

**Escola Universitària d'Enginyeria Tècnica
de Telecomunicació La Salle**

Treball Final de Grau

Grau en Enginyeria Informàtica

**Trolls Detector:
Herramienta para la detección
de trolls en el ámbito de los deportes**

Alumne

Professor Ponent

Jorge José Melguizo Torres

Elisabet Golobardes Ribé

**Escola Universitària d'Enginyeria Tècnica
de Telecomunicació La Salle**

Treball Final de Grau

Grau en Enginyeria Informàtica

**Trolls Detector:
Herramienta para la detección
de trolls en el ámbito de los deportes**

Alumne

Jorge José Melguizo Torres

Professor Ponent

Elisabet Golobardes Ribé

ACTA DE L'EXAMEN DEL TREBALL FI DE CARRERA

Reunit el Tribunal qualificador en el dia de la data, l'alumne

D. Jorge José Melguizo Torres

va exposar el seu Treball de Fi de Carrera, el qual va tractar sobre el tema següent:

Detección de tweets fraudulentos en el ámbito de los deportes

Acabada l'exposició i contestades per part de l'alumne les objeccions formulades pels Srs. membres del tribunal, aquest valorà l'esmentat Treball amb la qualificació de

Barcelona,

VOCAL DEL TRIBUNAL

VOCAL DEL TRIBUNAL

PRESIDENT DEL TRIBUNAL

laSalle

UNIVERSITAT RAMON LLULL

TFG

**Trolls Detector: Herramienta para la
detección de *trolls* en el ámbito de los
deportes**

Jorge Melguizo

Tutora: Elisabet Golobardes

DEPARTAMENTO DE INGENIERÍA INFORMÁTICA

Curso académico 2017/18

Palabras clave

Data Science, Twitter, Trust Tweets, Text Classification, Sentiment Analysis, Fake news, Trolls.

Abstract

La población cada vez es más exigente, en el mundo del periodismo esto se resume en inmediatez a la hora de publicar una noticia. Para ello los periodistas recurren cada vez más a las redes sociales como fuente de información y con la exigencia de ser los primeros en publicar los sucesos, no siempre contrastan la información con fuentes fidedignas. Si lo combinamos con que gran parte de la información que está en las redes sociales suele ser falsa hace que un periodista publique una noticia falsa y provocó que la sociedad se crea una mentira por el hecho de haberlo visto en televisión o leído en algún medio de información.

Este proyecto analizará los perfiles del usuario cuyo tweet se estudiando su veracidad y para ello se investigará sobre qué temas escribe, de qué forma los escribe, a quien hace menciones y retweets, la información de su perfil como si su cuenta es verificada, si siempre publica lo mismo y se comparará con los últimos tweets publicados en Twitter sobre esa temática para dar una aproximación de la veracidad del tweet a tratar.

Por lo tanto se pretende dar caza a mentiras en las redes sociales (*liar troll*) pero acortándolo al mundo de los deportes, también detectar aquellos usuarios *bots* o *observer troll* que no suben ninguna publicación y solo tienen twitter para espionar o ampliar los seguidores de otros perfiles y, por último, aquellos usuarios que siempre están furiosos y se podrían clasificar como *hater troll*. Si se detecta alguno de estos casos se le comunicará al usuario y se le aconsejará dejarle de seguir.

Se trata de un problema de Big Data, a tiempo real y dinámico. Es un proyecto de análisis del lenguaje natural con todo lo que ello conlleva.

*A mis padres Herminia y Jorge y a mis abuelos Herminia y Benedicto por estar
siempre a mi lado.*

*A Elisabet Golobardes por haberme dado esta oportunidad y haberme escuchado
siempre.*

Índice

1. Introducción	13
1.1. Contexto	13
1.2. Motivaciones	16
1.3. Objetivos Trabajo Final de Grado	17
2. Trabajo Relacionado	19
3. Trolls Detector	21
3.1. Tecnologías utilizadas en todas las fases	23
3.1.1. MongoDB	24
3.1.2. Pymongo 3.7.1	25
3.1.3. AVL Tree	25
3.2. Fase 0 - Extracción de la información y creación de los datasets	27
3.2.1. Concepto	27
3.2.2. Software	28
3.3. Fase 1 - Clasificación de texto según temática	32
3.3.1. Concepto	32
3.3.2. Software	33
3.4. Fase 2 - Análisis del sentimiento	41
3.4.1. Concepto	41
3.4.2. Software	42
3.5. Fase 3 - Resultado del análisis	45
4. Experimentación	47
4.1. Problemática lenguaje natural	47
4.1.1. Homonimia	47
4.1.2. Polisemia	48
4.1.3. Anáfora y elipses	49
4.1.4. Sintaxis no normalizada	49
4.1.5. Contexto	50
4.1.6. Otras circunstancias	50
4.2. Fase 0 - Extracción de la información	51
4.3. Fase 1 - Clasificación de texto según temática	58
4.4. Fase 2 - Análisis del sentimiento	64

4.5. Fase 3 - Resultado del análisis	67
5. Costes del proyecto	76
5.1. Costes temporales	76
5.2. Costes económicos	76
6. Conclusiones y líneas de futuro	77
6.1. Conclusiones	77
6.2. Líneas de futuro	78
7. Bibliografía	80
8. Anexos	84
8.1. Manual MongoDB	84
8.2. Manual Tweepy	86
8.3. Ontología de los deportes	88
8.4. Diccionarios	89
8.4.1. Palabras clave	89
8.4.2. Palabras secundarias	93
8.4.3. Palabras excluyentes	98
8.4.4. Palabras vacías	100
8.4.5. Palabras en análisis del sentimiento	101

Índice de figuras

1.	Esquema general del proyecto	22
2.	Clase DB utilizada para la comunicación entre el programa y la base de datos en MongoDB	25
3.	Ejemplo AVL Tree	26
4.	Diagrama de la clase propia AVL-TREE utilizada para la interacción con el árbol	26
5.	Estructura de extracción de tweets y interacción con diccionario	28
6.	Tablas en la base de datos del proyecto	29
7.	Conectar con la Base de Datos	30
8.	Obtener el diccionario de nuestra BBDD	30
9.	Insertar en la BBDD	30
10.	Diagrama de la clase propia Twitter encargada de la comunicación con twitter mediante la librería tweepy	31
11.	Diagrama fase 1 clasificación de texto según temática	33
12.	Código ejemplo utilización VADER	34
13.	Output ejemplo utilización VADER	35
14.	Formula Naive Bayes	35
15.	Formula Multinomial Bayes	35
16.	Ejemplo SVM [28] con diferentes kernels.	37
17.	Ejemplo SVM [30]	39
18.	Diagrama fase 2 análisis del sentimiento	42
19.	Código ejemplo utilización VADER	43
20.	Output ejemplo utilización VADER	43
21.	Función que devuelve el id o el nombre del tipo en esa posición.	52
22.	Base de datos del proyecto.	53
23.	Diccionario de los deportes guardados obtenidos por Wikipedia.	54
24.	Diccionario de los deportes propio.	54
25.	Diccionario de los sentimientos propio.	54
26.	Esquema estructuración de los datos	55
27.	Ejemplo de salida de la frase una vez tratada.	56
28.	Clases propias más importantes.	57
29.	Idea principal del clasificador de texto	58
30.	Resultados obtenidos en clasificación de texto sin diccionario.	59
31.	Comparativa entre guardar diccionario en una <i>array</i> o en un árbol AVL. .	61

32. Ejemplos de ejecución del algoritmo por diccionario almacenado en un árbol AVL en análisis por temática.	62
33. Ejemplos de error en el uso de un <i>stemmer</i> en clasificación de texto.	63
34. Ejemplos de error en el uso de un <i>stemmer</i> con derivaciones de la palabra amor.	64
35. Resultados obtenidos en clasificación de sentimientos.	66
36. Menú principal del programa.	67
37. Ejemplo de un usuario no verificado que publica una verdad.	68
38. Ejemplo de un usuario verificado que publica una verdad.	68
39. Ejemplo de un usuario verificado que publica una verdad.	68
40. Ejemplo de un usuario verificado que publica un tweet que no es verdad pero hay más gente opinando igual.	69
41. Ejemplo de un usuario verificado que publica una opinion.	69
42. Ejemplo de un usuario no verificado que publica una opinion.	70
43. Ejemplo de tweet demasiado corto que no se encuentra sobre que tema trata.	70
44. Análisis de un usuario considerado <i>hater troll</i>	71
45. Análisis de si el periodista Josep Pedrerol es o no un <i>hater troll</i>	71
46. Bot o observer troll.	72
47. Bot or observer troll detectrado por <i>trolls detector</i>	72
48. Resultados obtenidos con un usuario clasificado como <i>troll spammer</i> (usuario inventado).	73
49. Clasificación suario 'BangtanPromoARG' por <i>trolls detector</i>	73
50. Usuario 'BangtanPromoARG' clasificado como <i>troll spammer</i>	74
51. Resultados obtenidos con un usuario que no es no es un bot, spammer o observer troll.	74
52. Ejemplo opción 4, analizar un tweet sobre rugby.	74
53. Ejemplo opción 4, analizar un tweet sobre boxeo.	75
54. Ejemplo opción 5, analizar un tweet alegre.	75
55. Ejemplo opción 5, analizar un tweet sobre miedo.	75
56. Instalar MongoDB.	84
57. Ejecutar MongoDB en nuestro ordenador.	84
58. Funcionalidades MongoDB.	85
59. Conexión con Twitter desde Python.	86
60. Obtener tweets por Hashtag.	86
61. Obtener últimos <i>X</i> tweets.	87

62. Obtener información del usuario.	87
63. Ontología de los deportes.	88

Acrónimos

SVM: Support Vector Machine.

