condició de fi THEN
12.             Acabat := TRUE
13.       END
14. END
```

2.2. Regressió simbòlica.

Una de les aplicacions pràctiques dels algorismes genètics són les regressions simbòliques.

Les regressions simbòliques són un mètode que busca dintre de l'espai d'expressions matemàtiques amb l'objectiu de trobar, mitjançant un conjunt de valors donats i el seu valor resultant, aquella expressió matemàtica que més s'ajusta a una funció desconeguda i que fa que es compleixin els resultats.

La generació de regressions simbòliques mitjançant algorismes genètics, fa possible que en molts casos es pugui arribar a una funció matemàtica que s'aproximi amb un cert marge d'error a aquella funció que ha donat origen al conjunt de dades de que disposem.

El fet que es tracti d'una meta-heurística provoca que en molts casos, sobretot en aquells en que no es disposa d'un conjunt suficientment gran de valors, els marges d'error poden ser considerablement elevats. No obstant, tot i els marges d'error resultants poden ser de gran utilitat a l'hora de comprendre com és el funcionament d'un algorisme.

En aquest treball, es farà ús d'algorismes genètics per tal de poder crear una regressió simbòlica sobre el major nombre de paràmetres possible que es puguin aconseguir a través de l'exploració i obtenció de dades. L'objectiu últim d'aquesta aplicació no és altre que poder trobar una funció que s'ajusti el màxim possible al la funció de classificació emprada per Yahoo! News a l'hora de ponderar i classificar els comentaris existents en una notícia.

3. Yahoo! News: Un primer contacte.

Durant aquest apartat, ens centrarem en prendre contacte amb Yahoo! News, així com amb les diferents notícies que podem trobar, les categories existents, i els mètodes d'obtenció d'informació per tal de poder implementar una extracció de dades efectiva.

3.1. Les notícies publicades.

El fet de ser un agregador de notícies provoca que la gran majoria de notícies presentaran el mateix format, fet que ens facilitarà enormement l'extracció de dades.

El format de les notícies a Yahoo News es pot observar a continuació.

Figura 1. Imatge d'una notícia de Yahoo! News

https://news.yahoo.com/ohio-senator-sherrod-brown-blasts-trump-but-praises-republican-governors-response-to-coronavirus-185910370.html

Podem veure que la notícia disposa del seu títol, la font, la data, el contingut, i una icona per visualitzar els comentaris juntament amb el nombre de comentaris de la notícia.

3.2. Els comentaris d'una notícia.

Si cliquem sobre la icona de comentaris de la notícia, apareixeran al peu de la notícia amb el següent format:

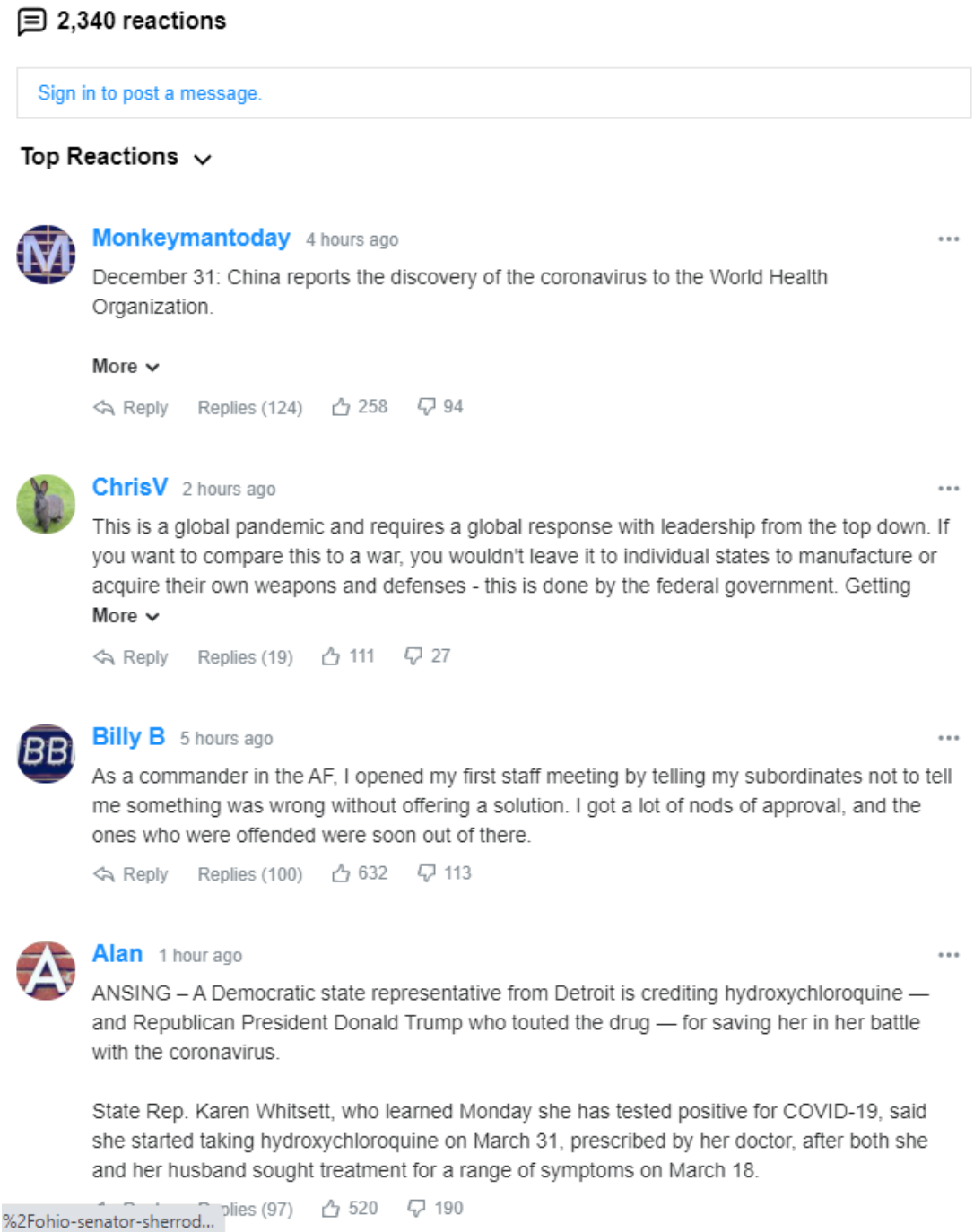

Figura 2. Imatge del llistat de comentaris en una notícia.

Tal com es pot observar, en aquest apartat podem veure el contingut de cada comentari, així com la seva antiguetat, el seu autor, el nombre de respostes, així com el nombre de "m'agrada" i "no m'agrada" que ha obtingut.

Com podem observar, tant en la propia notícia com a la secció de continguts podem obtenir una gran quantitat d'informació que ens serà d'un gran ús.

Arribats a aquest punt, és important fer menció a la classificacio que podem observar, ja que com es pot veure, per defecte es troba en una classificació de TOP reactions que no sembla seguir cap ordre lògic. L'ordre de classificació de tipus TOP, no es fa en funció de la data de publicació d'un comentari, així com tampoc en funció del nombre de respostes ni el nombre de likes o dislikes que tinguin els diferents comentaris.

És en aquest ordre de classificació TOP en el qual centrarem aquest treball, doncs es tracta de saber quina funció matemàtica està seguint yahoo per decidir quins comentaris són més TOP que d'altres. Tal com s'ha explicat a la introducció d'aquest treball, un dels fenòmens que provoquen aquest tipus de comentaris, és la capacitat de crear opinió entre els seus lectors, i per tant pot resultar de vital importància per certes persones o entitats interessades en la difusió d'informació i la creació d'opinió el fet de conèixer una aproximació a aquests mètodes de classificació, ja que pot resultar de gran utilitat de cara a obtenir la major quantitat de lectors possibles amb l'objectiu de crear una determinada opinió.

3.3. RSS.

RSS és un estàndard en la redifusió de notícies de forma eficient entre els diferents serveis que es poden trobar a internet. RSS és un tipus de document XML que conté informació que s'actualitza de forma constant. De tal forma que aquelles persones o entitats subscrites a un determinat RSS podran rebre a temps real la informació que es distribueixi a través d'aquest canal.

Yahoo! News disposa de la seva pròpia eina de RSS a la qual totes les persones o entitats que ho desitgin s'hi poden subscriure de forma gratuïta. En podem veure el seu funcionament al següent enllaç.
https://news.yahoo.com/rss/

Tot seguit s'adjunta una imatge d'allò que podem trobar a un RSS de Yahoo! News.

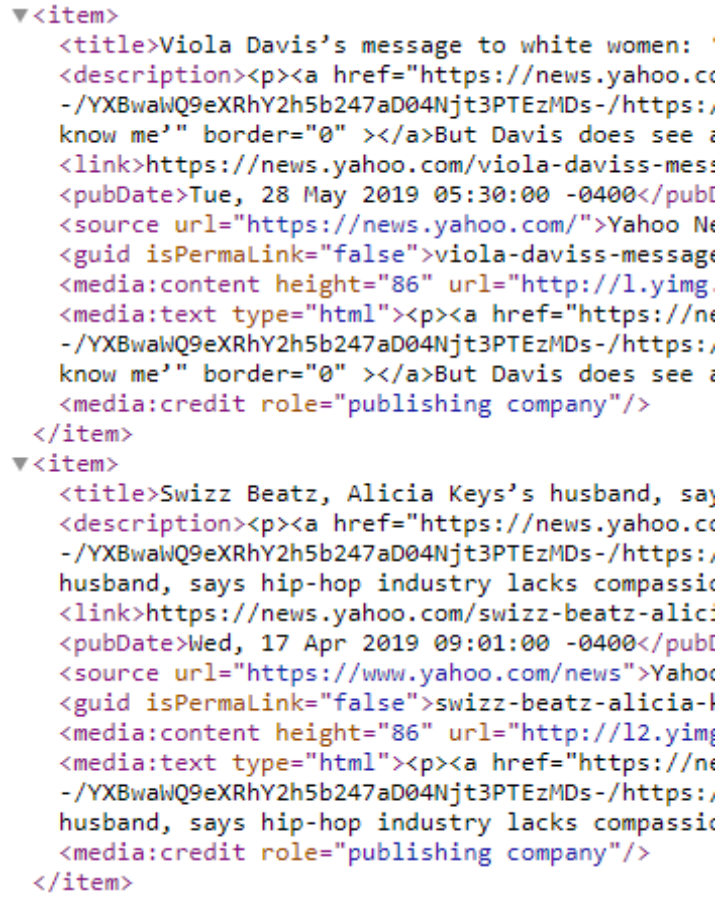

Figura 3. Imatge del contingut RSS retornat per Yahoo! News.

Tal com es pot observar, a través del resultat RSS de Yahoo! News, podem extreure'n dades tals com el títol de la notícia, la seva descripció, el seu enllaç públic, la seva data de publicació, la font de la informació així com el seu identificador únic emprat per Yahoo! News anomenat guid. Aquest guid ens serà útil de cara a guardar la informació a la nostra pròpia base de dades com a clau primària.

Aquesta eina ens resultarà molt útil, ja que ens facilitarà l'obtenció d'una gran quantitat d'informació que d'una altra forma hauríem d'haver implementat al nostre web scrapper.

3.4. API de comentaris.

Durant l'etapa d'observació d'aquest treball, s'ha trobat l'eina pròpia que empra Yahoo! News per tal de poder accedir als seus comentaris i presentarlos a l'usuari final d'una forma eficient. Es tracta de la seva pròpia API de comentaris.

Aquesta API la podem veure en funcionament des de l'inspector d'elements de Google Chrome quan cliquem sobre la icona de comentaris.

Figura 4. Imatge de l'inspector d'elements de chrome, amb la cridada a la API remarcada.

Tal com es pot observar a la imatge, un cop clicat sobre la icona de comentaris, la pàgina web fa una cridada XHR a aquest fitxer anomentat canvass.getMessageListForContext. Aquesta és la cridada que retorna els comentaris a la web per poder ser presentats d'una forma eficient.

Si veiem els resultats que retorna aquesta cridada, podem veure les següents dades:

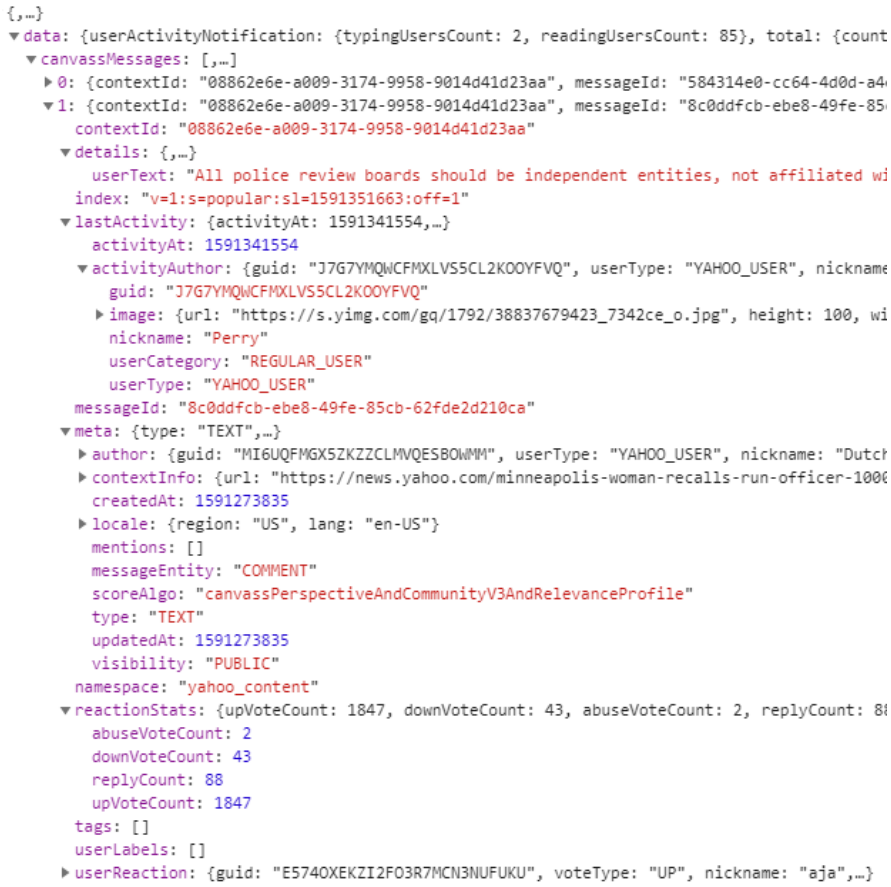

Figura 5. Imatge de resultats retornats per l'API de Yahoo! News.

Tal i com es pot observar, es tracta d'una resposta JSON on podem veure tots els resultats dels comentaris, així com tot un seguit de paràmetres per cadascun dels comentaris que ens resultaran molt útils de cara a veure quina funció matemàtica empra yahoo a l'hora de classificar els seus comentaris.

Més enllà de les dades dels propis comentaris, la resposta en JSON obtinguda també retorna tot un seguit de dades:

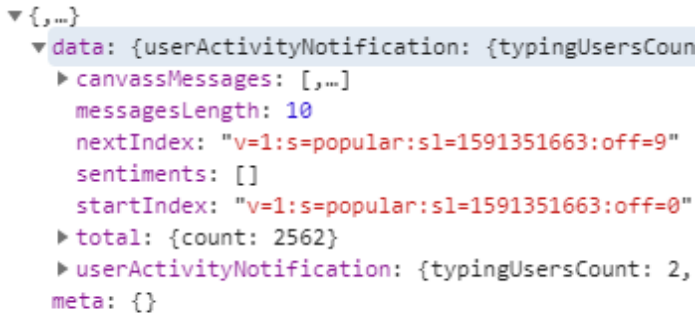

Figura 6. Imatge de resultats de capçalera retornats per l'API de Yahoo! News.

Ens retorna el nombre de comentaris obtinguts amb aquesta cridada, l'index emprat, el proper index, el nombre total de comentaris, el nombre d'usuaris que estan reaccionant en aquest moment a la notícia, així com un apartat de sentiments.

Tal com podem veure, sentiments actualment no retorna valors. No obstant, a l'hora d'iniciar aquest treball, retornava el nombre de comentaris negatius, positius, i neutres que es podien trobar a la notícia. Aquest fet indica clarament que Yahoo! News té implementat un algorisme d'anàlisi de sentiments, mitjançant el qual pondera els comentaris amb un grau de sentiment que properament s'emprarà per la seva classificació.

El fet que no disposem del paràmetre de sentiments, provocarà que disposem d'un cert marge d'error a l'hora de classificar els nostres comentaris, ja que no és l'objectiu d'aquest treball desenvolupar un algorisme d'anàlisi de sentiments, i tot i que ho féssim, aquest no tendria perquè retornar resultats similars als que retorna l'anàlisi de sentiments que fa Yahoo! News. No obstant, tots els altres paràmetres també tenen una gran importancia a l'hora de classificar els comentaris i caldran ser tinguts en compte.

Els camps que també es poden trobar tals com l'índex emprat, el proper índex, i el nombre de comentaris, seran molt útils a l'hora d'extreure les dades de forma iterativa fent ús de l'API de Yahoo! News.

No obstant, és molt important que ens fixem amb la cridada que fa i la forma en que la genera per tal de poder crear les nostres pròpies cridades a aquesta API. Tal com es pot observar a la següent imatge:

Figura 7. Imatge d'una petició a l'API amb el context remarcat.

Tal com podem observar, en aquesta cridada hi ha dues seccions molt importants, els paràmetres que passem a través de la secció path i la cookie emprada.

La secció de la cookie no té una gran importància a l'hora de fer la implementació ja que per fer un scrapper efectiu emprarem els propis divers de Chrome i la llibreria selenium de c# que s'encarregaran de generar totes les cookies necessàries sense cap dificultat. No obstant, a

la secció de paràmetres que es poden trobar a path hem de destacar els l'índex actual, que ens indica a partir de quin comentari cal començar, i la quantitat de comentaris a obtenir. Aquests dos paràmetres seran importants ja que són els que empra Yahoo! News per fer la seva paginació i seran els que emprarem nosaltres per tal de recuperar els comentaris de forma iterativa.

3.5. El paràmetre context de la notícia.

A banda dels paràmetres esmentats anteriorment, cal prestar molta atenció a un paràmetre anomenat context (ressaltat en vermell a la imatge). Aquest paràmetre és el que defineix la notícia sobre la qual es volen obtenir els comentaris. Tot i que seria comprensible emprar el paràmetre guid de la notícia, Yahoo! News a considerat més apropiat crear un paràmetre context per cada notícia amb l'objectiu de recuperar la informació dels comentaris.

El paràmetre context, no ha aparegut en cap moment fins ara al llarg d'aquest document, i per tant fins a hores d'ara no s'ha estudiat la forma de conèixer aquest paràmetre per cada notícia amb l'objectiu d'emprar-lo posteriorment.

Per tal d'obtenir aquest paràmetre en algun lloc, s'ha decidit inspeccionar el codi font de la notícia en qüestió tal com es pot veure a la següent imatge:

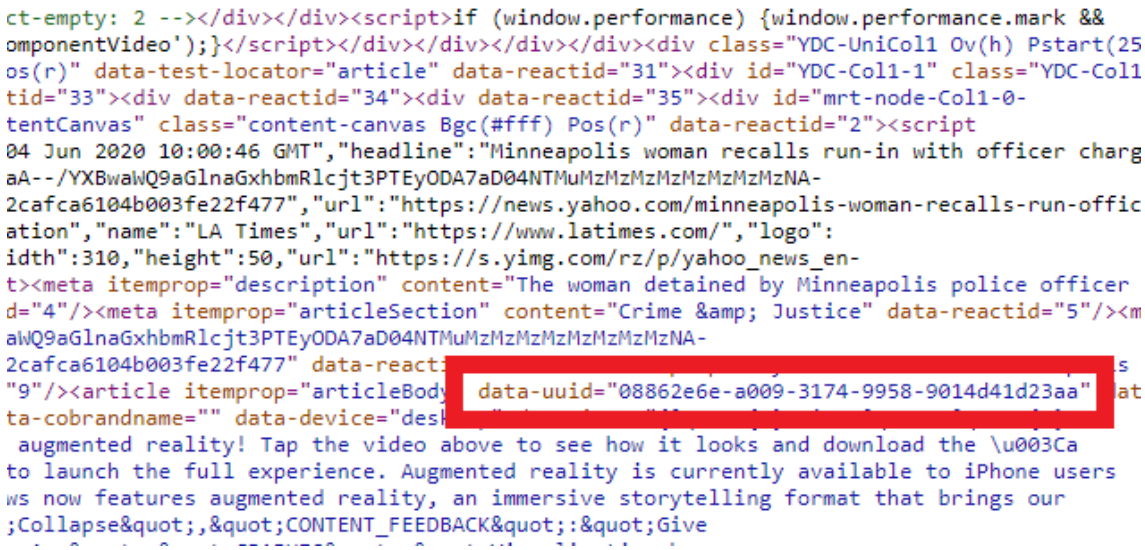

Figura 8. Imatge del codi font de la pàgina amb el paràmetre uuid remarcat.

Tal i com es pot observar dintre del codi font de la notícia, podem trobar un paràmetre anomenat data-uuid el qual sempre es correspon amb el paràmetre context de la notícia. Un cop trobat aquest paràmetre només caldrà obtenir-lo a través del web srapper per tal de poder ser emprat en la posterior obtenció de comentaris de la notícia.

4. Preparació de l'entorn de desenvolupament.

Per tal de poder dur a terme el desenvolupament d'aquesta aplicació, serà necessari definir les eines de les quals farem ús, així com també de la seva configuració.

4.1. Visual Studio.

Per dur a terme aquesta aplicació s'ha pres la decisió de desenvolupar els algorismes genètics en codi c#, ja que presenta una facilitat de desenvolupament i d'integració amb sistemes de bases de dades i repositoris que ens seran molt útils.

En primer lloc descarregarem la versió gratuïta de Visual Studio anomenada Visual Studio Community, la qual disposa d'eines suficients per aquella aplicació que volem desenvolupar.

Tot seguit, un cop descarregat i instal·lat es procedirà a la instal·lació dels paquets NuGet que es detallen a continuació.

- EntityFramework: Aquest paquet serveix per tractar totes les dades que tenim disponibles com a entitats, de tal forma que ens resultarà molt més simple interactuar amb totes les dades que tinguem disponibles.

- MySql.Data: Es tracta d'un connector que implementa les funcions necessàries per connectar el nostre projecte a un motor de bases de dades MySql. Necessitarem aquesta eina ja que la nostra base de dades serà en aquesta tecnologia.

- MySql.Data.EntityFramework: Es tracta d'una eina pont entre els dos paquets anteriorment esmentats. Aquesta eina serà l'encarregada de transformar els resultats obtinguts a través del nostre connector amb la base de dades en entitats que es podran tractar a través de EntityFramework.

- Selenium.WebDriver: Es tracta d'una API capaç d'interactuar amb un navegador d'internet. El que fa és crear una nova instancia d'un navegador, que en el nostre cas serà Chrome, i simula totes les accions que indiquem sobre el propi navegador. Aquest paquet NuGet ens servirà per crear el nostre web scrapper, de tal forma que podrem extreure dades que d'una altra forma no podríem ja que no es trobarien ni a la API de comentaris ni als fitxers RSS.

- Newtonjoft.Json: Aquest paquet serveix per tractar informació en Json. És capaç de serialitzar i deserialitzar dades per tal que siguin tractades amb major facilitat. Aquest paquet ens servirà per processar millor la informació extreta de l'API de comentaris pròpia de Yahoo! News.

- System.ValueTuple: Aquest paquet, tot i que seria prescindible, ens serà útil per crear tuples de dades amb un camp per la ponderació del nostre algorisme i amb un altre camp amb l'entitat dem missatge en sí.

- mXparser: Aquest paquet serveix per interpretar i executar expressions matemàtiques. Això ens resultarà molt útil ja que els resultats del nostre algorisme seran sempre funcions matemàtiques amb tantes variables com paràmetres es considerin.

4.2. Base de dades MySql.

Per tal de poder emmagatzemar les dades que obtindrem de Yahoo News, serà necessari que disposem d'una base de dades. En aquest sentit, per desenvolupar aquest treball, la tecnologia que s'emprarà serà MySql.

S'ha decidit crear les següents tables MySql:

- News: Aquesta tabla servirà per emmagatzemar les notícies de yahoo news, contindrà camps obtinguts a través del web scrapper i camps obtinguts per rss.

- Message: Aquesta tabla servirà per emmagatzemar informació dels comentaris que podem trobar a les notícies. Contindrà una gran quantitat de camps, tant els obtinguts a través de la API pròpia de Yahoo! News com aquells obtinguts a través del web scrapper.

- HeuristicResult: Aquesta tabla servirà per emmagatzemar els resultats d'execució del nostre algorisme genètic. Quan un resultat es consideri que té un fitness apropiat i per tant un marge d'error relativament baix, es desarà en aquesta tabla en forma de funció matemàtica, juntament amb altres paràmetres com el fitness obtingut, el nombre d'iteracions, o la data d'execució.

- Error: Durant l'execució dels nostres algorismes, ja siguin els algorismes genètics que cerquen una expressió matemàtica com els algorismes per extreure informació, ens podem trobar amb una gran quantitat d'errors. Tots aquests errors s'emmagatzemaran en aquesta tabla amb els detalls propis de l'error. D'aquesta forma podrem fer un seguiment dels errors i solucionar-los d'una forma més eficient.

A continuació es pot observar el resultat final del diagrama de la base de dades:

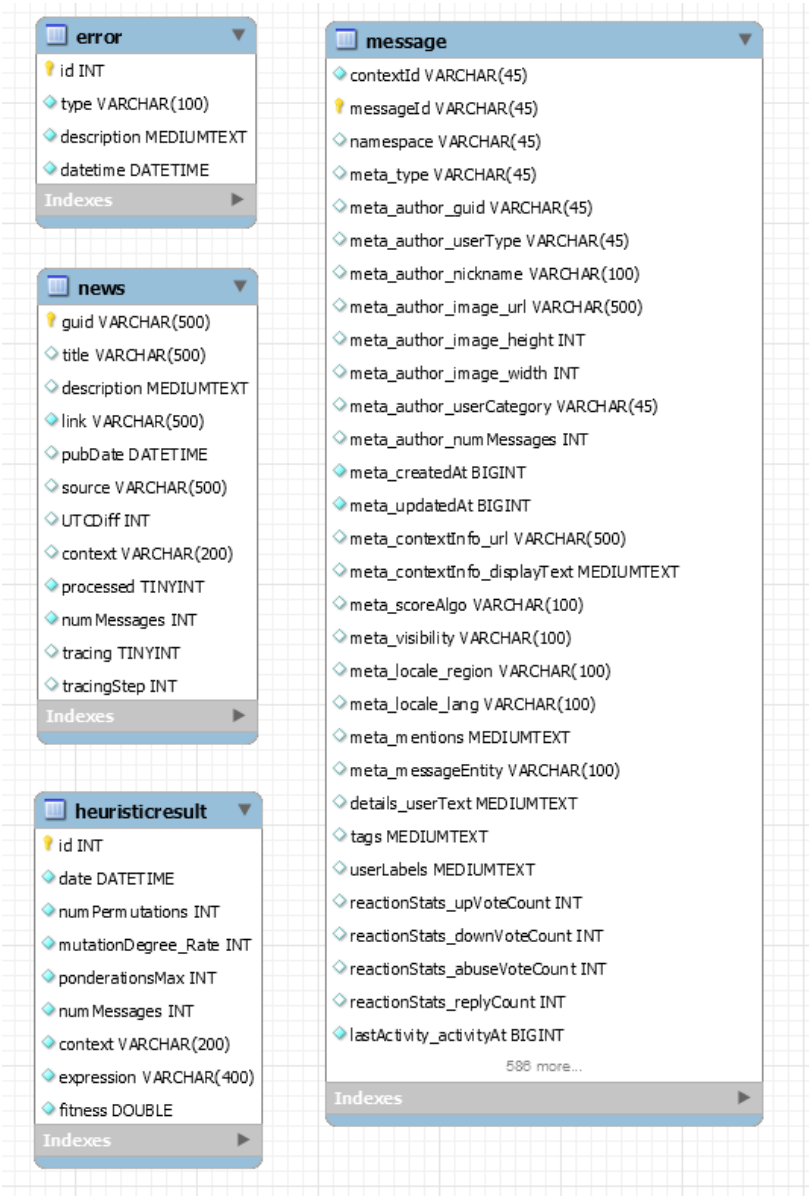

Figura 9. Diagrama de la base de dades del projecte.

4.3. El controlador de versions Git.

Durant aquest treball s'escriuran desenes de milers de línies de codi, i per tant, serà necessari que disposem d'un sistema de control de versions, en primer lloc per no perdre tot el treball realitzat en cas que l'ordinador sofreixi una averia, i en segon lloc, per tal de ser capaços de restaurar versions anteriors en cas que la versió actual no funcioni de la forma esperada.

Per tal de poder fer un control de versions efectiu, el controlador de versions que emprarem serà Git. Git és una plataforma de caràcter gratuït que ens permet crear repositoris de codi, ja siguin públics o privats, i disposa d'integració amb Visual Studio de forma nativa, de tal forma que podrem desar totes les nostres versions d'una forma molt senzilla i intuïtiva.

A continuació es mostren imatges del repositori i de la seva integració amb Visual Studio:

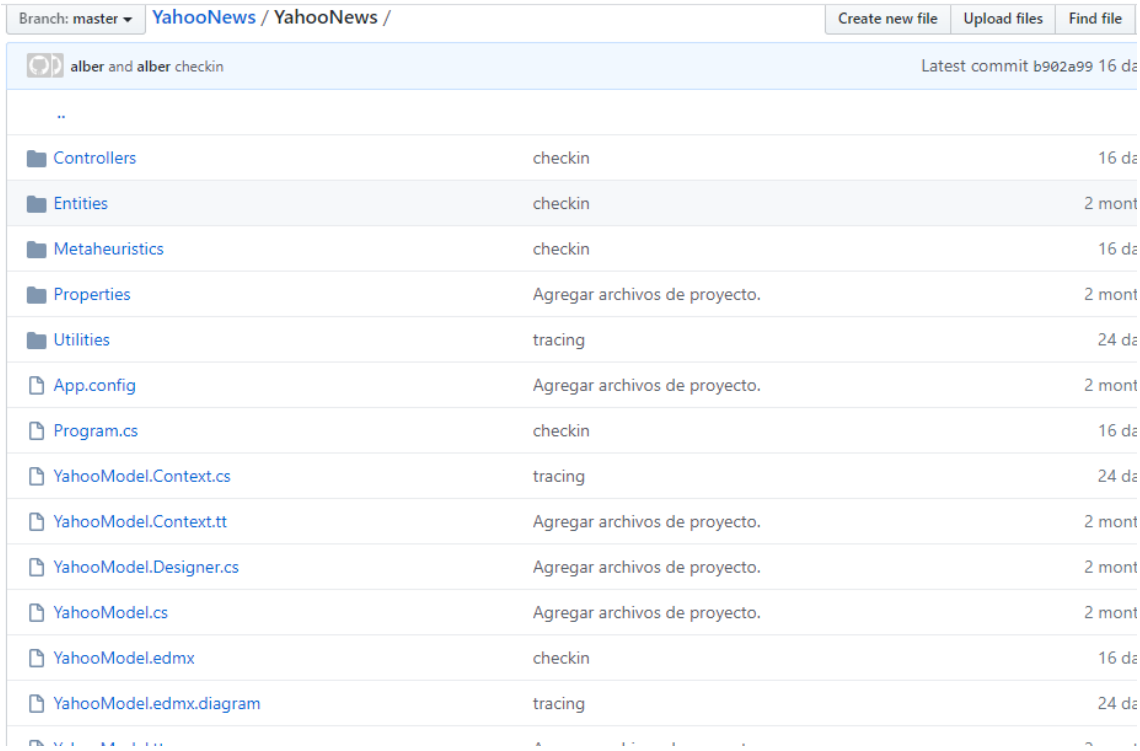

Figura 10. Imatge del repositori.

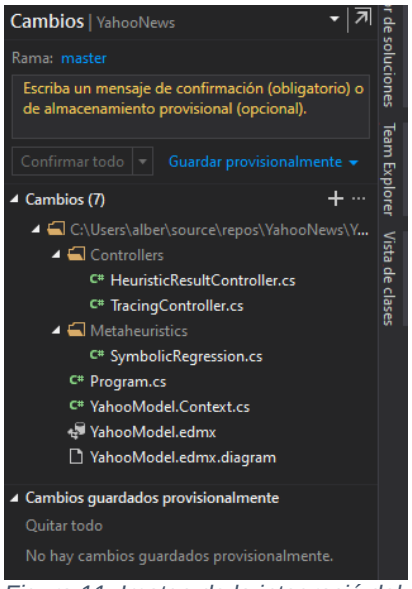

Figura 11. Imatge de la integració del repositori amb Visual Studio.

5. Obtenció de dades.

5.1. Obtenció de dades de les notícies per RSS

Tal i com ja s'ha explicat en els anteriors apartats, Yahoo News! disposa d'un servei RSS per tal de facilitar la redifusió de les notícies de la seva pàgina web. Aquest servei es troba en actualització constant, de tal forma que cada cop que consultem el fitxer RSS el contingut que hi trobarem serà diferent.

Per tal de poder processar correctament un fitxer XML serà necessari fer ús de la biblioteca de classes System.Xml. Aquesta biblioteca inclou diverses classes per tal de poder obtenir un fitxer xml així com tractar-ne les dades obtingudes.

Un cop que el fitxer XML es trobi dintre d'una variable de tipus XmlDocument gràcies a la funció Load, en podrem extreure els nodes dintre d'una variable de tipus XmlNodeList.

Un cop les dades són accessibles en el nostre codi, només caldrà recórrer els ítems de les notícies de forma iterativa, i desar-los a la nostra base de dades amb l'ajuda dels paquets NuGet de Entity framework i les llibreries de Linq.

A continuació es pot veure un breu extracte de codi amb la implementació de les funcions descrites.