AVL Tree: Adelson-Velskii y Landis Tree.

XGBoost: Xtreme Gradient Boosting Model.

PLN: Procesamiento de Lenguajes Naturales

GLM: Modelos Lineales Generalizados

NLTK: Natural Language ToolKit.

RFM: Random Forest Model.

OS: Operative System.

DB: Data Base.

FC: Fútbol Club.

NB: Naive Bayes.

1. Introducción

1.1. Contexto

En la era de la tecnología en la que vivimos cada vez se da más importancia a las redes sociales. Ya no se usan las redes sociales exclusivamente para hablar y compartir cosas con amigos, ahora también lo usan las marcas para hacer campañas publicitarias, los informativos para adelantar noticias y los equipos para informar a sus seguidores. Además de la importancia que de por sí tienen, actualmente, la sociedad informatizada en la que nos hemos adentrado en el último siglo, ha hecho que el interés por el análisis de la opinión y veracidad de la información crezca exponencialmente en estos últimos años.

Los seres humanos siempre han querido conocer de inmediato los acontecimientos que suceden a su alrededor. Para ello antiguamente se recurrió de los diarios pero solo te decían lo que había sucedido el día anterior, después fue el tiempo de la televisión con sus informativos dos veces al día, a continuación apareció internet y los diarios y cadenas de televisión se aprovecharon de ello para ganarle la partida al otro e informar de los hechos en cuestión de minutos.

A partir de esta guerra interna que hay en el mundo del periodismo de informar el primero y tener imágenes de todo el mundo, actualmente, los periodistas, se ayudan de las redes sociales para enterarse de lo que sucede en el mundo. Esto es una gran idea pero, en el mundo de internet, siempre hay gente mintiendo.

Dentro de este cambio que han sufrido las redes sociales cada vez es más importante saber en quien se puede confiar y que *posts* hay que ignorar. Quien sube estas noticias son los *trolls*, gente que publica falsos *posts* simplemente para divertirse o porque no saben sobre lo que están escribiendo o *retweeteando*.

De los nombrados *trolls* hay de muchos tipos. Una de las clasificaciones es según el periódico 'el confidencial' [6]:

1. **El principiante.** Es aquel que se abre un perfil, amparado en el anonimato, y con un número de seguidores insignificante. Generalmente, su troleo pasará inadvertido y no alcanzará su propósito ni de lejos.
2. **El estratega.** Tiene claro el objetivo, contactará con usuarios con identidad digi-

tal, buen nivel de penetración, credibilidad y un buen número de seguidores. Hará que ellos se encarguen de propagar su crítica y sin duda pueden conseguir hacer ruido en las redes.

3. **El sarcástico.** Te puede sacar de quicio en un momento dado. Le encanta liarte. Su finalidad suele ser la de provocar para que no te olvides de él y por supuesto que no te lo calles. Necesita que cuentes qué hace y que señales quién es a los demás.
4. **El sádico.** Su objetivo es disfrutar con tu dolor y humillación y no parará hasta saciar su sed con tu sufrimiento. Es capaz de sacar de contexto cualquier acción que hagas por el mero hecho de ver cómo los demás hacen carnaza con ello. Lo que peor que puedes hacer es contestar a su provocación pues esa será la señal de su ataque.
5. **El arrepentido.** Es el que finalmente saca a pasear su conciencia y le sobreponen sus acciones. Pero no te fíes, casi siempre vuelve.
6. **El cansino.** Aparece sorpresivamente de la nada metiéndose en una conversación y no para de darte palique hasta límites insospechados. Debes pararle los pies o se convertirá en tu peor pesadilla. Bloquearlo puede ayudar, aunque hay gestores de redes que permiten saltarse el bloqueo.
7. **El omnipresente.** No concibe otra forma de estar en redes sociales que no sea trolleando a diestro y siniestro, especialmente a usuarios con muchos seguidores. Su finalidad, en el caso de que no tenga credibilidad, es la del egocentrismo.
8. **El lerdo.** Va de listo utilizando un usuario falso y publica el mismo mensaje en su cuenta personal. Le pillas y le hundes, a la par que regalas un buen momento a tus seguidores.
9. **El frustrado.** Conocido también como hater. La difamación, el insulto son su sistema de trolley. Un desencadenante fundamental para ser un hater es el odio. Su frustración y decepción, en algunos casos interna, le hacen imponer su criterio sobre cualquier otro. Es más, harán lo imposible por demostrar que quien no piensa como él es despreciable y humillable. Cualquier cosa que hagas estará mal y será reprochable. Aunque tengas paciencia muy probablemente acabarás buscando un abogado.
10. **El oportunista.** Aprovecha el trending topic del momento para insultar a quienes estén relacionados con el tema o para hablar de sus temas recurrentes incluyendo

el hashtag sin que éste tenga relación alguna con ellos. Todo por intentar tener un poco de visibilidad.

11. **El suplantador.** Simula ser otro usuario para intentar dañar su imagen con mensajes que le perjudican o, si se trata de alguien popular, con el fin de captar muchos seguidores para terminar vendiendo la cuenta.
12. **El paródico.** No intenta suplantar a otro usuario, sino parodiarle, unas veces con el fin de burlarse de él de forma más o menos sana y otras con la intención de difamarle.
13. **El zombi.** Es una cuenta creada o comprada por un troll con el fin de lanzar mensajes automatizados para atacar a alguien.
14. **El vampiro.** Es uno de los más peligrosos. Se alimenta del sufrimiento de sus víctimas y en muchos casos vive obsesionado con ellas, monitorizando absolutamente todas sus actividades dentro y fuera de las redes con la finalidad de lanzar sus ataques. Es un delincuente y cada día hay más casos en los tribunales.
15. **El spammer** Es el típico taladro que insiste con un tema hasta la saciedad. Comenta sólo para insertar un enlace que le permita ganar tráfico o posicionamiento, y es capaz de escribir hasta 20 comentarios seguidos con el mismo link. Es insaciable y muy cansino, además de omnipresente: está en todos los espacios virtuales y en todas las redes sociales.
16. **El cazador de trolls.** Hay usuarios que terminan vengándose y trolean a sus propios trolls, ya sea desde sus cuentas personales o creando perfiles específicos para la ocasión.

1.2. Motivaciones

Las opiniones y sentimientos de las personas siempre han sido una valiosa fuente de información. Por ejemplo, las grandes empresas como Apple o Samsung necesitan conocer el impacto público que producen y mantener controlada al día la opinión pública que será su medidor de éxito de su venta ya que no les vale con vender mucho una vez si no que necesitan vender mucho y muchas veces. Esto les hace recolectar toda la información posible de los clientes ya sea vía mail, redes sociales o cualquier otra forma de saber la opinión de la gente sobre sus productos. Esto también se aplica a políticos para saber si están enfocando de manera correcta su campaña electoral e incluso las empresas de marketing siempre han tomado la opinión social como el centro de sus actividades. Incluso se ha utilizado esta información como estudios de mercado para venderlos a las empresas y no son más que una recolección de datos.

El problema de estos textos que depositamos en nuestras redes sociales es la fiabilidad de la fuente. Cualquier persona que tenga una cuenta puede acceder y comentar sobre el tema que él quiera y ahí es donde aparecen los trolls comentados anteriormente. A veces son comentarios inofensivos que no molestan a nadie como serían comentarios no verificados del estilo de 'Yo cocino mejor que Ferran Adrià' donde no hay ninguna maldad sino que se podría considerar como un comentario irónico o gracioso.

El motivo principal para la elección de este proyecto ha sido raíz a las mentiras que se llevaron a cabo durante el atentado de Barcelona en las Ramblas de Barcelona, yo era una de las personas que estaban de vacaciones pero que tenía familia en Barcelona y que la gente con o sin maldad se dedicaba a retuitear todo sin saber ni siquiera si era verdad o se trataba de una fuente fiable y lo único que hacían era provocar pánico, angustia y desconcierto a miles de familias así como aquellos usuarios que solo retuiteaban todo lo que veían provocando un SPAM en el muro y los que en momentos de crisis en vez de adoptar la calma e intentar entender a las familias que lo están pasando mal solo incitan a la violencia.

1.3. Objetivos Trabajo Final de Grado

Con el auge de las redes sociales en la última década, los usuarios de redes sociales tales como Twitter, Facebook o Instagram no paran de crecer, y en ellos los usuarios no paran de volcar y compartir opiniones y sentimientos sobre cualquier tema, evidentemente lo que más se comenta es lo que suceda en la actualidad de aquel momento, creando así cantidades inmanejables de datos. El problema de estos datos es el analizarlos y guardarlos de forma estructurada, que sea útil y que nos permita analizar los resultados de forma que éstos nos lleven a conclusiones beneficiosas.

Este proyecto se centrará en cazar a los *trolls* de *Tweeter* que intentan engañar en el mundo de los deportes, a aquellos usuarios que puedan estar utilizando su cuenta para hacer SPAM y los perfiles que sean agresivos. Para ello se hará un estudio previo del perfil del usuario. Se pretende clasificar *tweets* falsos y verdaderos clasificando estos primero en el deporte al que se vincula el *tweet* (fútbol, baloncesto, balonmano, golf, rugby, etc) y sobre qué tratan sus últimos tweets, según el tono en el que está escrito con el uso de análisis del sentimiento y si escribe tweets de forma que se podría considerar agresiva. Todo esto pretende no solo decir si el texto es positivo o negativo sino una clasificación más refinada sobre tipo de sentimiento con el que está escrito. Se analizará la información de su perfil como si su cuenta es verificada. También se tendrá en cuenta a quien hace menciones y retweets y como son esas cuentas aplicando todos los análisis anteriores.

Como ya se ha comentado nos centraremos en la sección de noticias donde hay más mentiras (los deportes) pero, se pretende hacer de tal manera que, solo cambiando los diccionarios de palabras usadas, se puedan aplicar las mismas técnicas y nos permita conocer „^a tiempo real” la fiabilidad de los tweets que se leen o que se están volcando en twitter en ese momento.

Destacar que este proyecto solo intenta ser una vía para ayudar a los usuarios a descubrir la veracidad sobre lo que están leyendo y a aconsejar dejar de seguir todas aquellas cuentas que les podrían causar problemas como las agresivas o que no les aporten nada como las que solo hacen anuncios o SPAM.

Para concluir se quiere comentar que muy poca gente ha investigado sobre esto y dichos estudios no se han hecho públicos y tampoco se tiene conciencia de su fiabilidad. También destacar que todos los análisis de texto se hacen sobre textos escritos en castellano la cual cosa complica aún más la investigación dado que todas las herramientas que se han usado se han tenido que adaptar a las necesidades del proyecto llegando al punto que se ha tenido que crear una ontología propia en castellano y un algoritmo para la clasificación tanto por temática como del sentimiento.

2. Trabajo Relacionado

Este trabajo se trata de un proyecto muy ambicioso y como tal abarca un gran abanico de temáticas, por lo tanto, hay mucho trabajo relacionado con partes del proyecto. La gran diferencia es que este proyecto está en español, un gran inconveniente en el momento de utilizar librerías ya creadas dado que la gran mayoría están en inglés. Otra diferencia sería que no se centra solo en un concepto en la hora de clasificar y que es fácilmente exportable a otros ámbitos la simple creación de un nuevo diccionario.

Empezamos con un proyecto que se está llevando a cabo desde el 2014 hay un proyecto financiado por la UE llamado *Pheme*. El cual es llamado el detector de mentiras de internet. Pheme dice pretender solucionar el cuarto problema del Big Data; la veracidad. Analizará las publicaciones sociales en redes como Twitter y Facebook, las comparará y las contrastará con otras fuentes de información, con el fin de establecer su veracidad y, en el caso de que se clasifique como bulo, señalar la razón (estableciendo si estamos ante una simple especulación, una controversia, un ejercicio deliberado de desinformación, etc).

Luego hay muchos trabajos relacionados respecto a la psicología que hay que aplicar para detectar perfiles falsos en twitter como un artículo que presentó el diario abc¹ y también sus características.² Incluso hay quien le dio la vuelta a la tortilla y decidió buscar las características de los tweets confiables³.

De trabajos teóricos para detectar noticias falsas hay muchos⁴⁵⁶ el problema se presenta en que siempre tiene que haber alguien leyéndolo y revisándolo presencialmente, nosotros queremos que solo haya que dar el visto bueno en el último punto de la decisión y solamente en casos que haya duda, es decir, hacerlo automático en los casos evidentes. También respecto a los tipos de *troll* que existen⁷⁸ incluso *papers* donde se hablan de las llamadas *fake news*⁹.

¹<http://www.abc.es/tecnologia/redes/20140320/abci-faketwitter-saber-twitter-falsos-201403191326.html> [1]

²<http://www.outono.net/elentir/2013/02/02/twitter-5-formas-de-identificar-a-un-troll> [2]

³<http://www.josemorenojimenez.com/2012/03/20/caracteristicas-de-un-tweet-confiable/>

⁴<https://www.entrepreneur.com/article/292342> [3]

⁵<http://www.eltiempo.com/tecnosfera/novedades-tecnologia/como-identificar-noticias-falsas-en-redes-sociales> [4]

⁶<http://www.abc.es/tecnologia/redes/20131210/abci-como-detectar-noticia-falsa-201312092125.html> [5]

⁷https://blogs.elconfidencial.com/tecnologia/elclubdelalucha/2015-05-09/troll-redes-sociales-ciberacoso_790558/ [6]

⁸<http://www.elmundo.es/f5/comparte/2017/10/19/59e77fb22601db82d8b459f.html> [7]

⁹http://www.kdd.org/exploration_files/19-1-Article2.pdf

Después de todos estos trabajos teóricos también podemos encontrar algunas implementaciones para encontrar bots¹⁰ e incluso herramientas para encontrar seguidores falsos¹¹. Se quiere destacar que este código no se encuentran en Internet dado que son de uso comercial y por lo tanto todo el código es propio y de lo único que no es propio son librerías externas como *scikit-learn* donde se encuentran algoritmos como *Naive Bayes* o *XGBoost* o API's como *Tweepy* para hacer la conexión con Twitter.

Este tema de encontrar trolls y de la preocupación que hay de que la tecnología no acabe con las personas es una área de interés donde se han introducido grandes empresas como Google con su *Troll detection*¹², la librería de Python de referencia Scikit-Learn¹³ e incluso el gran medio de comunicación internacional BBC habla al respecto¹⁴.

Por otra parte, esta clasificación de los usuarios en trolls, también hay quien, contando que hay trolls agresivos, y no que solo dicen mentiras si no que intentan incordiar a los usuarios o que aplican lo que ahora se conoce como Cyber Bullying, dado esto, se han escrito artículos para la detección de estos usuarios¹⁵.

Para finalizar solamente comentar que no se ha utilizado trabajo de terceros que no sean librerías y API's de Python, es decir que, tanto los diccionarios de los deportes y sentimientos son propios así como los árboles y demás código que no se encuentre en una librería o API de Python.

¹⁰ <https://botometer.iuni.iu.edu/#!/>

¹¹ <https://www.genbeta.com/redes-sociales-y-comunidades/7-herramientas-para-detectar-si-tienes-seguidores-falsos> [10]

¹² <https://www.theverge.com/2017/2/23/14713496/google-jigsaw-perspective-software-ai-machine-learning-developers> [11]

¹³ <http://blog.kaggle.com/2012/09/26/impermium-andreas-blog/>

¹⁴ <http://www.bbc.com/news/technology-38181158> [12]

¹⁵ <https://academic.oup.com/jigpal/article-abstract/24/1/42/2893010?redirectedFrom=fulltext> [13]

3. Trolls Detector

Troll Detector pretende ayudar a descubrir mentiras en Twitter o si más no, dar o quitar veracidad a las noticias que leemos en las redes sociales, concretamente en Twitter.

Este proyecto consta de cuatro grandes fases donde se intentara conseguir el objetivo de si un tweet de una persona es falso o no. Estas cuatro fases se dividen en:

1. **Fase 0**, donde se tendrá que obtener los últimos tweets de esa persona así como la creación y obtención de los diccionarios. Se llama fase 0 por ser una tarea previa a todo el análisis y ser la parte mecánica o tarea sucia del trabajo donde no hay realmente un análisis de la información más allá de descartar todas aquellas palabras cuya aportación de información es nula (como seria el caso de artículos y preposiciones entre otras).
2. La siguiente fase, **fase 1**, es la encargada de decir sobre que estamos hablando, en este trabajo acotado, nos dirá únicamente sobre que deporte o deportes habla esa persona en su Twitter usando la hipótesis propia inicial de que una persona no puede saber de todo o si más no, solo puede ser experto en un acotado número de deportes.
3. A continuación, la **fase 2** pretende saber el tono/sentimiento con el que el autor del tweet lo esta escribiendo. Esto se hace con una hipótesis propia donde se especula que cuando estamos enfadados se escribe lo primero que pensamos e igual no se ciñe del todo a la realidad así como, cuando estamos alegres lo publicamos pensando que es verdad aunque puede que nosotros mismos estamos equivocados y escribamos una mentira. Esta fase se encuentra con muchos problemas como la hironía o la mala sintaxis y de más problemas que serán comentados más adelante.
4. Para finalizar, la **fase 3**, donde se tendrá que recopilar la información anterior, contrastar si la cuenta que lo escribe es un perfil verificado o no, dado que las cuentas verificadas suelen ser de gente pública y por lo tanto, si cuentan mentiras, tendrán un castigo social más importante que si lo escribe alguien cuyo nombre no figura en Twitter. También detectara aquellos usuarios que siempre estén al ataque contra otros o que escriban siempre de una forma furiosa y los perfiles que intenten hacer SPAM o decir siempre lo mismo.

En la siguiente figura se puede observar un esquema general del proyecto para hacerse una idea de como funciona.

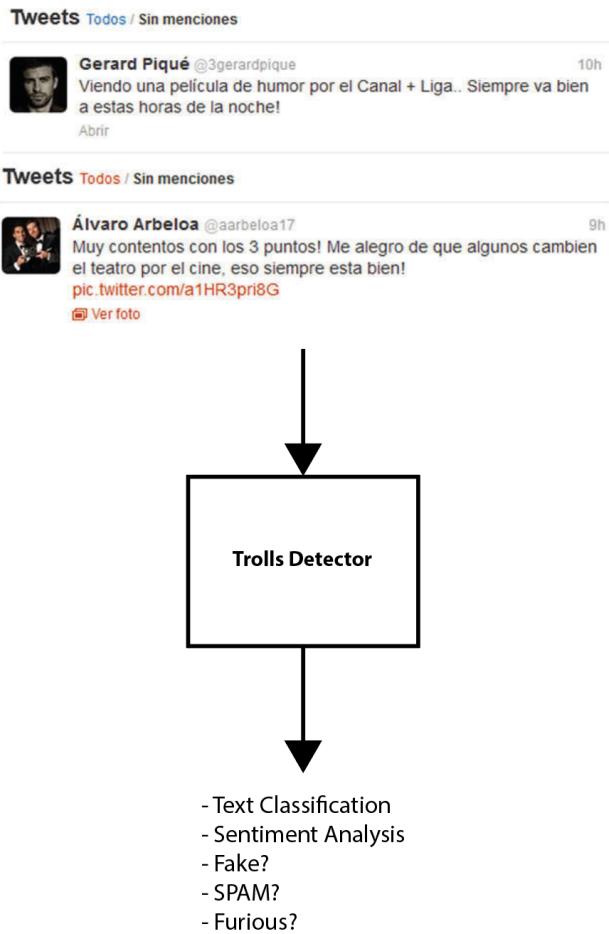


Figura 1: Esquema general del proyecto

3.1. Tecnologías utilizadas en todas las fases

En este apartado se darán a conocer las tecnologías usadas para la implementación del trabajo y su pertinente justificación. Se explican por separado las librerías de python y clases propias utilizadas en cada fase del proyecto.