```
YahooNews.yahooEntities yahoodb = new YahooNews.yahooEntities();
YahooNews.yahooEntities yahoodbErr = new YahooNews.yahooEntities();

XmlDocument rssXmlDoc = new XmlDocument();
rssXmlDoc.Load(RSS_url);
XmlNodeList rssNodes = rssXmlDoc.SelectNodes("rss/channel/item");

foreach (XmlNode rssNode in rssNodes)
{
    YahooNews.news newsItem = new YahooNews.news();

    newsItem.title = rssNode.SelectSingleNode("title") != null ? rssNode.SelectSingleNode("title").InnerText : "";
    newsItem.link = rssNode.SelectSingleNode("link") != null ? rssNode.SelectSingleNode("link").InnerText : "";
    newsItem.description = rssNode.SelectSingleNode("description") != null ? rssNode.SelectSingleNode("description").InnerText : "";
    newsItem.guid = rssNode.SelectSingleNode("guid") != null ? rssNode.SelectSingleNode("guid").InnerText : "";
    newsItem.source = rssNode.SelectSingleNode("source") != null ? rssNode.SelectSingleNode("source").InnerText : "";
    string pubDateString = rssNode.SelectSingleNode("pubDate") != null ? rssNode.SelectSingleNode("pubDate").InnerText : "";
```

Figura 12. Imatge de les funcions implementades per obtenir dades per RSS.

5.2. Obtenció de dades de les notícies per web scrapper.

Tal com ja s'ha comentat en anteriors apartats, no totes les dades que necessitem de les notícies es poden obtenir per RSS. Això implica que cal una altra forma d'obtenir informació que no depengui exclusivament d'aquella informació que podem obtenir per RSS. Aquesta forma d'extreure dades és el web scrapper.

Per desenvolupar el web scrapper, farem ús de l'eina Selenium.WebDriver anteriorment esmentada. Aquesta tecnologia ens permetrà obtenir les següents dades referents a la notícia que no es poden trobar dintre del fitxer RSS:

- Context de la notícia: Tal com hem vist anteriorment, necessitem obtenir el context d'una notícia, ja que serà necessari per tal d'obtenir els comentaris a la notícia. Aquest context es troba en un atribut anomenat data-uuid que podrem obtenir mitjançant el nostre web scrapper.

- Nombre de comentaris: De la mateixa forma que passa amb el context de la notícia, el nombre de comentaris que podem trobar en una notícia tampoc es troba entre la informació que podem obtenir per RSS. Aquesta dada ens resulta de gran interès ja que ens permetrà decidir de quines notiíies volem obtenir els comentaris, ja que no serà de gran interès obtenir els comentaris d'una notícia que disposi de pocs comentaris.

Per tal de fer ús del nostre web scrapper, en primer lloc, crearem una instancia de tipus IWebDriver, indicarem l'URL a la qual volem navegar, que en el nostre cas serà l'url de la notícia en si, i llavors seleccionarem els clicks que volem simular i els atributs que volem obtenir.

A continuació es mostra un breu exemple de com obtenir l'atribut data-uuid a través del nostre web scrapper.

```
try
{
    driver.Url = newsItem.link;
    System.Threading.Thread.Sleep(1000);
    IWebElement contextElement = driver.FindElement(By.XPath(".//*[@id='Col1-0-ContentCanvas']/article"));
    string newsContext = contextElement.GetAttribute("data-uuid");
    newsItem.context = newsContext;
    yahoodb.SaveChanges();
    Console.WriteLine("Context properly parsed {0}", newsItem.link);
}
catch (NoSuchElementException e)
{
    Console.WriteLine("Selenium exception. Error finding context. Details: {0}", e);
    yahoodbErr.error.Add(new error { description = string.Format("Selenium exception. Error finding context.
    yahoodbErr.SaveChanges();
}
```

Figura 13. Imatge de les funcions implementades per obtenir dades per Web Scrapper.

Com es pot observar a la imatge, si la selecció de l'atribut és correcta, aquest es desarà a través de Linq i Entity Framework a la nostra base de dades de notícies com a context de la notícia, mentre que si existeix algun error, desarem els detalls de l'error a la nostra tabla de errors dintre de la base de dades.

5.3. Obtenció de dades dels comentaris a través de l'API.

Tal com ja s'ha explicat anteriorment, la pagina web de Yahoo! News fa ús d'una API pròpia per obtenir les dades dels comentaris amb major comoditat. Tot i que aquesta API no és accessible a través de peticions de tercers, si que es poden emular les peticions a través del propi web scrapper tal com es faria emprant Yahoo! News.

L'API propia de Yahoo! News, pot retornar fins a 20 comentaris per cada petició rebuda, de tal forma que haurem de fer les peticions de forma iterativa recorrent tots els comentaris retornats amb cada petició.

Per dur a terme aquesta funcionalitat, en primer lloc es crearà un primer bucle on es recuperaran els missatges en conjunts de 20, i en segon lloc es recorrerà cada missatge per processar-lo correctament.

Tot seguit podem observar un extracte de codi per tal d'obtenir els missatges.

```
//Parse first messages
urlFirstMessages = string.Format(urlFirstMessages, news.context);
driver.Url = urlFirstMessages;
driver.Manage().Window.Maximize();
IWebElement jsonContainer = driver.FindElement(By.XPath("/html/body/pre"));
dynamic jsonMessages = JsonConvert.DeserializeObject(jsonContainer.GetAttribute("innerText"));
int position = 1;
//long unixTime = DateTimeOffset.Now.ToUnixTimeSeconds();
foreach (dynamic jsonMessage in jsonMessages.canvassMessages)
{
    AddJsonMessage(jsonMessage, position, unixTimestamp);
    position++;
}

//Loop parse next messages
for (int i = 1; i <= 98; i = i + 2)
{
    string urlCommentsPage = string.Format(urlMessages, news.context, i);
    driver.Url = urlCommentsPage;
    driver.Manage().Window.Maximize();
    jsonContainer = driver.FindElement(By.XPath("/html/body/pre"));
    jsonMessages = JsonConvert.DeserializeObject(jsonContainer.Text);
    foreach (dynamic jsonMessage in jsonMessages.canvassMessages)
    {
        AddJsonMessage(jsonMessage, position, unixTimestamp);
        position++;
    }

}
```

Figura 14. Imatge de les funcions implementades per obtenir dades a través de l'API.

Tal com es pot observar, els primers 20 missatges es recorren fora del bucle ja que el seu índex d'inici és NULL mentre que els demés missatges es recorren dintre d'un bucle amb un límit de 98. Aquest límit és per obtenir els primers 1.000 missatges d'una notícia, ja que com hem vist anteriorment el nombre de missatges es trobara dintre de la tabla news de la nostra base de dades.

Un cop obtingut cada missatge, aquest serà processat mitjançant una altra funció per tal de desar tots els seus valors a la tabla de messages de la nostra base de dades gracies a Entity Framework.

A continuació es pot veure una part de la funció de tractament de cada missatge.

```
2 referencias
public static void AddJsonMessage(dynamic jsonMessage, int position, long unixTimeStamp)
{
    YahooNews.yahooEntities yahoodb = new YahooNews.yahooEntities();
    YahooNews.yahooEntities yahoodbErr = new YahooNews.yahooEntities();
    YahooNews.message message = new YahooNews.message();
    Console.WriteLine("Processing message {0}", jsonMessage.messageId);
    message.contextId = jsonMessage.contextId;
    message.messageId = jsonMessage.messageId;
    message.@namespace = jsonMessage.namesapce;
    message.meta_type = jsonMessage.meta.type;
    message.meta_author_guid = jsonMessage.meta.author.guid;
    message.meta_author_userType = jsonMessage.meta.author.userType;
    message.meta_author_nickname = jsonMessage.meta.author.nickname;
```

Figura 15. Tractament dels missatges obtinguts en format JSON.

Tal com es pot observar quan es recuperen els comentaris d'una notícia, un dels paràmetres que es recuperen és la posició en la qual es troba aquest missatge. Aquesta informació s'obté mitjançant l'índex del bucle, i és una dada molt important per tal de conèixer millor l'algorisme de classificació que empra Yahoo! News. El fet que aquesta dada s'obtengui d'aquesta forma es deu al fet que l'API de comentaris no retorna la posició de cada comentari, no obstant, si que els retorna en el mateix ordre que es mostren a la web, i per tant resulta relativament senzill obtenir-ne la posició.

Tal com s'ha pogut comprovar al llarg d'aquest treball, en algunes ocasions es retornen missatges de forma duplicada, i per tant un mateix missatge pot apareixer en dues posicions diferents. Una de les decisions que s'han hagut de prendre en aquest treball, és la d'excloure els missatges repetits, de tal forma que un missatge només es mostri el primer cop que es rep a través de l'API, ja que el nostre objectiu és determinar la millor posició que pot obtenir un comentari determinat. Per tal de dur a terme aquest objectiu, és necessari comprovar l'identificador de cada missatge, i en cas que ja es trobi a la nostra base de dades, descartar-lo tal com podem veure a continuació.

```
try
{
    if (!yahoodb.message.Any(m => m.messageId == message.messageId))
    {
        yahoodb.message.Add(message);
        yahoodb.SaveChanges();
        Console.WriteLine("Message added to database.");
    }
    else
    {
        Console.WriteLine("DUPLICATED, Message not added to database.");
        YahooNews.error error = new YahooNews.error();
        error.type = "Message duplicated error";
        error.description = string.Format("DUPLICATED, Message not added to database. ID: {0}, Positi
        yahoodbErr.error.Add(error);
        yahoodbErr.SaveChanges();
    }
}
```

Figura 16. Tractament dels missatges duplicats.

6. Creació de gens i cromosomes.

6.1. Selecció de variables.

Per tal de poder obtenir una aproximació matemàtica a la funció emprada per Yahoo! News, el primer que cal fer és seleccionar aquells paràmetres que hem aconseguit mitjançant la prèvia extracció de dades que es tindran en compte a l'hora d'establir la nostra ponderació.

L'objectiu és crear una funció matemàtica amb les variables ponderades de tal forma que el resultat de la seva aplicació sobre cada comentari generi un resultat numèric que posteriorment pugui ser ordenat i s'aproximi el màxim possible a l'ordre retornat per Yahoo! News.

Les variables seleccionades són les que es presenten a continuació:

- **meta_createdAt:** Representa la data de creació del comentari en forma de valor numèric (unix timestamp).

- **reactionStats_upVoteCount:** Representa el nombre total de vots positius a un determinat comentari.

- **reactionStats_downVoteCount:** Representa el nombre total de vots negatius a un determinat comentari.

- **reactionStats_abuseVoteCount:** Representa el nombre total de vots abusius (fraudulents per part d'un mateix usuari) a un determinat comentari.

- **reactionStats_replyCount:** Representa el nombre total de respostes a un determinat comentari.

- **lastActivity_activityAt:** Representa l'instant de la darrera interacció que ha tingut un comentari en forma de valor numèric (unix timestamp).

Totes les variables seleccionades, es corresponen amb les variables numèriques extretes dels comentaris d'una mateixa notícia, i per tant poden tenir certa rellevància a l'hora d'establir l'ordre en que es presenten aquests comentaris.

Tal com es podrà comprovar posteriorment, aquests paràmetres no són els únics que cal tenir en compte, ja que Yahoo! News fa ús de més variables a l'hora d'establir l'ordre. No obstant, en una primera instància provarem d'esbrinar la relació entre aquests paràmetres i l'ordre de comentaris així com el seu marge d'error.

Per tal de poder tractar amb certa facilitat aquests paràmetres, es crearà una classe de tipus enum amb el conjunt de noms de les mateixes tal com es mostra a continuació.

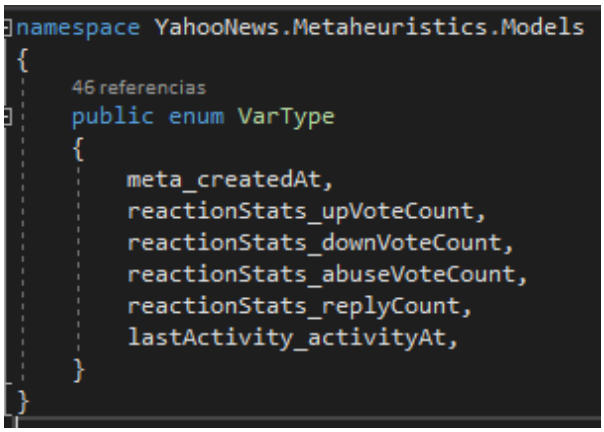

Figura 17. Implementació de l'enumerable VarType que indica els paràmetres que pot tenir un gen.

6.2. Selecció de les operacions.

Per tal de poder generar una expressió matemàtica en forma de funció mitjançant els paràmetres seleccionats, serà necessari que no solament tinguem definits els nostres paràmetres, sinó també les operacions que emprarem per tal de ponderar les nostres variables.

Al llarg d'aquest treball, s'han emprat múltiples operacions que es seleccionaven de forma aleatòria de la mateixa forma que es seleccionen els valors de ponderació dintre d'un algorisme genètic. Després de moltes proves, els millors resultats s'han obtingut amb les següents operacions:

- **AddPonderatedVar:** Aquesta operació serveix per afegir ponderacions positives a les nostres variables. Es seleccionarà el valor de la ponderació i es multiplicarà a la variable com a nombre decimal positiu.

- **SubtractPonderatedVar:** Aquesta operació serveix per afegir ponderacions negatives a les nostres variables. Es seleccionarà el valor de la ponderació i es multiplicarà a la variable com a nombre decimal negatiu.

- **AddTimeDifferenceVar:** Aquesta operació s'empra amb les variables de tipus temporal, i consisteix en multiplicar el valor de la ponderació en forma de nombre decimal positiu per una operació on s'obté el nombre d'hores que han passat des que s'ha creat el comentari fins que s'ha retornat en una determinada posició.

- **SubstractTimeDifferenceVar:** Aquesta operació s'empra amb les variables de tipus temporal, i consisteix en multiplicar el valor de la ponderació en forma de nombre decimal negatiu per una operació on s'obté el nombre d'hores que han passat des de que s'ha creat el comentari fins que s'ha retornat en una determinada posició.

La forma de definir aquestes operacions dintre del nostre codi serà anàloga a la forma en la que s'han definit les variables a tenir en compte. Per tant es crearà una classe de tipus enum on es definiran les operacions a realitzar tal com es mostra a continuació:

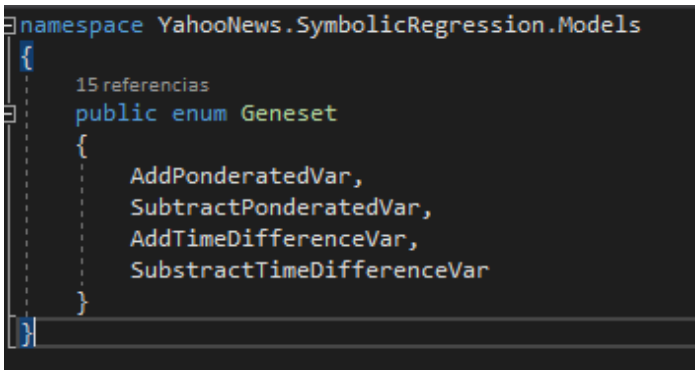

Figura 18. Implementació de l'enumerable Geneset que indica les operacions que pot tenir un gen.

Tal com es pot observar amb el nom d'aquesta classe, aquestes operacions conformaran el geneset dels nostres gens que crearem per tal d'executar el nostre algorisme metaheuristic.

6.3. Definició dels gens.

Els gens que emprarem en el nostre algorisme genètic es composaran bàsicament de 3 variables que es detallen a continuació.

- **Operation:** L'operació definida anteriorment dintre de la classe geneset.
- **Value:** El valor aleatori que generarà el nostre algorisme i que servirà com a ponderació dels paràmetres.
- **VarType:** Els paràmetres definits anteriorment que formen part dels propis comentaris.

Per tal de poder tractar amb comoditat els gens, es crearà una nova estructura de dades mitjançant una classe anomenada Gene tal com es mostra a continuació.