Python y R son los lenguajes de programación más usados en *data science*. Para este proyecto se ha escogido el uso de Python dado que [24] [25]:

1. Python es fácil de utilizar: Python tiene una reputación de ser fácil de aprender.
Con una sintaxis legible,
2. Python es ideal para principiantes o para científicos de datos que desean adquirir conocimiento en este lenguaje.
3. Python es versátil: Como un lenguaje de propósito general, Python es una herramienta rápida y potente que tiene mucha capacidad. No importa cual sea el problema que quieras resolver, Python puede ayudarte a realizar la tarea, esto gracias a la gran cantidad de librerías con las que cuenta.
4. Python es mejor para construir herramientas de análisis: R y Python son bastante buenos si desea analizar un conjunto de datos, pero cuando se trata de construir un servicio web para que otros utilicen los métodos desarrollados, Python es el camino a seguir.
5. Visualización de datos con Python: En este aspecto es donde R generalmente le gana a Python. R tiene un amplio rango de herramientas de visualización, tales como, ggplot2, rCharts y googleVis. Más aunque
6. Python no se presta en forma natural para la visualización, tiene una amplia gama de librerías disponibles, tales como Matplotlib, Plot.ly y Seaborn.
7. La comunidad de Python esta creciendo: Python tiene una gran comunidad, que incluye una fuerte y creciente presencia en la comunidad de las ciencias de datos. PyPi es un lugar útil para explorar todo el alcance de lo que está desarrollando la comunidad.
8. Python es mejor para Deep Learning: Hay muchos paquetes, como Theano, Keras y TensorFlow, que hacen que sea realmente fácil crear redes neuronales profundas con Python, y aunque algunos de estos paquetes están siendo portados a R, el soporte disponible en Python es muy superior.

3.1.1. MongoDB

MongoDB es un sistema de bases de datos NoSql de código abierto orientado a documentos. En lugar de almacenar la información en tablas como se haría en una base de datos relacional, MongoDB guarda la información en forma de documentos JSON, haciendo que la integración de este sistema de información sea mucho más simple y rápido que otros sistemas de bases de datos.

MongoDB ha sido muy útil para almacenar de manera eficiente grandes conjuntos de tuits. Para trabajar con este sistema de bases de datos se ha empleado la librería PyMongo, que es la distribución para Python recomendada en la web oficial de MongoDB y que contiene todas las herramientas necesarias. Para el proyecto se ha empleado la versión 3.0.4 de MongoDB.

MongoDB es un sistema fácil de utilizar y que funciona muy bien para guardar textos pero vamos a ver una tabla donde se puede comparar una base de datos como MongoDB con una SQL [26].

Feature	NoSQL Database	SQL Database
Performance	High ✓	Low
Reliability	Poor	Good ✓
Availability	Good	Good
Consistency	Poor	Good ✓
Dara storage	Optimized for huge data ✓	Medium sized to large
Scalability	High ✓	High but expensive

3.1.2. Pymongo 3.7.1

Pymongo es una librería de Python para poder conectarnos a una base de datos MongoDB. Para utilizar esta librería se utilizará una clase propia [8].

DB
+pymongo: MongoClient
+Dictionary GET_dictionary_from_DB()
+Sentiment GET_SA_from_DB()
+String GET_text_to_classify(type)
+INSERT_json_toDB(db_name, cl_name, data_json)
+INSERT_toDB(db_name, cl_name, text, word_list, top_words, top_words_percentages, type)
+INSERT_to_nacional_DB(data)

Figura 2: Clase DB utilizada para la comunicación entre el programa y la base de datos en MongoDB

3.1.3. AVL Tree

Todas las palabras están guardadas en una base de datos MongoDB pero con la finalidad de hacer la búsqueda más rápida una vez se consultan por primera vez se quedan guardadas en un árbol propio, concretamente en un AVL tree.

Un árbol es un tipo abstracto de datos (TAD) muy utilizado que imita la estructura jerárquica de un árbol, con un valor en la raíz y subárboles con un nodo padre, representado como un conjunto de nodos enlazados.

La ventaja que tiene este tipo de árboles (AVL) es que están siempre equilibrados de tal modo que para todos los nodos, la altura de la rama izquierda no difiere en más de una unidad de la altura de la rama derecha o viceversa. Gracias a esta forma de equilibrio (o balanceo), la complejidad de una búsqueda en uno de estos árboles se mantiene siempre en orden de complejidad $O(\log n)$. El factor de equilibrio puede ser almacenado directamente en cada nodo o ser computado a partir de las alturas de los subárboles.

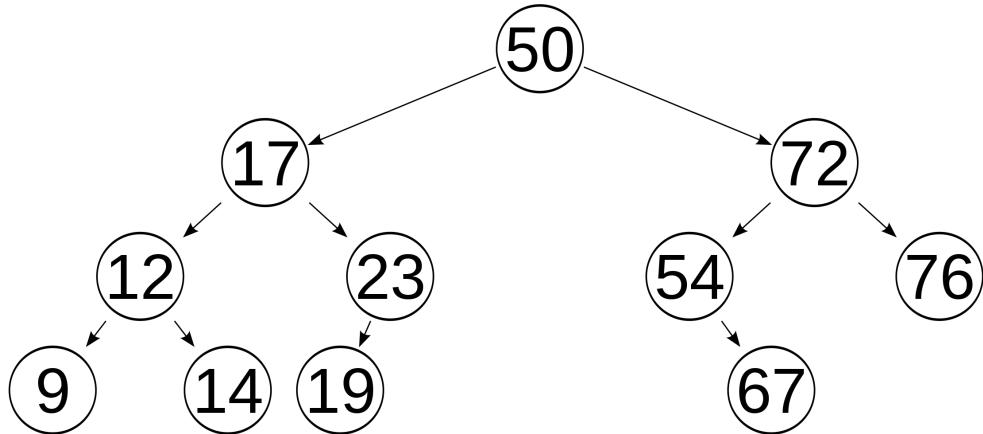


Figura 3: Ejemplo AVL Tree

Como se puede observar en la figura anterior, un árbol es únicamente una estructura de datos ordenada. En la figura es un orden numérico (de menos a más valor) y el árbol utilizado en este trabajo está ordenado alfabéticamente. Con este árbol tendremos un coste inicial de creación pero luego el coste medio de la búsqueda de palabras será mucho menor.

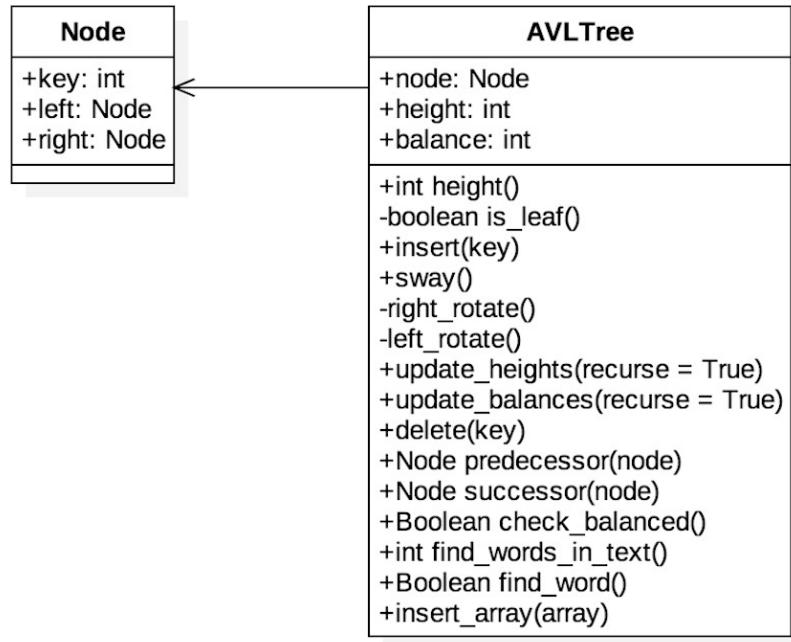


Figura 4: Diagrama de la clase propia AVL-TREE utilizada para la interacción con el árbol

3.2. Fase 0 - Extracción de la información y creación de los datasets

3.2.1. Concepto

Esta primera fase 0 o fase inicial se irá modificando a lo largo de la implementación según se necesite. Pretende englobar tanto la extracción de los tweets como la creación de los diccionarios.

Se pretende así que cada deporte tenga tres diccionarios asociados (palabras excluyentes, palabras vinculantes principales y palabras vinculantes secundarias). También se tendrán diccionarios para el análisis de las frases; palabras vacías, que son aquellas palabras que no aportan nada a la hora de clasificar el texto como podrían ser artículos y preposiciones. Por último se tendrá un diccionario asociado a cada uno de los sentimientos a analizar; alegría, amor, enfado, miedo, sorpresa y tristeza.

Esta fase también es la encargada de la extracción de los tweets, no solo los X últimos del usuario para saber de qué deportes habla sino también de guardar información sobre el usuario (como por ejemplo si el perfil se trata de una cuenta verificada), de si el tweet se trata de un retweet entrar en el perfil del usuario que lo ha publicado y hacer lo mismo que si fuese esta persona inicial y por último comprobar los hashtags que ese tweet pueda contener.

3.2.2. Software

El esquema de esta fase se puede ver en la siguiente imagen:

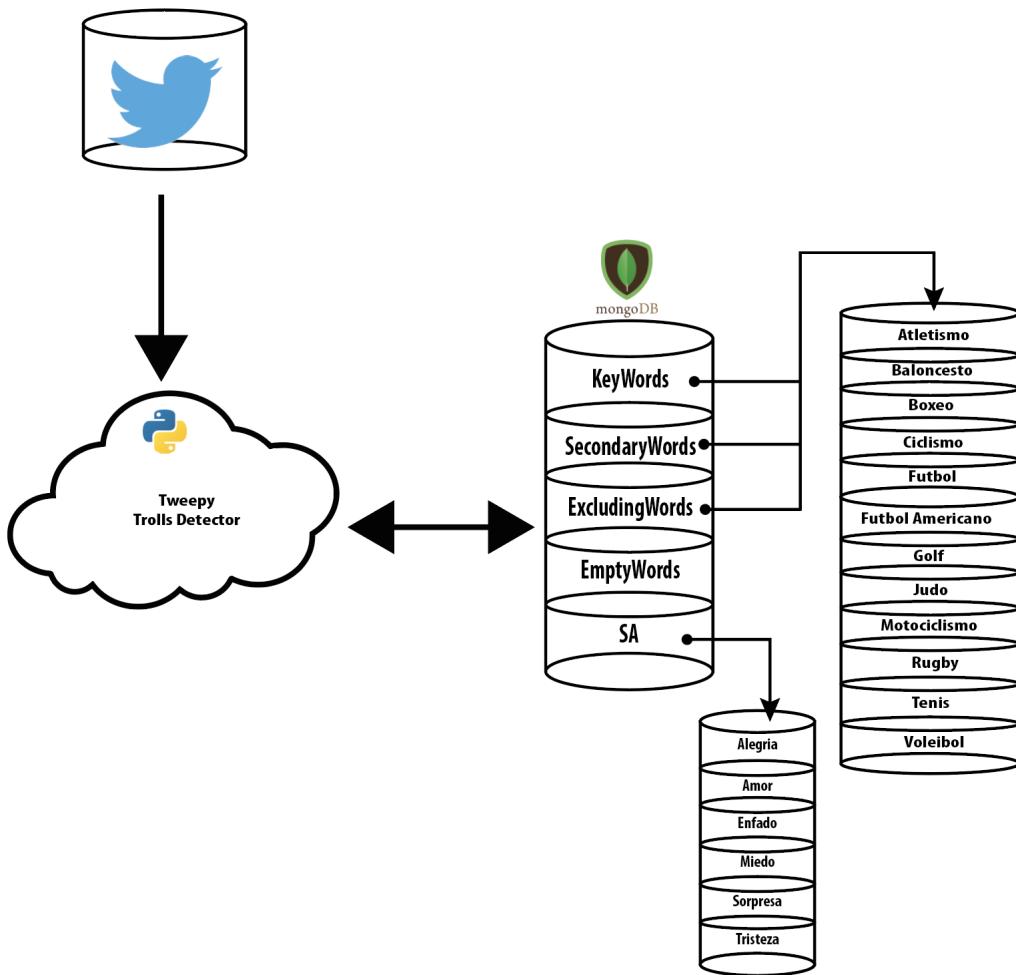


Figura 5: Estructura de extracción de tweets y interacción con diccionario

Donde se puede ver que se extrae la información de twitter a través de la librería tweepy, se proceso con el detector de mentiras y se interacciona con la base de datos donde previamente se habían guardado los diccionarios.

Las tablas de la base de datos son las siguientes:

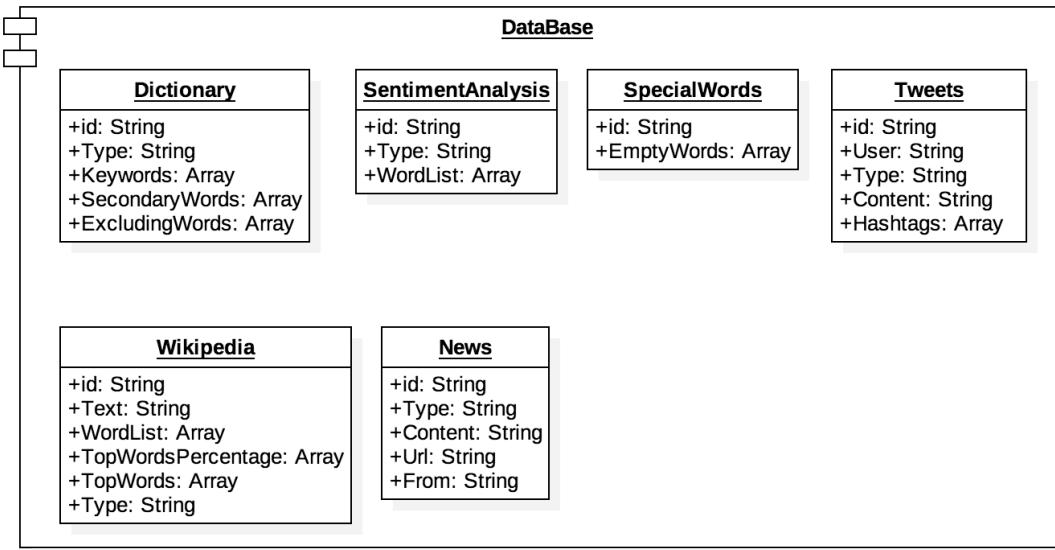


Figura 6: Tablas en la base de datos del proyecto

3.2.2.1 Wikipedia 1.4.0

Esta librería se empezó a usar con la finalidad de crear diccionarios automáticamente, simplemente coger todas las palabras que salen de una búsqueda en wikipedia, excluir todos los artículos, proposicionales y palabras que no aportan valor de clasificación.

De todas las palabras obtenidas se cogían el 30 % de las palabras más repetidas. Esto permitió empezar a hacer pruebas de los algoritmos pero se descarto por falta de precisión y se empezó a hacer un diccionario propio que ganó riqueza por algunas palabras encontradas gracias a este algoritmo [18].

3.2.2.2 Pymongo 2.7

Pymongo es una librería de Python para poder conectarnos a una base de datos MongoDB. A continuación se muestra como hacer las principales operaciones con la librería [8].

```
def connectDB(name_db, name_collection):
    client = MongoClient('localhost', 27017)
    return client[name_db], client[name_db][name_collection]
```

Figura 7: Conectar con la Base de Datos

```
def GET_dictionary_from_DB():
    db, cl = connectDB('train', 'dictionary')

    cursor = cl.find({})
    data = []
    for document in cursor:

        data.append(Utils.Dictionary(
            document['type'],
            document['keywords'],
            document['secondaryWords'],
            document['excludingWords']
        ))
    return data
```

Figura 8: Obtener el diccionario de nuestra BBDD

```
def INSERT_toDB(db_name, cl_name, text, word_list, top_words, top_words_percentages, type):
    db, cl = connectDB(db_name, cl_name)
    data_json = {
        'text': text,
        'word_list': word_list,
        'top_words': top_words,
        'top_words_percentages': top_words_percentages,
        'type': type
    }
    result = cl.insert_one(data_json)
```

Figura 9: Insertar en la BBDD

3.2.2.3 Tweepy 3.5

Tweepy es una librería de código abierto para Python que incluye todo el conjunto de funciones necesarias para comunicar con Twitter mediante las API's definidas por este. Las funciones definidas por Tweepy simplifican la conexión y búsquedas con Twitter. Por ejemplo, toda la conexión con Twitter debe estar certificada con un autor y, mientras que por defecto habría que configurar esta conexión mediante otra librería como sería Python-Auth y establecer cada conexión manualmente, Tweepy simplifica esto con unas funciones que simplemente esperan está autenticación como parámetro para poder configurarlo automáticamente. Estos parámetros son 4 tokens necesarios y en el caso de la búsqueda solo tenemos que indicarle los parámetros que solicita Twitter, toda la complejidad de las conexiones la trata internamente simplificando el trabajo inmensamente. Para el proyecto se ha empleado la versión 3.3 de Tweepy [16].

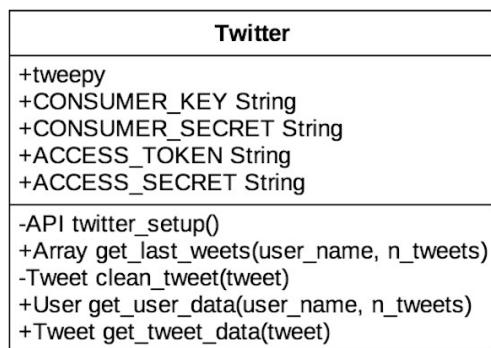


Figura 10: Diagrama de la clase propia Twitter encargada de la comunicación con twitter mediante la librería tweepy

3.3. Fase 1 - Clasificación de texto según temática

3.3.1. Concepto

Para la clasificación de los tweets según su temática se hará uso de diferentes algoritmos y técnicas con la finalidad de conocer sobre qué temas escribe cada usuario de Twitter. La finalidad de esta fase por una parte es acotar los temas sobre los que habla cada usuario dada la suposición que lo más común en usuarios de Twitter (si nos centramos en los deportes) es que les guste mucho uno o dos deportes y sepan mucho de ellos y que luego sean simpatizantes de otros deportes. Por ejemplo en mi caso me gusta mucho el fútbol y el rugby y a veces miro el baloncesto o las carreras de motos gp pero no entiendo tanto de esos deportes como de mis favoritos.