```
namespace YahooNews.SymbolicRegression.Models
{
    11 referencias
    public class Gene
    {
        2 referencias
        public Gene(Geneset operation, VarType varType, double value)
        {
            Operation = operation;
            Value = value;
            VarType = varType;
        }
        4 referencias
        public Geneset Operation { get; set; }
        15 referencias
        public VarType VarType { get; set; }
        11 referencias
        public double Value { get; set; }
    }
}
```

Figura 19. Implementació de l'estructura de dades d'un gen.

6.4. Resultats d'operacions d'un gen.

Tal com hem indicat en els anteriors apartats, els paràmetres seleccionats es combinaran amb les operacions definides, de forma que cada paràmetre es trobarà ponderat dintre d'una funció matemàtica resultant.

Per dur a terme aquesta funció i obtenir-ne el resultat, serà necessari crear un algorisme que s'encarregui de gestionar totes les operacions de cada gen. Aquestes operacions es trobaran a la funció Compute dintre de la classe Computation tal i com es mostra a continuació.

```
1 referencia
public static double Compute(List<Gene> Genes, message message)
{
    double y = 0;

    foreach (Gene g in Genes)
    {
        switch (g.Operation)
        {
            case Geneset.AddPonderatedVar:
                y += (g.Value * GetVarValue(g.VarType, message));
                break;
            case Geneset.SubtractPonderatedVar:
                y -= (g.Value * GetVarValue(g.VarType, message));
                break;
            case Geneset.AddTimeDifferenceVar:
                y += (g.Value * ((message.timestamp - GetVarValue(g.VarType, message)) / 3600));
                break;
            case Geneset.SubstractTimeDifferenceVar:
                y -= (g.Value * ((message.timestamp - GetVarValue(g.VarType, message)) / 3600));
                break;
            default:
                break;
        }
    }

    return y;
}
```

Figura 20. Implementació de la funció Compute.

6.5. Definició dels cromosomes.

Els algorismes de selecció genètica fan ús d'una estructura de dades anomenada cromosoma. Un cromosoma és una estructura de dades que consisteix bàsicament en un conjunt de gens, cadascun d'ells amb les variables anteriorment esmentades de valor, operació, i paràmetre.

Per dur a terme un cromosoma simple, seria suficient amb definir un llistat de gens que el conformen, no obstant, en el nostre cas, s'ha decidit implementar tot un seguit de variables i operacions sobre un cromosoma que seran necessaris per tal de poder assolir l'objectiu d'aconseguir els millors resultats en la funció matemàtica resultant. Aquestes variables seran el llistat de paràmetres a tenir en compte, el llistat de gens que conformen el cromosoma, i en rang de les ponderacions aplicables a cadascun dels termes de la funció, mentre que les operacions seran tals com la generació d'un pare, la mutació, la creació d'un gen, la clonació, etc. Aquests paràmetres i funcions es definiran en els següents apartats.

6.6. Variables d'un cromosoma.

Les variables que conformen un cromosoma són les que s'expliquen a continuació:

- **Llistat de gens:** Tot cromosoma es conforma principalment per un conjunt de gens que l'algorisme de selecció genètica s'encarregarà de modificar per tal de millorar els resultats de l'execució.

- **Llistat de paràmetres:** En el nostre cas, ens interessa que els nostres cromosomes tinguin solament un gen de cada tipus de paràmetre, i per millorar l'escalabilitat del nostre codi amb l'objectiu de poder afegir o llevar paràmetres en un futur, s'ha creat aquesta variable, de tal forma que el nostre algorisme genètic crearà tants gens com paràmetres contingui la nostra llista de paràmetres.

- **Ponderacions:** Cada tipus de paràmetre estarà associat a un rang de ponderacions. Aquest fet produirà que quan es doni una mutació sobre un gen en concret, aquest no pugui adquirir qualsevol valor sinó que s'hagi de limitar a un valor dintre del rang de ponderacions associades al paràmetre del propi gen.

6.7. Creació d'un cromosoma pare.

En tot algorisme genètic, és necessari que en els primers passos es generin un o varis cromosomes pare, i per tant dintre de la classe del propi cromosoma requerirem d'una funció amb l'objectiu de crear un cromosoma pare.

En el nostre cas, i tal com s'ha explicat anteriorment, requerirem de la creació de tants gens com tipus de paràmetres existeixin, cadascuna d'elles ponderades dintre d'un rang de ponderacions determinat. En el cas

dels paràmetres, sempre seran els mateixos, no obstant en el cas de les ponderacions aquestes poden variar en funció dels nivells d'exploració o d'explotació de la funció que vulguem en cada pas.

Tot seguit s'adjunta una part del codi del cromosoma, on es pot veure com es genera un cromosoma i en què consisteix la funció de generar un cromosoma pare.

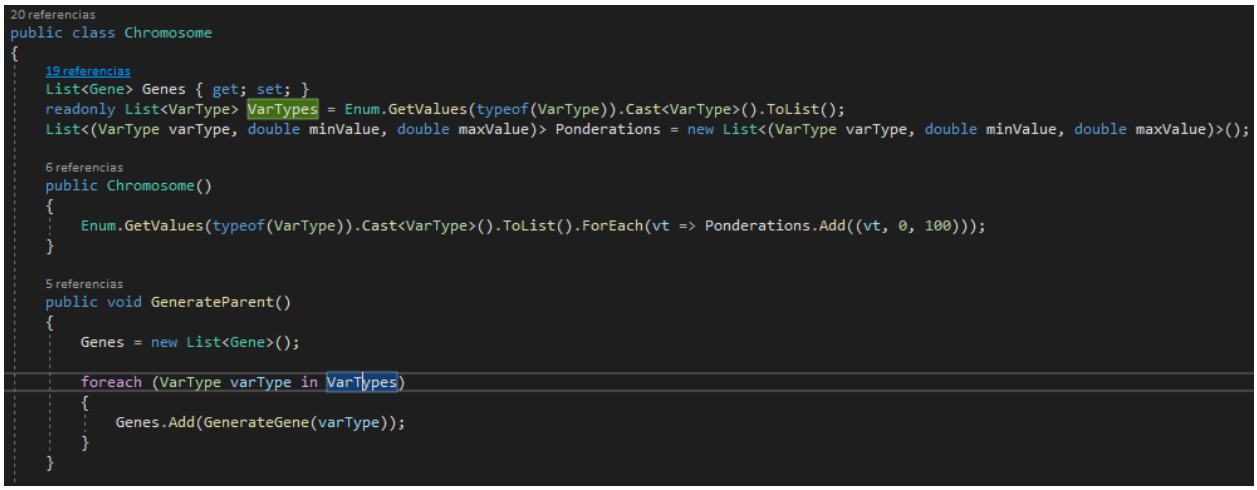

Figura 21. Implementació de l'estructura de dades d'un cromosoma i de la funció de generació.

En la imatge anterior es pot comprovar que la variable Vartypes, que conté els paràmetres que pot adquirir un gen és una variable fixa, mentre que les ponderacions poden canviar, i per defecte es trobaran establertes entre 0 i 100 quan es crea el cromosoma. Així mateix es pot observar com es genera un cromosoma pare amb la creació d'un gen per cada tipus de variable.

6.8. Generació d'un gen.

Tal com s'ha pogut observar anteriorment, per crear un cromosoma és necessari fer ús d'una funció determinada per tal de crear cadascun dels gens, de tal forma que es creïn tants de gens com paràmetres es vulguin contemplar a la funció matemàtica resultant.

La funció per generar un gen, rep com a paràmetre el tipus de variable que contindrà el gen que es vol crear. En un primer moment, tant l'operació com el valor de la ponderació es seleccionaven de forma aleatòria, no obstant, després de diverses execucions s'ha pogut comprovar que cada tipus de paràmetre es relaciona amb un tipus d'operació concreta, i per tant per tal d'optimitzar el nostre algorisme genètic s'ha pres la decisió de relacionar cada tipus de paràmetre amb sa seva operació corresponent.

Un cop tenim definits el tipus de paràmetre i l'operació del gen, només cal establir aleatòriament el valor de la ponderació dintre del rang de ponderacions establert, i ja podrem retornar el gen amb tots els seus valors.

A continuació es pot observar la funció de creació d'un gen:

```
5 referencias
public Gene GenerateGene(VarType varType)
{
    Utilities.SecureRandom rnd = new Utilities.SecureRandom();

    //SET GENESET
    Geneset geneset = new Geneset();
    string vartypeString = Enum.GetName(typeof(VarType), varType);
    if (vartypeString.Contains("abuseVoteCount") || vartypeString.Contains("downVoteCount"))
    {
        geneset = Geneset.AddPonderatedVar;
    }
    else if (varType == VarType.meta_author_numMessages || vartypeString.Contains("replyCount") || vartypeStri
    {
        geneset = Geneset.SubtractPonderatedVar;
    }
    else if (varType == VarType.lastActivity_activityAt || varType == VarType.meta_createdAt)
    {
        geneset = Geneset.AddTimeDifferenceVar;
    }

    //SET RANDOM VALUE
    double geneValue = (rnd.NextDouble(Ponderations.Where(p => p.varType == varType).First().minValue, Pondera
    return new Gene(geneset, varType, geneValue);
}
```

Figura 22. Implementació de la funció per generar un gen.

6.9. Clonació d'un cromosoma.

Els algorismes genètics necessiten fer canvis en els cromosomes tal com ho faria la naturalesa, ja sigui a través de mutacions o a través de combinacions genètiques per creuament d'altres cromosomes. Sigui quin sigui el cas, quan hem creat un nou cromosoma el que volem saber és si aquest nou cromosoma és millor o pitjor que el seu cromosoma pare.

En el cas del nostre algorisme genètic, els cromosomes són mutats per tal de crear nous cromosomes, aquest fet provoca que quan es muta un cromosoma, es perd la informació genètica d'aquell cromosoma abans que fos mutat, i per tant, aquest fer suposaria un problema a l'hora de voler esbrinar quins dels cromosomes ofereix millors resultats.

Per tal de poder comparar un cromosoma amb si mateix abans de ser mutat, cal que fem una clonació del cromosoma, de tal forma que es generi un cromosoma nou que sigui completament idèntic a l'anterior. Per tal de dur a terme aquesta operació, s'ha implementat una funció de clonació dintre de la pròpia classe del cromosoma tal i com es mostra a continuació.