Para ello esta fase se centrará en recopilar los últimos tweets de un usuario, analizar sus *hashtags*, investigar la posible mención de usuarios en los tweets y averiguar también los deportes de aquel usuario y por último si hizo algún *retweet* obtener la información sobre la otra cuenta obtenida en la fase anterior.

Con el fin de conseguir todo esto se aplicarán diferentes técnicas para obtener lo que realmente nos interesa de cada tweet (palabras clave, *hashtags*, menciones a otros usuarios, etc) y una vez extraídas y *tokenizadas* se aplicarán diferentes técnicas para su clasificación como sería primero eliminación de las palabras que no nos interesan, aplicación de diferentes algoritmos como Naive Bayes y el uso de un diccionario propio con palabras clave, palabras secundarias y palabras excluyentes de cada deporte.

3.3.2. Software

Para esta fase damos por sentado que ya tenemos acceso a la API de Twitter para obtener los tweets. Se ha decidido descargar los 20 tweets de cada usuario, teniendo en cuenta que si un tweet tiene un *hashtag* o algún usuario se investigara también ese caso, se cree que ya será suficiente pero si los resultados (porcentaje del deporte al que pertenece) no es suficiente se cogerán 10 más.

Por cada tweet se elimina las palabras que se cree que no aportan nada. Estas palabras se encuentran en la base de datos de MongoDB con el nombre de *emptyWords* y para acceder a ellas se hace mediante la librería *pymongo*. También se eliminan los signos de puntuación mediante la combinación de las librerías *re* y *string*. Por último se duplica la frase pero esta vez solo con la raíz de cada palabra, esto se conseguirá con la librería *SnowballStemmer* la cual se puede utilizar en Español aunque no se obtienen los mejores resultados con ella.

Una vez se tienen las palabras clave de cada frase se analiza con *Naive Bayes*, SVM y con el diccionario propio creado.

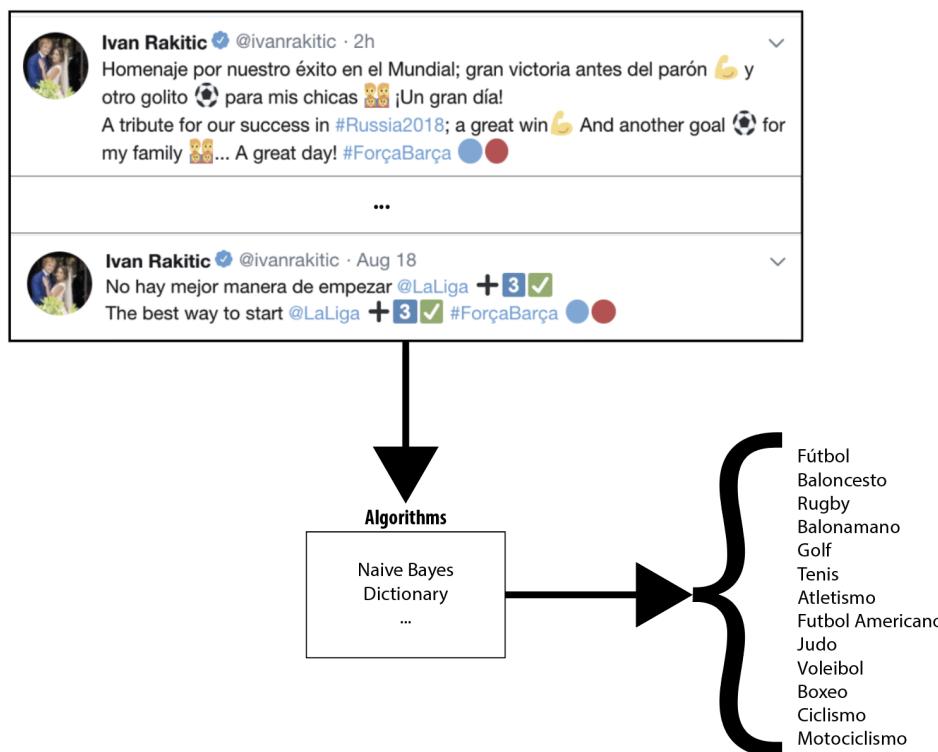


Figura 11: Diagrama fase 1 clasificación de texto según temática

3.3.2.1 Stemmer

Por razones gramaticales, los documentos van a utilizar diferentes formas de una palabra, como jugar, juegan y jugaron o jugador y jugadores. Además, hay familias de palabras derivadas relacionadas con significados similares, como vivir y convivir. En muchas situaciones, parece que sería útil buscar una de estas palabras para devolver documentos que contengan otra palabra en el conjunto.

El objetivo tanto de la derivación (stemmer) como de la lematización (lemmatization) es reducir las formas flexivas y, a veces, las formas derivadas de una palabra a una forma básica común.

Sin embargo, las dos palabras difieren en su forma de hacerlo. Stemming usualmente se refiere a un crudo proceso heurístico que elimina los extremos de las palabras con la esperanza de lograr este objetivo correctamente con un alto porcentaje, y a menudo incluye la eliminación de los afijos derivacionales. La lematización generalmente se refiere a hacer las cosas correctamente con el uso de un vocabulario y un análisis morfológico de las palabras, normalmente con el objetivo de eliminar solamente las terminaciones y devolver la forma base o diccionario de una palabra, lo que se conoce como el lema (lemma).

Como este trabajo esta en castellano, una lengua muy rica en su vocabulario y derivaciones, se ha optado por usar un Stemmer y no un lematizador dado que se obtienen mejores resultados.

```
from nltk.stem import SnowballStemmer
from nltk.tokenize import sent_tokenize, word_tokenize
def stem(word):
    stemmer = SnowballStemmer('spanish')
    return stemmer.stem(word)
new_text = "Jugando con mi sobrino nos dimos cuenta de lo divertido que es el escondite"
text_tokens = word_tokenize(new_text)
stemmed = []
for item in text_tokens:
    stemmed.append(stem(item))
print()
print('Input: ', new_text )
print('Output:', stemmed )
```

Figura 12: Código ejemplo utilización VADER

```
Input: Jugando con mi sobrino nos dimos cuenta de lo divertido que es el escondite
Output: ['jug', 'con', 'mi', 'sobrin', 'nos', 'dim', 'cuent', 'de', 'lo', 'divert', 'que', 'es', 'el', 'escondit']
```

Figura 13: Output ejemplo utilización VADER

3.3.2.2 Naive Bayes

El clasificador Naive Bayes es un clasificador probabilístico que se basa en el teorema de Bayes con suposiciones de independencia. Es una de las técnicas de clasificación de texto más básicas con varias aplicaciones en la detección de spam en el correo electrónico, clasificación de correo electrónico personal, categorización de documentos, detección de lenguaje y detección de sentimiento.

Como se ha comentado se basa en el teorema de Bayes así que primero lo comentaremos y se usará como ejemplo la clasificación de texto. Se usa para predecir la probabilidad condicional de que un documento pertenezca a una clase $P(c_i|d_j)$ a partir de la probabilidad de los documentos dada la clase $P(d_j | c_i)$ y la probabilidad a priori de la clase en el conjunto de entrenamiento $P(c_i)$ [27].

$$P(c_i|d_j) = \frac{P(c_i)P(d_j|c_i)}{P(d_j)}$$

Figura 14: Formula Naive Bayes

Dado que la probabilidad de cada documento $P(d_j)$ no aporta información para la clasificación, el término suele omitirse. La probabilidad de un documento dada la clase suele asumirse como la probabilidad conjunta de los términos que aparecen en dichos documentos dada la clase y se calculan como:

$$P(d_j|c_i) = \prod_{t=1}^{|V|} P(w_t|c_i)$$

Figura 15: Formula Multinomial Bayes

Este algoritmo también se usará en la fase 2 para el análisis del sentimiento.

3.3.2.3 Support Vector Machine (SVM)

Las Máquinas de Vectores Soporte, creadas por Vladimir Vapnik, constituyen un método basado en aprendizaje para la resolución de problemas de clasificación y regresión. En ambos casos, esta resolución se basa en una primera fase de entrenamiento (donde se les informa con múltiples ejemplos ya resueltos, en forma de pares (problema, solución)) y una segunda fase de uso para la resolución de problemas. En ella, las SVM se convierten en una “caja negra” que proporciona una respuesta (salida) a un problema dado (entrada) [28].

Dado un conjunto de puntos, subconjunto de un conjunto mayor (espacio), en el que cada uno de ellos pertenece a una de dos posibles categorías, un algoritmo basado en SVM construye un modelo capaz de predecir si un punto nuevo (cuya categoría desconocemos) pertenece a una categoría o a la otra.

Como en la mayoría de los métodos de clasificación supervisada, los datos de entrada (los puntos) son vistos como un vector p -dimensional (una lista ordenada de p números). La SVM busca un hiperplano que separe de forma óptima a los puntos de una clase de la de otra, que eventualmente han podido ser previamente proyectados a un espacio de dimensionalidad superior. En la siguiente figura se puede observar ejemplos de como divide el espacio del conjunto de datos mas famoso del mundo en el ámbito de la inteligencia artificial (iris dataset) donde p sería tres.

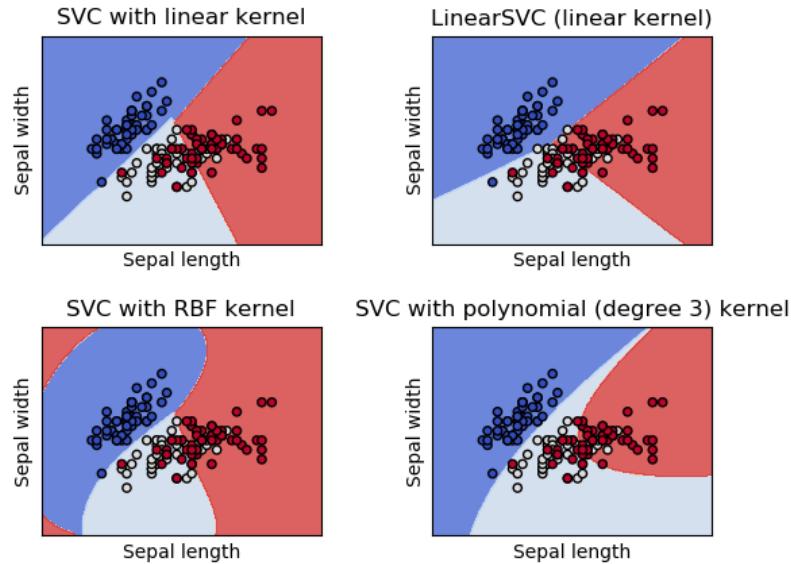


Figura 16: Ejemplo SVM [28] con diferentes kernels.

En ese concepto de "separación óptima." donde reside la característica fundamental de las SVM: este tipo de algoritmos buscan el hiperplano que tenga la máxima distancia (margen) con los puntos que estén más cerca de él mismo. Por eso también a veces se les conoce a las SVM como clasificadores de margen máximo. De esta forma, los puntos del vector que son etiquetados con una categoría estarán a un lado del hiperplano y los casos que se encuentren en la otra categoría estarán al otro lado.

3.3.2.4 Random Forest Model

Se trata de un método desarrollado por Leo Breiman y Adele Cutler en 2001 . Este método utiliza un conjunto (o bosque) de árboles de decisión para predecir un valor de salida. Para la clasificación de un nuevo ejemplo, cada árbol en el ensamble produce un voto, es decir, toma el ejemplo como entrada y produce una salida indicando a que clase pertenece. La decisión final del ensamble se realiza a partir de la clase con mayor cantidad de votos.

La idea esencial del *bagging* es promediar muchos modelos ruidosos pero aproximadamente imparciales, y por tanto reducir la variación. Los árboles son los candidatos ideales para el *bagging*, dado que ellos pueden registrar estructuras de interacción compleja en los datos, y si crecen suficientemente profundo, tienen relativamente baja parcialidad. Producto de que los árboles son notoriamente ruidosos, ellos se benefician enormemente al promediar.

Cada árbol es construido usando el siguiente algoritmo [29]:

1. Sea N el número de casos de prueba, M es el número de variables en el clasificador.
2. Sea m el número de variables de entrada a ser usado para determinar la decisión en un nodo dado; m debe ser mucho menor que M
3. Elegir un conjunto de entrenamiento para este árbol y usar el resto de los casos de prueba para estimar el error.
4. Para cada nodo del árbol, elegir aleatoriamente m variables en las cuales basar la decisión. Calcular la mejor partición del conjunto de entrenamiento a partir de las m variables.

Para la predicción un nuevo caso es empujado hacia abajo por el árbol. Luego se le asigna la etiqueta del nodo terminal donde termina. Este proceso es iterado por todos los árboles en el ensamblado, y la etiqueta que obtenga la mayor cantidad de incidencias es reportada como la predicción.

3.3.2.5 XGBoost (Xtreme Gradient Boosting Model)

XGBoost significa eXtreme Gradient Boosting. Es el algoritmo que ha estado dominando recientemente los problemas Machine learning y las competiciones de Kaggle con datos estructurados o tabulares. XGBoost es una implementación de árboles de decisión con Gradient boosting diseñada para minimizar la velocidad de ejecución y maximizar el rendimiento. Posee una interface para varios lenguajes de programación, entre los que se incluyen Python, R, Julia y Scala.

XGBoost es una implementación de código abierto popular y eficiente del algoritmo de árboles aumentados. La potenciación de gradientes es un algoritmo de aprendizaje supervisado que intenta predecir de forma apropiada una variable de destino mediante la combinación de estimaciones de un conjunto de modelos más simples y más débiles. [30].

Input : Data set \mathcal{D} .
A loss function L .
The number of iterations M .
The learning rate η .
The number of terminal nodes T

1 Initialize $\hat{f}^{(0)}(x) = \hat{f}_0(x) = \hat{\theta}_0 = \arg \min_{\theta} \sum_{i=1}^n L(y_i, \theta)$;

2 for $m = 1, 2, \dots, M$ **do**

3 $\hat{g}_m(x_i) = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} \right]_{f(x)=\hat{f}^{(m-1)}(x)}$;

4 $\hat{h}_m(x_i) = \left[\frac{\partial^2 L(y_i, f(x_i))}{\partial f(x_i)^2} \right]_{f(x)=\hat{f}^{(m-1)}(x)}$;

5 Determine the structure $\{\hat{R}_{jm}\}_{j=1}^T$ by selecting splits which maximize
 $Gain = \frac{1}{2} \left[\frac{G_L^2}{H_L} + \frac{G_R^2}{H_R} - \frac{G_{jm}^2}{H_{jm}} \right]$;

6 Determine the leaf weights $\{\hat{w}_{jm}\}_{j=1}^T$ for the learnt structure by
 $\hat{w}_{jm} = -\frac{G_{jm}}{H_{jm}}$;

7 $\hat{f}_m(x) = \eta \sum_{j=1}^T \hat{w}_{jm} I(x \in \hat{R}_{jm})$;

8 $\hat{f}^{(m)}(x) = \hat{f}^{(m-1)}(x) + \hat{f}_m(x)$;

9 end

Output: $\hat{f}(x) \equiv \hat{f}^{(M)}(x) = \sum_{m=0}^M \hat{f}_m(x)$

Figura 17: Ejemplo SVM [30]

Internamente, XGBoost representa todos los problemas como un caso de modelado predictivo de regresión que sólo toma valores numéricos como entrada. Si nuestros datos están en un formato diferente, primero vamos a tener que transformarlos para poder

hacer uso de todo el poder de esta librería. El hecho de trabajar sólo con datos numéricos es lo que hace que esta librería sea tan eficiente.

3.3.2.6 Diccionario

Se ha creado un algoritmo propio muy sencillo y con el cual se han tenido muy buenos resultados. Recordemos que tenemos tres diccionarios por cada uno de los deportes (para la parte de clasificación de texto según la temática), uno con las palabras clave donde si se encuentra esa palabra ya sabremos con bastante seguridad que se trata de ese deporte, uno con palabras secundarias con el cual el que más repeticiones de palabras tenga será el deporte del cual se estará hablando y un diccionario de palabras excluyentes para poder romper posibles empates en cuanto palabras secundarias.

3.4. Fase 2 - Análisis del sentimiento

3.4.1. Concepto

En esta segunda fase se quiere analizar el comportamiento del usuario para analizar si es muy critico o si solo tiene buenas palabras para su equipo como seria el ejemplo de un aficionado extremista del Sevilla CF o el Real Betis Balompié donde su rivalidad es histórica y los cuales siempre tienen buenas palabras para su equipo y malas formas e incluso insultos hacia la otra afición.

Para la realización de esta fase y poder analizar el sentimiento de los últimos tweets del usuario se hará uso de diferentes librerías como *VaderSentiment*, *TextBlob* y *NLTK* para definir si su sentimiento es positivo, negativo o neutral y para dar un resultado más detallado y poder determinar si el sentimiento es de alegría, miedo, sorpresa y amor entre otros se utilizará un diccionario propio con palabras que definen cada uno de los sentimientos.

El primer estudio se hace para dar un resultado más preciso ya que si sale un sentimiento negativo pero una segunda clasificación de alegría sabremos que algo está mal y por otro lado si da que el sentimiento es negativo se buscará antes sentimientos como enfado o tristeza a los de amor y alegría.

3.4.2. Software

Para esta frase se utiliza la frase entera sin modificar dado que las librerías *VADER*, *Sentiment* y *Textblob* ya las tratan internamente. Con esto ya sabremos si la frase es positiva, negativa o neutral [31].

Una vez sabemos si es positiva, negativa o neutral se utilizará la frase modificada por nosotros (sin las palabras que nosotros mismos hemos llamado palabras vacías) y se aplicará un algoritmo con el diccionario propio para saber si los tweets del usuario hablan de alegría, amor, enfado, miedo, sorpresa o tristeza.