```
7 referencias
public Chromosome Clone()
{
    Chromosome chromosome = new Chromosome();
    chromosome.Genes = new List<Gene>();
    foreach (Gene g in Genes)
    {
        chromosome.Genes.Add(new Gene(g.Operation, g.VarType, g.Value));
    }

    chromosome.Ponderations = new List<(VarType varType, double minValue, double maxValue)>();
    foreach ((VarType varType, int minValue, int maxValue) ponderation in Ponderations)
    {
        chromosome.Ponderations.Add((ponderation.varType, ponderation.minValue, ponderation.maxValue));
    }

    return chromosome;
}
```

Figura 23. Implementació de la funció de clonació.

Tal com es pot observar, s'estableixen com a llistat de gens els mateixos gens del cromosoma pare, i com a llistat de ponderacions les mateixes que el cromosoma pare. Els tipus de paràmetres no s'estableix en la clonació ja que com hem vist es tracta d'una variable fixa de només lectura que és la mateixa en tots els cromosomes.

6.10. Mutació d'un cromosoma.

Tal com ja s'ha explicat en anteriors apartats, una de les formes de crear nous cromosomes és a través de la mutació de cromosomes ja existents. En el nostre algorisme genètic es farà ús d'aquesta tècnica per tal de millorar els resultats de la funció matemàtica resultant.

Per tal de crear una mutació sobre un dels cromosomes existents, crearem una funció que seleccioni aleatòriament un gen del cromosoma, i que generi un nou gen que en substituirà el gen seleccionat, mantenint el tipus de variable del gen original per tal que només es modifiqui el valor de la ponderació i no el tipus de variable. D'aquesta forma haurem generat un cromosoma lleugerament alterat en un dels seus gens, el que es coneix com a mutació. Aquest cromosoma el podrem comparar posteriorment amb el seu cromosoma pare per tal d'avaluar-ne la idoneïtat.

A continuació es pot observar el funcionament de la nostra funció de mutació:

```
public void Mutate()
{
    Utilities.SecureRandom rnd = new Utilities.SecureRandom();
    int index = rnd.Next(0, Genes.Count);
    Genes[index] = GenerateGene(Genes[index].VarType);
}
```

Figura 24. Implementació de la funció de mutació.

6.11. Avaluació del cromosoma: Funció fitness.

Per tal de poder avaluar els nostres cromosomes, és necessari que disposem d'un indicador numèric sobre l'acompliment d'aquests respecte al nostre objectiu final.

El nostre objectiu final és obtenir un cromosoma que aproximi al màxim possible l'ordre amb el que Yahoo! News retorna un llistat de comentaris. Per tant, el que cal fer és donar un valor numèric a cada cromosoma per tal de poder comparar-lo amb un altre cromosoma.

Per poder donar un valor numèric a la idoneïtat de l'ordre, el primer que haurem de fer és definir aquest ordre, i aquí és on entren en joc els paràmetres, operacions, i ponderacions, de cadascun dels gens del cromosoma aplicats a cadascun dels comentaris.

Es tracta de crear una llista de comentaris ponderats mitjançant una variable que anomenarem Score, i que contindrà el resultat de l'operació matemàtica definida per un cromosoma aplicada a un comentari concret. Per tal de poder obtenir el valor Score de cada comentari, farem ús de la funció Compute explicada al punt 6.4 d'aquest document.

Un cop disposem del llistat amb tots els comentaris i la seva valoració dintre de la variable Score, caldrà que ordenem aquesta llista i que en comparem l'ordre obtingut mitjançant el cromosoma amb l'ordre real retornat per Yahoo! News. la diferencia que hi hagi entre la posició real del comentari i la posició calculada amb el cromosoma s'afegirà iterativament a una variable anomenada fitness. Un cop recorreguts tots els comentaris, disposarem del fitness total d'un algorisme. Com més alt sigui aquest fitness, més grans han sigut les diferències entre les ubicacions reals dels comentaris i les ubicacions calculades fent ús del cromosoma actual.

A continuació podem observar la funció per obtenir el fitness d'un cromosoma concret:

```
8 referencias
public int GetFitness(List<message> messages)
{
    int fitness = 0;

    List<(double scoreAlgorithm, message message)> ponderatedMessages = new List<(double scoreAlgorithm, message message)>();
    foreach (message message in messages)
    {
        ponderatedMessages.Add((ComputationManager.Compute(Genes, message), message));
    }

    ponderatedMessages = ponderatedMessages.OrderBy(pm => pm.scoreAlgorithm).ToList();

    foreach ((double scoreAlgorithm, message message) ponderatedMessage in ponderatedMessages)
    {
        fitness -= Math.Abs(ponderatedMessages.IndexOf(ponderatedMessage) - messages.IndexOf(ponderatedMessage.message));
    }

    return fitness;
}
```

Figura 25. Implementació de la funció per obtenir el fitness d'un cromosoma.

7. Implementació de l'algorisme genètic.

Fins ara s'han estat desenvolupant i implementant les estructures de dades i les funcions que requerirà el nostre algorisme genètic per tal d'obtenir la funció matemàtica que millor s'aproximi al nostre objectiu, que no és altre que classificar una llista de comentaris de la forma més semblant possible a la classificació obtinguda per Yahoo! News.

Un cop disposem de totes les estructures de dades i de les funcions necessàries, ja és possible desenvolupar el nostre algorisme genètic. Els pròxims punts d'aquest document tractaran de donar solució a alguns dels problemes inherents al nostre cas.

7.1. Algorisme genètic simple.

En primer lloc, es desenvoluparà un algorisme genètic per tal d'iterar solucions candidates que aniran mutant fins a trobar la funció objectiu que més s'apropi a la classificació real de comentaris.

Aquest algorisme, comença creant un cromosoma que contindrà un conjunt de gens amb la seva ponderació per defecte de 0 a 100. Un cop creat aquest cromosoma mitjançant la funció GenerateParent, es tracta de poder mutar aquest cromosoma i comparar-lo amb la seva versió anterior. Si la nova versió disposa d'un millor fitness, el cromosoma pare passarà a ser una clonació del fill, en cas contrari el cromosoma pare mantindrà el seu valor i el cromosoma fill serà descartat.

Per tal de dur a terme aquests passos de forma iterativa, es farà ús de les funcions GenerateParent, Mutare i Clone.

Tot seguit es pot observar la implementació d'aquest algorisme.

```
Chromosome parent = new Chromosome();
parent.GenerateParent();
int parentFitness = parent.GetFitness(messages);

for (int i = 0; i < 10000000; i++)
{
    Chromosome child = parent.Clone();
    child.Mutate();
    int childFitness = child.GetFitness(messages);

    if (childFitness > parentFitness)
    {
        parent = child.Clone();
        parentFitness = childFitness;
    }
}
```

Figura 26. Implementació d'un algorisme genètic simple.

Tal com es pot observar, un cop finalitzades 10 milions de permutacions, el cromosoma pare disposarà de la solució que més s'apropa a la nostra funció objectiu.

7.2. Algorisme genètic amb auto-increment d'exploració.

Un dels principals problemes que podem trobar dintre del nostre algorisme genètic, és que si les modificacions que fem en la nostra solució candidata, els trobarem constantment explotant un màxim o un mínim local, i serà molt difícil o tal vegada impossible sortir d'aquest màxim o mínim local.

En el nostre cas, com que es tracta d'una mutació en una sola variable i volem fer 10 milions de permutacions, és molt fàcil que arribats a 1 milió de permutacions aquest arribi a un mínim local i tota la resta de permutacions siguin una explotació constant d'aquest.

Per tal de donar solució a aquest fet, s'ha decidit modificar la funció de mutació explicada en el punt 6.10 d'aquest document, de tal forma que la funció de mutació tingui un paràmetre de grau de mutació, i la mutació sigui més o menys elevada en funció d'aquest grau. D'aquesta forma, es pot donar una major exploració en el conjunt de la funció o una major explotació d'un màxim o un mínim local.

Aquí es mostra la funció de mutació modificada amb un paràmetre de grau de mutació.

```
2 referencias
public void Mutate(int mutationDegree)
{
    for (int i = 0; i < mutationDegree; i++)
    {
        Utilities.SecureRandom rnd = new Utilities.SecureRandom();
        int index = rnd.Next(0, Genes.Count);
        Genes[index] = GenerateGene(Genes[index].VarType);
    }
}
```

Figura 27. Modificació de la funció de mutació amb grau de mutació.

Tal com es pot observar, el paràmetre mutationDegree serveix per definir la quantitat de vegades que es seleccionarà aleatoriament un get per tal d'aplicar-li la mutació. Com més elevat sigui aquest paràmetre, més probabilitats tindrem d'obtenir una major mutació i per tant obtindrem una major exploració de l'espai de solucions.

Per tal de poder fer un bon ús d'aquesta funcionalitat, el nostre algorisme ha de saber triar correctament entre els graus d'exploració de tot l'espai de solucions i els graus d'explotació d'un mínim local concret. El nostre objectiu és que la nostra funció exploti un mínim local fins un cert punt, i arribats a aquest punt incrementi progressivament la seva exploració en detriment de l'explotació. D'aquesta forma podrem garantir que un mínim

local haura sigut suficientment explotat abans d'augmentar-ne l'exploració.

Per tal de dur a terme aquesta funcionalitat, modificarem l'algorisme genètic simple desenvolupat en el punt 7.1 d'aquest document, i afegirem les variables i funcionalitats que es poden observar a la següent imatge.

```
Chromosome parent = new Chromosome();
parent.GenerateParent();
int parentFitness = parent.GetFitness(messages);

for (int i = 0; i < numPermutations; i++)
{
    mutationDegree = 1;
    if (i % (numPermutations / 10) == 0) {
        mutationDegree += 1;
    }
    Chromosome child = parent.Clone();
    child.Mutate(mutationDegree);
    int childFitness = child.GetFitness(messages);

    if (childFitness > parentFitness)
    {
        mutationDegree = 1;
        parent = child.Clone();
        parentFitness = childFitness;
    }
}
```

Figura 28. Modificació de l'algorisme genètic amb auto-increment d'exploració.

Tal i com es pot observar, en aquest cas fem ús de les variables numPermutations i mutationDegree, de tal forma que cada cop que una dècima part de les permutacions totals s'hagin executat sense trobar un cromosoma amb un millor fitness, això vol dir que hem explotat de forma suficient aquell mínim local i que cal augmentar l'exploració. Per augmentar l'exploració incrementarem la variable mutationDegree, que és la mateixa que emprarem com a paràmetre de la funció Mutate. En cas que tornem a trobar un cromosoma amb millors resultats que el cromosoma pare, el grau de mutació tornarà a ser establert a 1 per tal de començar novament una explotació d'un mínim local de la funció.

La forma que es proposa permet que el nostre algorisme genètic tracti de forma molt eficient els problemes que solen aparèixer entre exploració i explotació en aquest tipus d'algorismes, i per tant això ens permetrà trobar més i millors solucions dintre de l'espai de solucions.

7.3. Algorisme genètic elitista.

Per tal de millorar els nostres resultats, s'ha decidit implementar una elit en el nostre algorisme genètic. Una elit no és més que un conjunt de cromosomes que representen les solucions candidates que han obtingut un millor fitness. En algunes ocasions s'empra aquesta elit per crear nous

cromosomes fills per creuament de membres de l'elit amb l'esperança que aquests disposaran d'un bon fitness.

En el nostre cas, definirem una elit de cromosomes els quals hauran de disposar de marges d'error inferiors a un percentatge determinat, de tal forma que un cop acabades totes les iteracions del nostre algorisme, disposem d'una elit amb els 10 cromosomes amb millors resultats.

Per dur a terme aquesta elit s'ha decidit crear un bucle on s'executarà tot el nostre algorisme genètic fins que no s'arribi a una elit de 10 membres tots ells amb un marge d'error inferior a un 30%. D'aquesta forma, un cop acabades les iteracions podrem emprar aquesta elit per cercar solucions més òptimes mitjançant els valors genètics dels cromosomes que formen part de l'elit.

A continuació podem observar com s'ha implementat aquest sistema elitista en el nostre algorisme genètic:

```
do
{
    Chromosome parent = new Chromosome();
    parent.GenerateParent();
    int parentFitness = parent.GetFitness(messages);

    for (int i = 0; i < numPermutations; i++)
    {
        mutationDegree = 1;
        if (i % (numPermutations / 10) == 0) {
            mutationDegree += 1;
        }
        Chromosome child = parent.Clone();
        child.Mutate(mutationDegree);
        int childFitness = child.GetFitness(messages);

        if (childFitness > parentFitness)
        {
            mutationDegree = 1;
            parent = child.Clone();
            parentFitness = childFitness;
        }
    }

    if (elite.Count() < 10)
    {
        elite.Add((parentFitness, parent));
    }
    else if (elite.Min(e => e.fitness) < parentFitness)
    {
        elite.Remove(elite.Where(e => e.fitness == elite.Min(el => el.fitness)).First());
        elite.Add((parentFitness, parent));
    }
} while (elite.Count < 10 || GetRelativeError(elite.Min(e => e.fitness), countMessages) > 30);

elite.ForEach(e => SaveResult(e.chromosome, e.fitness));
```

Figura 29. Modificació de l'algorisme genètic amb establiment d'una elit.

Tal com es pot observar, un cop finalitzada l'execució del nostre algorisme, aquest disposarà d'una elit de 10 cromosomes amb marges d'error inferiors al 30% (el marge d'error es pot comprovar mitjançant una

funció anomenada GetRelativeError la qual rep com a paràmetres el fitness d'un cromosoma i el nombre de missatges que conte).

L'elit obtinguda finalment es desarà a la nostra base de dades per poder-la consultar en tot moment i veure els resultats de cada cromosoma que ha sigut inclòs a l'elit.

7.4. Exploració de l'elit.

En tot algorisme genètic de tipus elitista es fa ús d'aquesta elit per tal de poder obtenir nous cromosomes derivats de les dades genètiques dels cromosomes que formen part d'aquesta elit. En el nostre cas, l'elit servirà per crear aquests nous cromosomes mitjançant les ponderacions de la nostra funció objectiu.

Quan s'ha desenvolupat l'algorisme genètic, s'han obviat les ponderacions dels paràmetres de tal forma que aquestes han sigut establertes de forma aleatòria dintre d'un rang que per defecte va de 0 a 100. En el moment en què volem que els nous cromosomes siguin creats a partir de la nostra elit, caldrà definir aquestes ponderacions i crear nous cromosomes dintre d'aquest rang.

Per tal de definir les ponderacions, s'ha creat una nova funció dintre de la classe del cromosoma anomenada SetPonderations que podem veure a continuació:

```
2 referencias
public void SetPonderations(List<Chromosome> chromosomes)
{
    foreach (VarType varType in VarTypes)
    {
        List<double> values = chromosomes.SelectMany(c => c.Genes).Where(g => g.VarType == varType).ToList().Select(ge => ge.Value).OrderBy(gv => gv).ToList();
        double avg = values.Average();
        double sd = Math.Sqrt(values.Average(v => Math.Pow(v - avg, 2)));
        double minValue = avg - sd;
        double maxValue = avg + sd;
        int ponderationIndex = Ponderations.IndexOf(Ponderations.First(p => p.varType == varType));
        Ponderations[ponderationIndex] = ((varType, minValue, maxValue));
    }
}
```

Figura 30. Implementació de la funció per establir les ponderacions de les operacions.

Tal i com es pot observar, aquesta funció estableix per cada gen un rang de ponderacions que anirà des d'una desviació estandard per sota de la mitjana del conjunt de ponderacions que han obtingut els cromosomes de l'elit per aquell determinat gen, fins a una desviació estandard per sobre de la mitjana de les ponderacions de l'elit per aquell gen.

D'aquesta forma, podrem explorar tot un espai de solucions que es trobara acotat en aquells valors que sabem que s'ha pogut trobar l'elit sense necessitat a dur a terme una explotació de mínims locals.

Un cop definides les ponderacions, tan sols haurem de fer un algorisme genètic que explori tot l'espai de solucions definit per les ponderacions calculades. A continuació es pot observar l'algorisme genètic que emprarem per dur a terme aquesta funcionalitat:

```
Chromosome parent = new Chromosome();
parent.SetPonderations(elite.Select(e => e.chromosome).ToList());
parent.GenerateParent();
int parentFitness = parent.GetFitness(messages);

for (int i = 0; i < 10000000; i++)
{
    Chromosome child = new Chromosome();
    child.SetPonderations(elite.Select(e => e.chromosome).ToList());
    child.GenerateParent();
    int childFitness = child.GetFitness(messages);

    if (childFitness > parentFitness)
    {
        parent = child.Clone();
        parentFitness = childFitness;
    }
}

SaveResult(parent, parentFitness);

Console.WriteLine("END OF PERMUTATIONS");
Console.WriteLine("\nPress any key to continue...");
Console.ReadKey();
```

Figura 31. Implementació d'un algorisme genètic auxiliar per explotar l'espai de solucions de l'elit.

Tal com podem observar, en aquest cas no farem ús de la funció de mutació per tal de crear nous cromosomes fills a partir de clons d'un cromosoma pare, sinó que cada nou fill serà generat de forma aleatòria de la mateixa forma que es genera un pare, simplement que les ponderacions es trobaran definides per l'elit obtinguda en l'algorisme genètic anterior.

És per aixo que parlem d'exploració de l'elit, ja que en aquest cas no es dona una explotació i no implementarem tampoc un sistema de grau de mutació ja que la mutació no es dona en cap moment.

Un cop executat un nombre suficientment gran de iteracions, que en el nostre cas hem decidit establir en 10 milions, desarem el millor resultat a la nostra base de dades per tal de poder-lo consultar posteriorment.

A partir d'aquest moment podem afirmar que tenim un algorisme genètic plenament funcional i es poden començar a analitzar les dades dels seus resultats.

8. Execució i anàlisi dels primers resultats

8.1. Error relatiu d'una llista ordenada.

Quan parlem d'error relatiu en una funció matemàtica, ens referim a aquell error que es comet respecte a l'error màxim que podem cometre, de tal forma que en una funció matemàtica encarregada d'ordenar un conjunt d'elements, l'error relatiu serà aquell resultant de comparar la posició de cada element amb la posició calculada pel nostre algorisme. El sumatori de diferències d'aquestes posicions (real i calculada), serà l'error absolut de la funció, el qual anomenarem fitness, i es pot veure com es calcula mitjançant la següent formula matemàtica:

$$Fitness = Err_{abs} = \sum_{i=0}^{n} |PR_i - PC_i|$$

On n es correspon amb el nombre d'elements, PR es correspon amb la posició real d'un element, i PC es correspon amb la posició calculada d'un element.

L'error relatiu l'obtindrem de la divisió entre l'error absolut o fitness i l'error màxim que es pot donar. Aquest error màxim, en el cas d'un conjunt ordenat d'elements, ve donat per la següent formula matemàtica:

$$Err_{max} = \frac{n^2}{2}$$

On n es correspon amb el nombre total d'elements que conté la nostra llista ordenada. Per tant, l'error relatiu que podem cometre es correspon amb la divisió entre l'error absolut o fitness i l'error màxim que es pot obtenir. Per tal d'una millor comprensió multiplicarem aquest resultat per 100 amb l'objectiu d'obtenir un percentatge.

Un cop tenim la forma de calcular l'error relatiu comès per un cromosoma sobre una llista d'elements, només cal implementar-lo en forma algorítmica tenint en compte que ja sabem el nombre d'elements que conté la llista i el resultat del fitness del nostre cromosoma. L'algorisme quedaria de la següent forma:

```
5 referencias
public double GetRelativeError(int fitness, int numItems)
{
    return Math.Abs((double)fitness / ((long)Math.Pow(numItems, 2) / 2)) * 100;
}
```

Figura 32. Implementació de la funció per l'error relatiu d'una llista ordenada.

Tal com podem observar, el valor resultant serà un nombre de punt flotant de doble precisió. Aquest valor ens permetrà establir la nostra elit per sota d'un determinat marge d'error així com ens permetrà una millor claredat en els nostres resultats per un posterior anàlisi.

8.2. Obtenció de notícies.

Tal i com s'ha explicat anteriorment, farem ús de les funcions de RSS implementades en el punt 5.1 i de les funcions de web scrapper implementades en el punt 5.2 per tal d'obtenir les dades de les notícies.

Per dur a terme l'execució, simplement haurem de fer ús de la nostra aplicació de consola tal com es mostra a continuació:

- En primer lloc seleccionarem el punt de menú News:

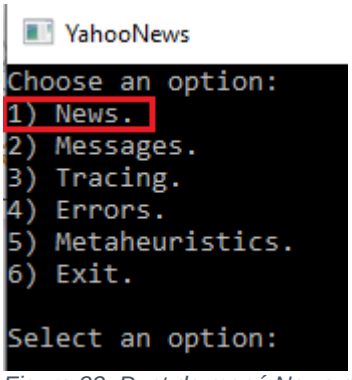

Figura 33. Punt de menú News de l'aplicació de consola.

- En segon lloc seleccionarem el punt de menú Pull News per tal d'obtenir les dades que es poden extreure mitjançant RSS:

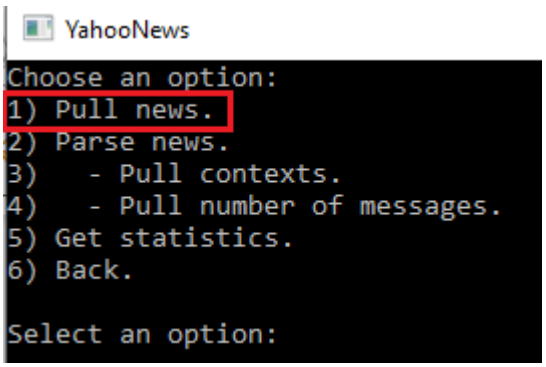

Figura 34. Punt de menú Pull News de l'aplicació de consola.

- Tot seguit podrem comprovar les notícies afegides a la nostra base de dades obtingudes per RSS:

Figura 35. Obtenció de les noticies a l'aplicació de consola.

- A continuació, seleccionarem el punt de menú Parse news, que s'encarregarà d'obtenir per web scrapper tant el nombre de missatges de cada notícia com el seu context, que serà necessari per obtenir posteriorment els comentaris.

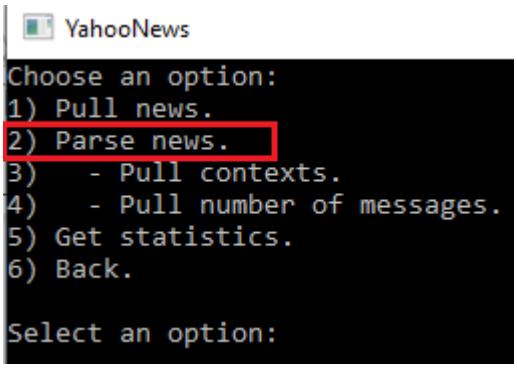

Figura 36. Punt de menú Parse news de l'aplicació de consola.

- Finalment podrem comprovar com s'afegeixen les dades de context i de nombre de missatges a la nostra base de dades:

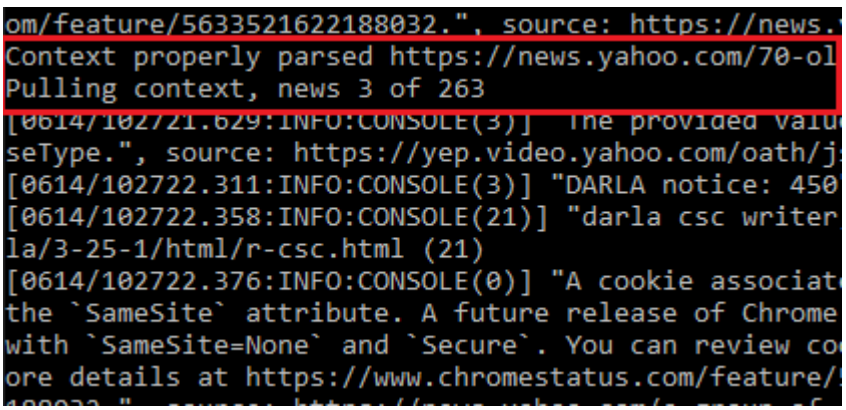

Figura 37. Obtenció de contexts a l'aplicació de consola.

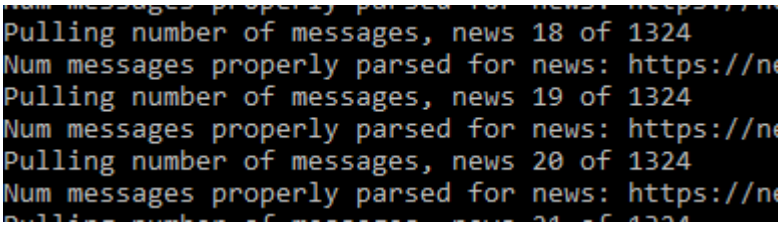

Figura 38. Obtenció de nombre de missatges a l'aplicació de consola.

Un cop finalitzats aquests passos, la nostra base de dades contindrà tot un conjunt de notícies amb les dades necessàries per poder obtenir el llistat de comentaris necessari pel nostre algorisme genètic.

8.3. Obtenció de comentaris.

Un cop disposem de les dades de les notícies a la nostra base de dades, només caldrà obtenir els comentaris a les notícies fent ús de les eines desenvolupades i que s'han implementat tal com s'explica en el punt 5.3.

Novament per tal d'executar aquestes funcions farem ús de la nostra aplicació de consola seguint els passos següents:

- En primer lloc seleccionarem el punt de menú Messages:

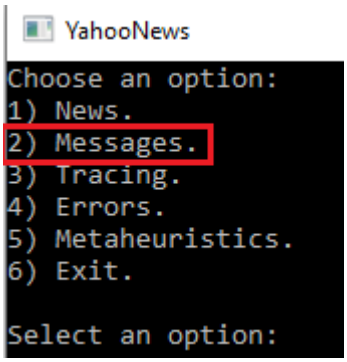

Figura 39. Punt de menú Messages de l'aplicació de consola.

- En segon lloc, seleccionarem el punt de menú Pull messages, que s'encarregarà d'obtenir totes les dades dels missatges.

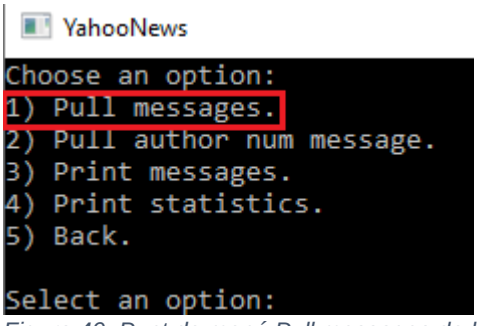

Figura 40. Punt de menú Pull messages de l'aplicació de consola.

- Finalment podrem comprovar com s'afegeixen les dades dels comentaris a la nostra base de dades:

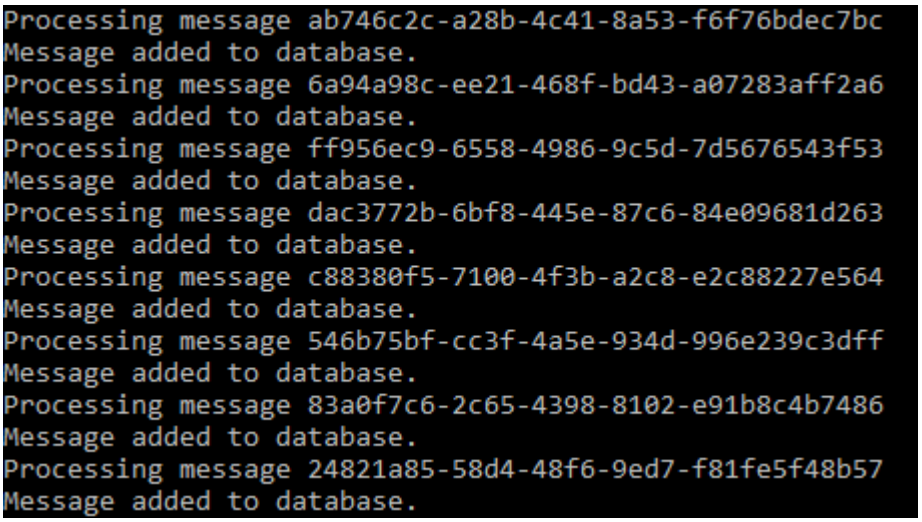

Figura 41. Obtenció de missatges a l'aplicació de consola.

8.4. Execució de l'algorisme genètic.

Un cop disposem d'un conjunt suficientment gran de comentaris d'una mateixa notícia a la nostra base de dades, ja podem executar el nostre algorisme genètic implementat en els punts 6 i 7 d'aquest document.

Per tal de facilitar-ne l'execució, l'algorisme genètic s'ha integrat a l'aplicació de consola que conté totes les funcions disponibles, i haurem de seguir els següents passos:

- En primer lloc, seleccionarem el punt de menú de Metaheuristics:

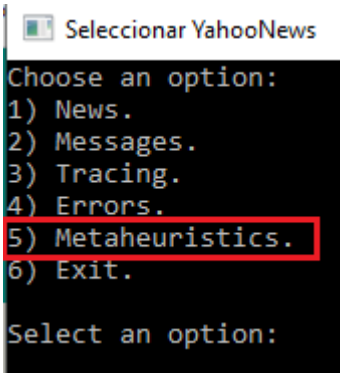

Figura 42. Punt de menú Metaheuristics de l'aplicació de consola.

- Tot seguit, seleccionarem el punt de menú anomenat Run genetic algorithm:

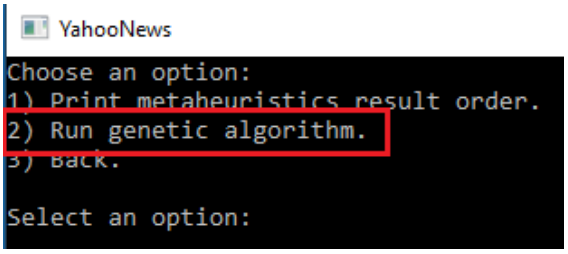

Figura 43. Punt de menú Run genetic algorithm de l'aplicació de consola.

- Ara ja només cal introduir el context de la notícia sobre la qual volem executar el nostre algorisme genètic:

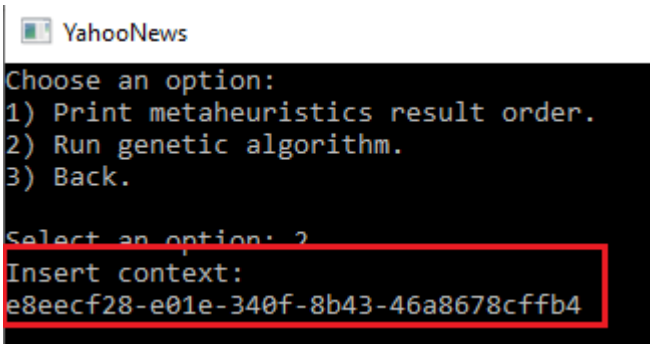

Figura 44. Introducció del context per l'algorisme genètic a l'aplicació de consola.

- Finalment podem comprovar l'execució del nostre algorisme genètic:

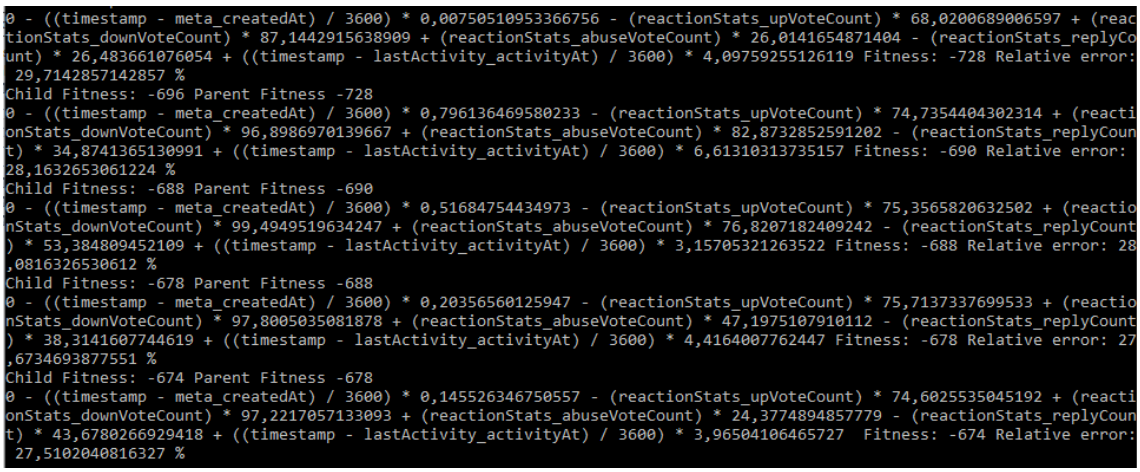

Figura 45. Execució de l'algorisme genètic a l'aplicació de consola.

8.5. Anàlisi de resultats.

Un cop executat el nostre algorisme genètic, podem comprovar que s'executa correctament, i podem comprovar que per una gran quantitat de notícies els resultats seran molt similars, podem observar patrons molt definits com que les reaccions negatives (vots negatius i vots abusius) compten més que les reaccions positives (vots positius i nombre de respostes), que la data de creació dels comentaris és molt poc significativa a l'hora d'establir un ordre determinat, i que la data de la darrera interacció no és altament significativa, però no obstant cal tenir-la present.

Tot i això, tot i que tenim una gran quantitat de resultats d'execucions del nostre algorisme genètic, i tot i que podem observar uns patrons de forma molt definida en totes les execucions sobre totes les notícies, també és cert que el marge d'error mínim al qual arribarem és al voltant d'un 25%. Per tant, això vol dir que tenim en un 75% d'encert la formula per aconseguir que els nostres comentaris adquireixin una gran visibilitat assolint bones posicions a la llista de comentaris retornada per Yahoo! News.

Tot i que els resultats no són dolents tenint en compte l'opacitat de l'algorisme de classificació de Yahoo News!, ens hem marcat com a objectiu aconseguir cotes per sobre del 80% d'encert, o el que és el mateix, un marge d'error per sota del 20%, de tal forma que seguint les pautes marcades pel resultat del nostre algorisme ens resulti relativament fàcil aconseguir bones posicions pels nostres comentaris en el resultat de la llista ordenada.

9. Presa de decisions i implementació de millores.

9.1. Inclusió de nous paràmetres.

El fet que els resultats obtinguts no siguin tan acurats com desitjaríem, i que mostrin un error superior al 20% desitjat, és indicatiu de que ens falten alguns paràmetres que Yahoo! News està considerant a l'hora d'establir l'ordre amb el qual genera la seva llista de comentaris. Aquests paràmetres poden ser molt diversos, i alguns d'ells, tals com anàlisi de sentiments, que ja s'ha explicat en el punt 3.4, no es poden disposar per part de l'usuari ni formen part de l'objectiu d'aquest treball. No obstant, altres paràmetres, tot i que són manco accessibles, si que es poden disposar en aquest treball. A continuació s'exposaran els paràmetres que s'han decidit implementar.

9.2. Nombre de comentaris d'un usuari.

Un paràmetre que sembla que seria lògic tenir en compte a l'hora de ponderar un comentari, és el nombre de comentaris que ha fet anteriorment un usuari abans d'haver escrit el comentari.

Per tal de poder implementar aquesta millora, cal que desenvolupem un nou algorisme a través de la nostra eina de Web Scrapper, que consulti mitjançant l'API pròpia de Yahoo! News el nombre de comentaris d'un usuari.

Tot seguit s'adjunta una part del codi que s'ha emprat per tal d'extreure la informació del nombre de comentaris d'un mateix usuari i desar-la a la nostra base de dades:

```
Console.WriteLine("Pulling number of messages, author {0} of {1}", count, messagesNumber);

try
{
    driver.Url = string.Format(urlAuthorMessages, message.meta_author_guid);
    IWebElement jsonContainer = driver.FindElement(By.XPath("/html/body/pre"));
    dynamic jsonResponse = JsonConvert.DeserializeObject(jsonContainer.GetAttribute("innerText"));
    int jsonNumMessages = jsonResponse.data.total.count;
    message.meta_author_numMessages = jsonNumMessages;
    yahoodb.SaveChanges();
    Console.WriteLine("Num messages properly parsed for autor: {0}", message.meta_author_nickname);
}
```

Figura 46. Implementació de la funció per extreure el nombre de comentaris d'un usuari.

Tal com es pot veure, aquesta informació és recuperada gràcies a l'API de Yahoo!, que retorna els resultats en forma de JSON. Un cop transformat el resultat en JSON en forma d'objectes, podem obtenir fàcilment aquesta informació i desar-la a la nostra base de dades en un nou camp que crearem anomenat meta_author_numMessages.

9.3. Distribució temporal de les interaccions.

Un dels majors problemes que ens trobem a l'hora d'obtenir informació, és el fet que la informació obtinguda es correspon exclusivament al moment concret en el que és obtinguda. Tota la informació sobre el nombre d'interaccions en un comentari (positives, negatives, abusives i nombre de respostes) són únicament les que es poden observar en el moment en el qual s'han obtingut.

Aquest fet provoca que no es pugui tenir en compte com han estat distribuïdes aquestes interaccions en el temps, no podem saber si s'han distribuït de forma uniforme, si han sigut totes en els primers instants del comentari, si han sigut en les darreres hores, etc. I això provoca que el nostre algorisme no sigui suficientment precís per no disposar d'aquesta informació.

Per tal de poder extreure aquesta informació, no cal que desenvolupem nous mètodes per tal d'obtenir informació, sinó que únicament cal que els nostres algorismes d'extreure informació es puguin executar de forma cíclica recollint totes aquestes variables en diferents instants de temps.

Per tal de dur a terme aquesta millora, en primer lloc modificarem la nostra base de dades, afegint camps de tipus "upVoteCount_stepX", on la X es correspon amb el pas en el que s'ha recollit aquesta informació. D'aquesta forma, cada un dels steps o passos recollirà el nombre de comentaris que hi ha hagut des de l'step anterior. I així serà més fàcil poder crear una distribució en el temps.

En aquest cas, s'ha decidit crear una nova classe Tracing en el nostre projecte, que contindrà algorismes molt similars a la forma en la que obteníem els comentaris, però també contindrà les funcionalitats necessàries per tal de poder recórrer aquests comentaris durant el temps.

Bàsicament es tracta de crear una funció que anomenarem TraceMessages, i que donat un identificador de la notícia recorrerà una quantitat determinada de vegades la llista de comentaris amb una diferència de temps entre cada iteració determinada. A continuació es pot veure la capçalera d'aquesta funció així com la part on es defineix el temps d'espera entre cada iteració:

```
2 referencias
public static void TraceMessages(string guid, int currentStep = 0)
{
    YahooNews.yahooEntities yahoodb = new YahooNews.yahooEntities();
    yahooEntities yahoodbErr = new YahooNews.yahooEntities();
    news newsItem = yahoodb.news.First(n => n.guid == guid);
    for (int step = currentStep; step < 144; step++)
    {
```

Figura 47. Definició de la funció per traçar els missatges en el temps.

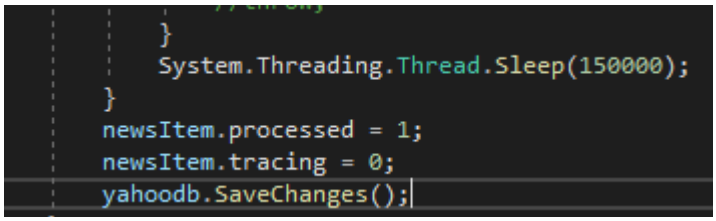

Figura 48. Temps d'espera entre cada cicle d'execució i finalització de la traçabilitat.

9.4. Anàlisi de la distribució temporal de les interaccions.

Tal i com s'ha pogut observar en el punt anterior, la traçabilitat dels comentaris d'un algorisme inclourà un total de 144 iteracions separades entre si per 150 segons, això implica una traçabilitat total de 6 hores sobre una mateixa notícia recollint les interaccions sobre els comentaris cada 2,5 minuts.

El fet que disposem d'aquesta informació, per si sola no implica que puguem tenir en compte totes i cadascuna de les iteracions de la nostra funció de traçabilitat, ja que si féssim un desenvolupament d'aquest tipus, el resultat obtingut en forma de funció objectiu seria excessivament especifica per aquella notícia en concret, i per tant, tot i obtenir un marge d'error relativament baix, aquesta funció objectiu seria poc extrapolable a altres notícies, donant marges d'error molt més elevats quan s'extrapolés la funció entre notícies.

Per tal de donar solució al problema, cal que fem un anàlisi sobre els resultats de la traçabilitat per tal de poder decidir quina serà la millor forma d'implementar aquesta sobre el nostre algorisme genètic.

Per tal de fer aquest anàlisi s'ha decidit implementar una funció anomenada ExportTracing que es troba a la pròpia classe de Tracing i serà accessible des de l'aplicació de consola. Aquesta funció recollirà totes les dades necessàries a la base de dades per tal de crear un conjunt de fitxers CSV que podrem tractar fàcilment amb Matlab.

Un cop exportats els fitxers, només caldrà desenvolupar un Script en Matlab per tal de poder observar la distribució temporal de les interaccions de determinats comentaris en Matlab. A continuació es pot observar un exemple d'aquesta distribució en el temps per determinats comentaris.

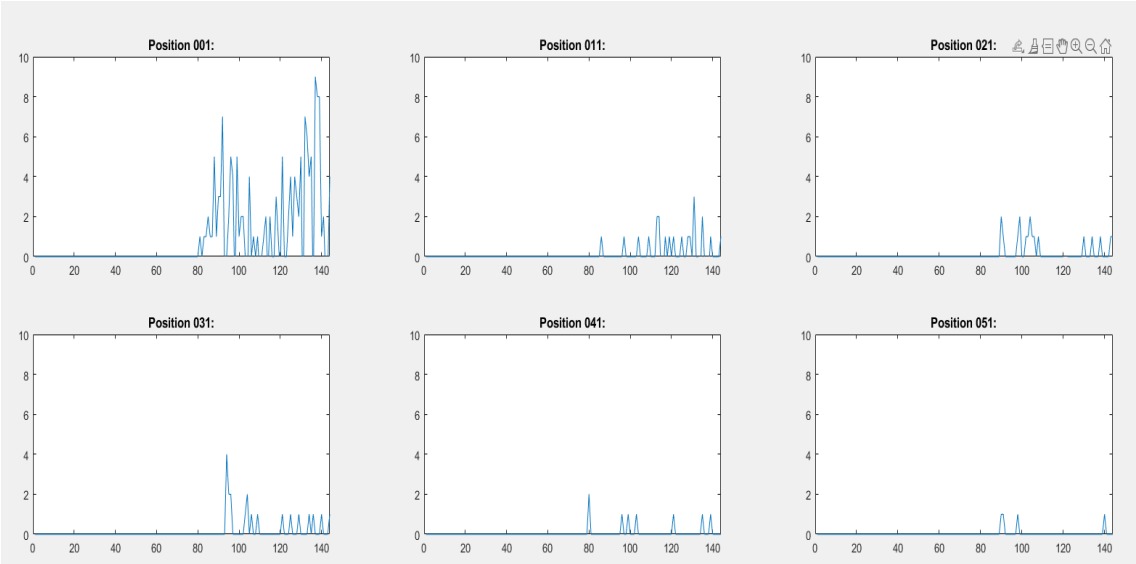

Figura 49. Distribució temporal de les interaccions a diferents comentaris.

Tal i com es pot veure en la gràfica, els comentaris que es troben en les primeres posicions, no solament tenen un major nombre de comentaris, sinó que tots ells presenten determinats patrons on es pot veure molt clarament que donen una gran importància a les interaccions que s'han fet en els darrers instants, i concretament en la darrera hora.

9.5. Inclusió de paràmetres de les darreres interaccions.

Un cop s'ha pogut comprovar que la distribució temporal és molt important a l'hora d'establir l'ordre dels comentaris i que les interaccions a la darrera hora són molt significatives, només cal que implementem aquesta casuística en el nostre algorisme genètic.

Per tal de dur a terme aquesta millora, en primer lloc afegirem aquestes dades com a paràmetres dintre de la classe VarType, que és la que tindrà en compte el nostre algorisme a l'hora de crear els seus gens, tal com es mostra a continuació:

```
46 referencias
public enum VarType
{
    meta_author_numMessages,
    meta_createdAt,
    reactionStats_upVoteCount,
    reactionStats_downVoteCount,
    reactionStats_abuseVoteCount,
    reactionStats_replyCount,
    lastActivity_activityAt,

    reactionStats_upVoteCount_last,
    reactionStats_downVoteCount_last,
    reactionStats_abuseVoteCount_last,
    reactionStats_replyCount_last,

}
```

Figura 50. Modificació de VarType per incloure nous parametres en un gen.

Dintre de la nostra classe Computation, on es duien a terme les operacions necessàries per calcular la ponderació de cada cromosoma, caldrà que quan els gens del cromosoma siguin de tipus xxx_last, ens faci un sumatori de les interaccions de l'última hora (les corresponents a les iteracions compreses entre la step120 i la step 144). Per tant caldrà crear una funció que simplement ens retorni la suma d'aquestes interaccions. A continuació es pot veure una part de la funció:

```
4 referencias
public static double GetLastSums(string type, message message)
{
    switch (type)
    {
        case "up":
            return (message.reactionStats_upVoteCount_step120 ?? 0) +
                        (message.reactionStats_upVoteCount_step121 ?? 0) +
                        (message.reactionStats_upVoteCount_step122 ?? 0) +
                        (message.reactionStats_upVoteCount_step123 ?? 0) +
                        (message.reactionStats_upVoteCount_step124 ?? 0) +
                        (message.reactionStats_upVoteCount_step125 ?? 0) +
                        (message.reactionStats_upVoteCount_step126 ?? 0) +
                        (message.reactionStats_upVoteCount_step127 ?? 0) +
                        (message.reactionStats_upVoteCount_step128 ?? 0) +
                        (message.reactionStats_upVoteCount_step129 ?? 0) +
                        (message.reactionStats_upVoteCount_step130 ?? 0) +
                        (message.reactionStats_upVoteCount_step131 ?? 0) +
```

Figura 51. Implementació de la funció per obtenir les interaccions a un comentari la darrera hora.

D'aquesta forma, quan la funció Compute arribi a una de les variables que hem creat per obtenir les darreres interaccions, serà capaç de retornar un valor numèric que resulti del sumatori de totes aquestes interaccions compreses en la darrera hora.

Un cop implementades aquestes millores respecte a l'algorisme genètic, ja podem tornar a executar el nostre algorisme genètic amb l'objectiu de comprovar els resultats de la inclusió d'aquests paràmetres.

10. Execució i anàlisi final dels resultats.

10.1. Obtenció de dades amb traçabilitat.

Tal com ja s'ha fet en el punt 8 d'aquest document, per tal de poder executar el nostre algorisme genètic, primer cal obtenir dades de la forma més adient amb l'objectiu d'obtenir els millors resultats. Per això el primer que farem serà recollir les dades d'una notícia amb la funció de la traçabilitat implementada en el punt 9 d'aquest document. Per dur-ho a terme caldrà executar les següents passes:

- En primer lloc, seleccionarem el punt de menú Tracing en el menú inicial de la nostra aplicació de consola.

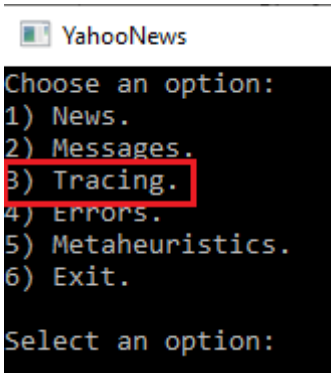

Figura 52. Punt de menú Tracing de l'aplicació de consola.

- A continuació seleccionarem el punt de menú Trace news.

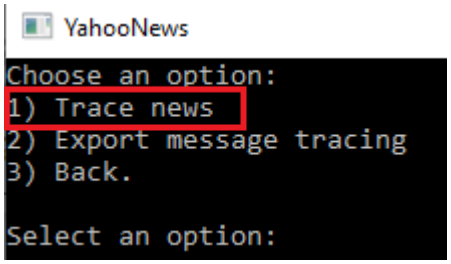

Figura 53. Punt de menú Trace news de l'aplicació de consola.

- Tot seguit haurem d'indicar la URL de la notícia sobre la qual volem realitzar un seguiment.

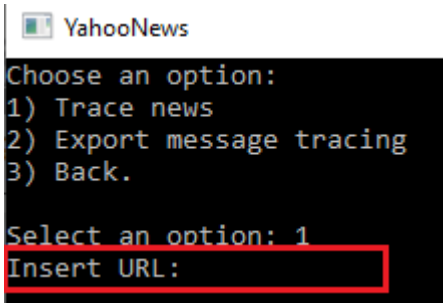

Figura 54. Inserció de la URL d'una noticia a traçar.

- Finalment podrem observar com s'obtenen les dades de la notícia i els seus comentaris de forma iterativa amb l'objectiu d'obtenir-ne la traçabilitat.

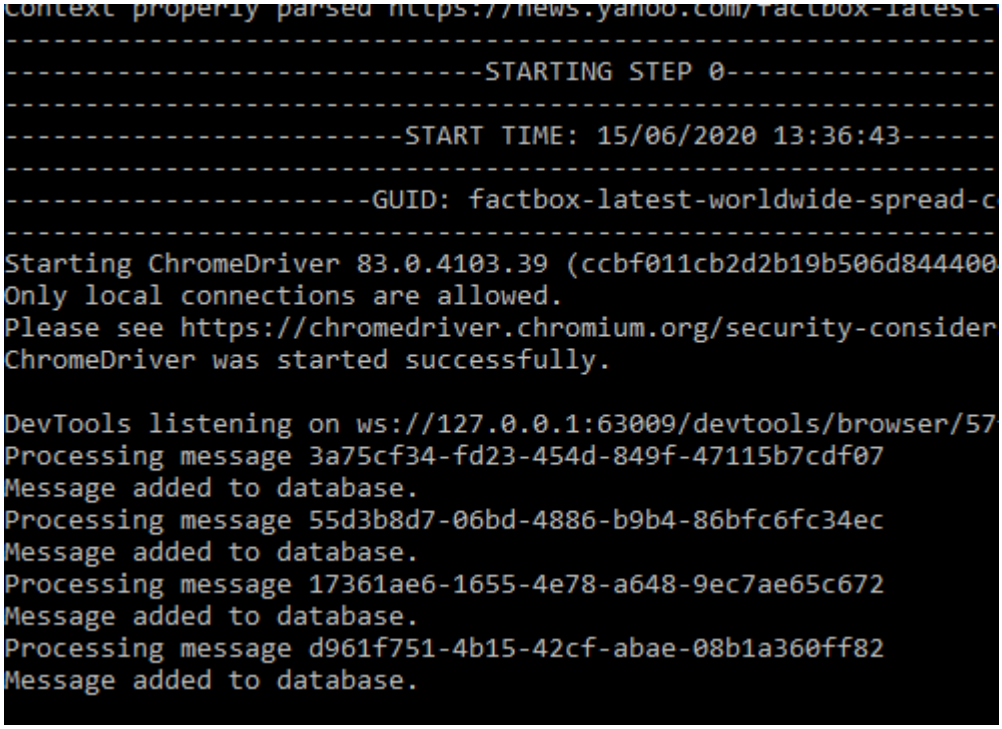

Figura 55. Execució de la traçabilitat a l'aplicació de consola.

10.2. Execució de l'algorisme genètic amb traçabilitat.

L'execució de l'algorisme genètic es durà a terme de la mateixa forma que s'ha fet en el punt 8.4 d'aquest document, i per tant no en detallarem novament les passes a seguir, simplement podrem comprovar que el nostre algorisme genètic inclou ara si, els nous paràmetres de traçabilitat i el nombre de comentaris d'un mateix usuari. A continuació es mostren els resultats de l'execució del nostre algorisme genètic.

Figura 56. Resultats de l'algorisme genètic.

10.3. Anàlisi de resultats

Tal i com es pot observar en els resultats, després de moltes iteracions de l'algorisme genètic, l'error relatiu obtingut és del voltant del 20%, podent arribar inclus al 12% en alguns casos, i els paràmetres seleccionats permeten obtenir una funció objectiu fàcilment extrapolable a altres notícies sense que l'error relatiu augmenti de forma dràstica.

A continuació es passen a explicar els paràmetres segons la importància que tenen dintre de la nostra funció objectiu.

- **meta_author_numMessages**: Tot i que aquest paràmetre ha sigut inclòs com una millora, es tracta d'un valor gairebé insignificant, les seves ponderacions sempre són molt properes al 0, i per tant podem concloure amb total seguretat que Yahoo! News no dona cap importància a la quantitat de missatges que pugui haver escrit un usuari per tal de ponderar els seus comentaris.

- **meta_createdAt**: Aquest paràmetre acostuma a prendre valors de ponderació entre el 60 i el 90 en una escala de 0 a 100, i per tant podem concloure que és molt important el temps d'antiguitat d'un comentari, el fet que les ponderacions siguin positives implica que com més temps d'antiguitat tingui un missatge, pitjors seran els seus resultats.

- **reactionStats_upVoteCount**: Amb una ponderació que oscil·la entre el 2 i el 16 en una escala de 0 a 100, podem concloure que el nombre total de vots positius a un comentari no és altament significatiu, tot i que és important tenir en compte aquest paràmetre, doncs a més vots positius millors resultats obtindrem.

- **reactionStats_downVoteCount**: En aquest cas les ponderacions oscil·len entre el 15 i el 30 en una escala de 0 a 100, i per tant podem concloure que és un paràmetre significatiu. El fet de trobar-se sempre amb signe positiu indica que com més vots negatius rebi un comentari, pitjor serà la seva classificació.

- **reactionStats_abuseVoteCount**: Aquest paràmetre presenta valors molt aleatoris, en algunes execucions pren valors propers a 0 i altres pren valor molt elevats de fins a 93. Aquest fet és simptomàtic que les reaccions abusives són molt escasses en els nostres comentaris, i tot i que poden ser significatives en la classificació de comentaris, no disposem d'informació suficient com per extreure una conclusió clara.

- **reactionStats_replyCount**: Aquest paràmetre representa el nombre de respostes totals que ha rebut un comentari. Les seves ponderacions van de 0 a 8, i per tant podem concloure que com a sumatori total són poc significatives.

- **lastActivity_activityAt**: El paràmetre de temps de la darrera interacció, també presenta valors anòmals molt dispersos, podent anar de 0 fins a 40, això és a causa que els valors que presenten els comentaris en aquest paràmetre no són gaire significatius, tot i que cal tenir en compte tal i com passava en el temps de creació del comentari, que com més temps hagi passat fins el moment actual pitjor seran els resultats.

- **reactionStats_upVoteCount_last**: Es tracta dels vots positius en la darrera hora. Aquest paràmetre es troba molt ben definit en totes les execucions del nostre comentari i presenta valors entre el 90 i el 100. Aquest fet indica de forma clara que aquest paràmetre és molt important a l'hora de establir l'ordre dels comentaris. El fet de trobar-se de forma negativa indica que com més vots positius ha rebut un comentari en l'última hora, més probabilitats hi ha de que es trobi entre les primeres posicions de la llista.

- **reactionStats_downVoteCount_last**: Els vots negatius de la darrera hora oscil·len entre el 0 i el 7 en una escala de 0 a 100. Aquest fet indica que tot i que cal que siguin tinguts en compte, no són gaire significatius en la classificació d'un comentari.

- **reactionStats_abuseVoteCount_last**: Tot i que es poden trobar alguns valors anòmals, la majoria de les execucions indiquen que el nombre de vots abusius a un comentari duran l'última hora és un paràmetre molt significatiu a l'hora d'ordenar els comentaris de la notícia. Amb la majoria dels seus valors propers a 100, podem concloure que com més vots abusius presenti un comentari durant l'última hora pitjors seran els seus resultats en la classificació.

- **reactionStats_replyCount_last**: El nombre de respostes que ha rebut un comentari durant l'última hora presenta unes ponderacions molt variades, que oscil·len entre el 30 i el 80 en algunes noticies, no obstant es troben en valors molt propers a 100 en d'altres. Per tant és difícil extreure conclusions sòlides d'aquest comentari, però podem dir que acostuma a tenir valors significatius que cal tenir presents i que pot influir molt en el resultat de la classificació. El fet de presentar-se en signe negatiu indica que com més respostes rep un comentari millor serà la seva posició en la classificació final de comentaris.

Podem concloure doncs, que el més important a l'hora d'obtenir una bona posició en la classificació final de comentaris retornats per Yahoo! News, és el fet d'haver obtingut un gran nombre de vots positius en la darrera hora, també és molt important que no hi hagi vots abusius en la darrera hora, i tenint en compte que el nombre de respostes obtingudes en la darrera hora i l'antiguitat del comentari també són paràmetres significatius.

Tot i que és recomanable tenir el mínim nombre de vots negatius, aquests no són prou significatius sempre i quan tinguem els demés paràmetres amb els valors apropiats. També podem concloure que el nombre de

comentaris que hagi fet un usuari en el passat és completament irrellevant en la posició que obtindran els seus comentaris.

Així doncs, per aconseguir un bon posicionament, podem afirmar que cal aconseguir els resultats desitjats principalment en aquests 4 paràmetres:

- **Vots positius a la darrera hora.** El paràmetre més important de tots, com més vots positius tingui un comentari a la darrera hora, millor posicionament obtindrà.

- **Respostes a la darrera hora.** Tot i que molt variat, força significatiu per ser tingut en compte, com més respostes tingui un comentari a la darrera hora, millor posicionament obtindrà.

- **Vots abusius a la darrera hora.** Yahoo! News tindrà en compte si un usuari esta votant de forma fraudulenta, com més vots abusius tingui un comentari a la darrera hora, pitjor posicionament obtindrà.

- **Antiguitat.** És important que els comentaris no tinguin una antiguitat excessiva, tot i que aquest paràmetre es menys important que els anteriors esmentats, com mes antic sigui un comentari mes difícil serà que obtingui un bon posicionament.

Els demes paràmetres que hi pot haver en un comentari, tot i que siguin mes o menys significatius, no son prou significatius per ser tinguts en compte a l'hora de voler obtenir un bon posicionament. Així doncs, si seguim les anteriors regles, existirà una probabilitat realment elevada d'aconseguir un bon posicionament en els comentaris que hagem deixat a les noticies.

11. Conclusions

11.1. Continguts treballats.

Al llarg d'aquest treball de fi de màster s'han treballat molts i diversos continguts del màster en enginyeria computacional i matemàtica, tals com l'optimització meta-heurística, els algorismes genètics, l'obtenció i les estructures de dades, etc.

Tots els continguts detallats anteriorment han sigut necessaris per tal de poder aconseguir un producte amb la qualitat suficient per tal d'assolir els nostres objectius

11.2. Assoliment dels objectius.

Els objectius inicials del present treball, tenien per objectiu aconseguir una aproximació a la funció emprada per Yahoo! News per classificar els comentaris de les seves notícies. Tot i que es tracta d'una funcionalitat molt opaca, i que la majoria de paràmetres emprats per Yahoo! no són accessibles a nivell d'usuari, s'han pogut extreure una gran quantitat de dades de cada comentari, s'han pogut obtenir més de 260.000 comentaris corresponents a més de 1.500 notícies diferents.

Amb totes aquestes dades, s'ha pogut executar amb èxit un algorisme genètic amb les funcionalitats suficients com per crear una regressió simbòlica sobre les dades obtingudes i ha permès així arribar a una funció objectiu que té un marge d'error de menys del 20%, en alguns casos inclús inferior al 13%.

Aquesta funció objectiu és prou genèrica com per poder ser extrapolada a altres notícies, i això permet a persones i entitats disposar de la informació suficient com per poder dur a terme les mesures necessàries amb l'objectiu que els seus comentaris disposin de la major visibilitat possible, podent així arribar al nombre més elevat de lectors possible.

Tot i que es considera que l'objectiu final ha sigut assolit de forma satisfactòria, el fet que no puguem disposar de molts dels paràmetres dels quals fa ús Yahoo! News, implica que no es puguin assolir marges d'error per sota del 12%, i per tant el fet de prendre les mesures que es poden identificar de la funció objectiu no garanteix l'èxit total dels resultats desitjats. No obstant, seguint les indicacions extretes d'aquesta informació tindrem unes possibilitats molt elevades d'aconseguir els nostres objectius.

11.3. Seguiment de la planificació.

Per tal de dur a terme aquest treball de fi de màster, s'ha seguit la planificació indicada en el punt 1.4 d'aquest document, segons el qual el treball s'ha dividit en les fases de: planificació, anàlisi, disseny, desenvolupament, proves i millora.

Aquestes fases són fàcilment identificables durant el desenvolupament d'aquest treball, de tal forma que cada capítol es correspon amb una de les fases descrites. Tots ells s'han dut a terme de forma cronològica i com era d'esperar s'han hagut de realitzar canvis en una fase de millora per tal d'obtenir millors resultats ja que l'objectiu principal del treball era el d'aconseguir cotes d'error relatiu inferiors al 20%.

11.4. Línies de treball futur:

Durant aquest treball, s'ha pogut confirmar l'enorme dificultat d'aconseguir una aproximació als algorismes de classificació opacs mitjançant tècniques meta-heurístiques d'algorismes genètics. Un dels principals problemes que s'han trobat durant aquest treball ha sigut la falta de dades que ens faciliti un resultat més òptim.

Una d'aquestes dades que no disposa aquest treball és l'anàlisi de sentiments de cada comentari que Yahoo! News fa i que probablement empra per la seva classificació.

Per tal de millorar el nostre algorisme genètic, seria necessari disposar d'aquestes dades, i per tant la línia de treball futur més clara i més necessària per tal d'obtenir aquests resultats, és el desenvolupament i la integració d'algorismes d'aprenentatge computacional capaços de realitzar un anàlisi de sentiments òptim de tal forma que els seus resultats s'assemblin el més possible als resultats de Yahoo! News.

Aquesta línia de treball permetrà millorar el nostre algorisme genètic i aproximar encara més l'algorisme de classificació emprat per Yahoo! News.

13. Glossari

MVC. *Model-vista-controlador:* és un patró d'arquitectura de programari, que separa les dades i principalment el que és la lògica de negoci d'una aplicació de la seva representació i el mòdul encarregat de gestionar els esdeveniments i les comunicacions.

API. *Application programming interface:* és un conjunt de subrutines, funcions i procediments (o mètodes, en la programació orientada a objectes) que ofereix certa biblioteca per ser utilitzat per un altre programari com una capa d'abstracció.

Web Scrapper. És una tècnica utilitzada mitjançant aplicacions de programari per extreure informació de llocs web.

EF. *Entity framework:* és un conjunt de tecnologies de ADO.NET que admeten el desenvolupament d'aplicacions de programari orientades a dades. Permet als desenvolupadors treballar amb dades en forma d'objectes, sense haver de preocupar de les taules i columnes de bases de dades en les que s'emmagatzemen.

C#. És un llenguatge de programació multiparadigma desenvolupat i estandarditzat per Microsoft com a part de la seva plataforma .NET.

LINQ. *Language Integrated Query:* és un component de la plataforma Microsoft .NET que afegeix capacitats de consulta a dades de manera nativa als llenguatges .NET.

JSON. *JavaScript Object Notation:* és un format de text senzill per a l'intercanvi de dades. Es tracta d'un subconjunt de la notació literal d'objectes de JavaScript.

JavaScript. És un llenguatge de programació interpretat, dialecte de l'estàndard ECMAScript. Es defineix com orientat a objectes, basat en prototips, imperatiu, dèbilment tipat i dinàmic.

14. Bibliografia

- DuBois, P. (2014). *MySQL cookbook*. O'Reilly.
- Galloway, J., Wilson, B., Allen, K., Matson, D., & Hanselman, S. *Professional ASP.NET MVC 5*.
- Liberty, J. (2005). *Programming C#*. O'Reilly.
- Luke, S. *Essentials of metaheuristics*.
- Mueller, J. (2013). *Microsoft® ADO. NET Entity Framework*. Microsoft Press.
- Russell, M. (2014). *Mining the social web*. O'Reilly.
- Benjamin, C. (2018). Web scraping con C#. Recuperat de: https://aspnetcoremaster.com/web/scraping/c%23/advientocsharp/dotnet/selenium/2018/12/24/web-scraping-con-csharp.html en data 10 de Desembre del 2019.
- Marley, T. (2019). Symbolic Regression From Scratch in C#. Resuperat de: https://medium.com/@taran_90075/symbolic-regression-from-scratch-in-c-part-1-f374771862f6 en data 12 de Desembre de 2019.

15. Annexos

15.1. Annex 1. Diagrama de flux d'un algorisme genètic simple.