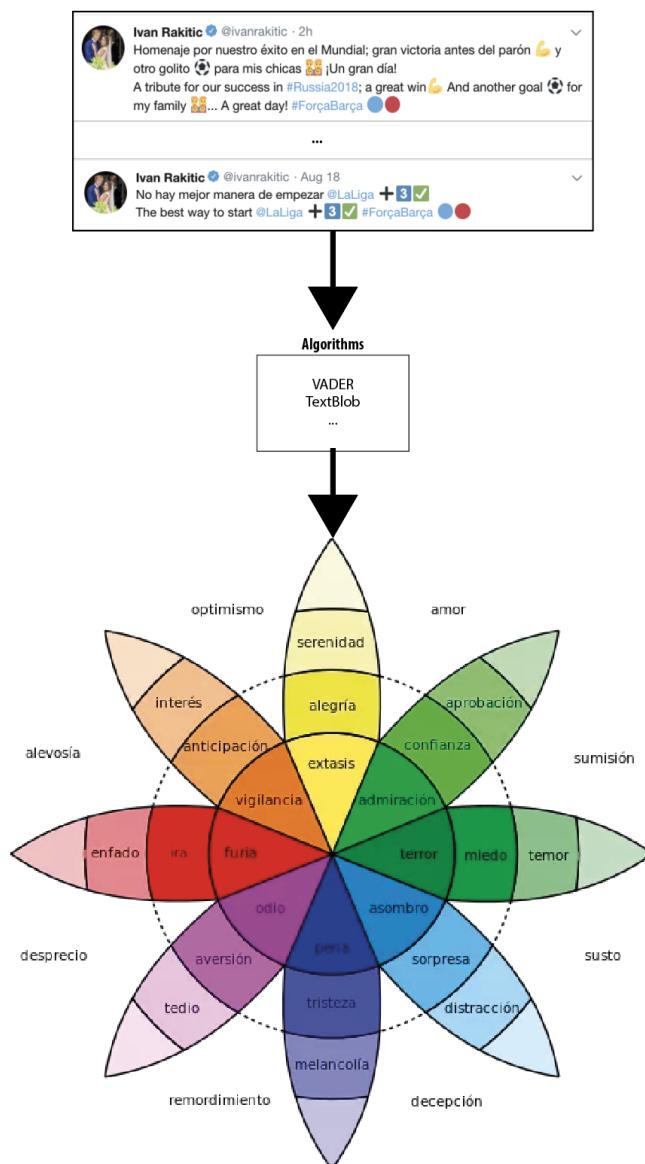


Figura 18: Diagrama fase 2 análisis del sentimiento

3.4.2.1 VADER Sentiment 3.3

VADER (Valence Aware Dictionary and sEntiment Reasoner) es un léxico y una biblioteca de análisis de sentimiento basada en reglas. Sirve como herramienta de análisis de sentimientos basada en reglas que está específicamente adaptada a los sentimientos expresados en las redes sociales.

Es completamente de código abierto bajo la [Licencia MIT]. VADER puede incluir el sentimiento de los emoticonos (por ejemplo, :-)), los acrónimos relacionados con el sentimiento (por ejemplo, LOL) y la jerga (por ejemplo, meh).

El método `vaderSentiment()` devuelve los valores para representar la cantidad de sentimiento negativo, positivo y neutral y también calcula el valor de sentimiento compuesto como un valor con signo para indicar la polaridad del sentimiento general.

A continuación se muestra un ejemplo de utilización de VADER:

```
sentences = ["Amo a Leo Messi es el mejor del mundo <3", # positive sentence example
            "Como es posible que me guste tanto Messi? Amor eterno!", # punctuation emphasis handled correctly (sentiment intensity adjusted)
            "Cristiano esta sobrevalorado, no es buen jugador y nunca lo fue", # negation sentence example
            "El golf no esta mal, meh.", # positive sentence
            "El golf es un deporte muy aburrido que no me gusta nada.", # negated negative sentence
            "No se si me gusta menos Cristiano Ronaldo o es peor el golf", # mixed negation sentence
            "Hoy jugaron fatal, no entiendo que apso lol", # mixed sentiment example with slang and contrastive conjunction "but"
            "Make sure you :) or :D today!", # emoticons handled
            "Si viese a mi amor Leo Messi 🌹 solo 🥺 no hay palabras😊", # emojis handled
            "Los lakers no estan mal del todo" , # Capitalized negation
            "Hola" # Neutral
            ]
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
analyzer = SentimentIntensityAnalyzer()
for sentence in sentences:
    vs = analyzer.polarity_scores(sentence)
    print("{:-<70} {}".format(sentence, str(vs)))
```

Figura 19: Código ejemplo utilización VADER

Amo a Leo Messi es el mejor del mundo <3-----	{'neg': 0.0, 'neu': 0.734, 'pos': 0.266, 'compound': 0.4404}
Como es posible que me guste tanto Messi? Amor eterno!-----	{'neg': 0.0, 'neu': 0.677, 'pos': 0.323, 'compound': 0.6476}
Cristiano esta sobrevalorado, no es buen jugador y nunca lo fue-----	{'neg': 0.196, 'neu': 0.804, 'pos': 0.0, 'compound': -0.296}
El golf no esta mal, meh.-----	{'neg': 0.467, 'neu': 0.533, 'pos': 0.0, 'compound': -0.3612}
El golf es un deporte muy aburrido que no me gusta nada.-----	{'neg': 0.167, 'neu': 0.833, 'pos': 0.0, 'compound': -0.296}
No se si me gusta menos Cristiano Ronaldo o es peor el golf-----	{'neg': 0.167, 'neu': 0.833, 'pos': 0.0, 'compound': -0.296}
Hoy jugaron fatal, no entiendo que apso lol-----	{'neg': 0.422, 'neu': 0.37, 'pos': 0.207, 'compound': -0.4404}
Make sure you :) or :D today!-----	{'neg': 0.0, 'neu': 0.294, 'pos': 0.706, 'compound': 0.8633}
Si viese a mi amor Leo Messi 🌹 solo 🥺 no hay palabras😊-----	{'neg': 0.105, 'neu': 0.571, 'pos': 0.324, 'compound': 0.6808}
Los lakers no estan mal del todo-----	{'neg': 0.268, 'neu': 0.732, 'pos': 0.0, 'compound': -0.296}
Hola-----	{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}

Figura 20: Output ejemplo utilización VADER

3.4.2.2 TextBlob 0.15.1

TextBlob es una librería de Python (2 y 3) para procesar datos textuales. Proporciona una API simple para adentrarse en tareas comunes de procesamiento del lenguaje natural (NLP), como etiquetado parcial, extracción de frase nominal, análisis de sentimiento, clasificación y traducción [32].

En esta fase se ha utilizado esta librería para traducir tweets de otros idiomas y para la clasificación del sentimiento se escogió esta librería porque que ofrece una API simple para acceder a sus métodos y realizar tareas NLP básicas.

3.4.2.3 NLTK 3.2.2

NLTK (Natural Language Toolkit) es una plataforma líder para construir programas de Python para trabajar con datos de texto natural. Proporciona interfaces fáciles de usar a más de 50 recursos léxicos como WordNet, junto con un conjunto de librerías de procesamiento de texto para clasificación, tokenización, derivación, etiquetado, análisis y razonamiento semántico, envoltorios para bibliotecas de PNL de fuerza industrial, y un foro de discusión activo [33].

Contiene una completa documentación (API). Está disponible para Windows, Mac OS X y Linux. NLTK es un proyecto gratuito y de código abierto que ha sido llamado “una herramienta maravillosa para enseñar y trabajar en lingüística computacional usando Python” y “una biblioteca increíble para jugar con el lenguaje natural”.

3.4.2.4 Diccionario

Para la implementación de este algoritmo, a diferencia de la fase anterior, solo tenemos un diccionario propio por cada una de las palabras. El algoritmo se basa simplemente en averiguar si las palabras escritas en el tweet se encuentran o no en alguno de los diccionarios implementados para cada uno de los sentimientos. Gracias al algoritmo VADER sentimetal ya podemos hacer un primer descarte de sentimientos.

3.5. Fase 3 - Resultado del análisis

Esta última fase es donde se encuentra el cerebro de la aplicación, tendrá que recopilar la información anterior y decidir si la persona ha mentido o no. Se trata de la fase más complicada y que por ello conlleva más problemas. Si en las fases anteriores se tenía dudas se puede imaginar que si sumas un grado fiabilidad alto pero lejano al 100 % como nos ocurre con la clasificación de texto según la temática con una fiabilidad cercana al 50 % como tenemos en el análisis del sentimiento, lo que se obtiene es una fiabilidad aun menor y si a ello le sumas hipótesis propias que no tienen un grado de fiabilidad definido, lo que se acaba obteniendo es un *cocktail* que habrá que saber manejar con cuidado.

Las hipótesis mas relevantes a tener en cuenta son:

1. Una persona que opina de todo es más propensa a mentir aunque sea por desconocimiento de las cosas a una persona que opina sobre uno o dos deportes donde igual es periodista, jugador de ese deporte o un aficionado interesado profundamente a ese deporte.
2. Un fanático o persona que habla sobre ese deporte y esta enfadado insulta y habla mal de los jugadores sin apenas pensar y no tiene por qué ser verdad al igual que cuando escribe con miedo espera que ese texto escrito por él mismo no sea verdad.
3. Contrastar si el perfil que lo escribe es una cuenta verificada o no, dado que las cuentas verificadas suelen ser de gente pública y por lo tanto, si cuentan mentiras, tendrán un castigo social más importante que si lo escribe alguien cuyo nombre no figura en Twitter.
4. Si la persona que lo escribe es alguien importante del deporte suele tener muchos *retweets* y por lo tanto numerosos tweets parecidos al suyo por lo tanto, si se busca ese *hashtag* se deberían tener comentarios similares.

5. Los textos con los emoticonos;



entre otros, suelen ser los escritos con hironía y por lo tanto, los más difíciles de clasificar.

6. Si escribes mal de un equipo o jugador o deportista no pertenece a tu equipo o en el caso de las olimpiadas a tu país, no habría que hacer mucho caso de lo escrito dado que se escribirá en parte desde el rencor y so por lo contrario escribes bien sobre el habría que intentar detectar la hironía porque a no ser que haya hecho realmente el mejor partido o carrera de su vida, seguramente se habrá escrito una mentira.

Finalmente este proyecto pretende encontrar tres tipos de trolls:

1. Los mentirosos: Aquellos usuarios que digan mentiras en Twitter.
2. Los furiosos:.. Aquellos perfiles que siempre estén poniendo tweets de una forma furiosa o hiriente.
3. Los *spammers*: Aquellos tuiteros que continuamente dicen lo mismo.

4. Experimentación

Una vez explicadas las tecnologías usadas y el marco del trabajo, se procederá a explicar detalladamente como se ha llegado a la resolución del trabajo detallando los procedimientos seguidos y justificando la elección de las librerías, algoritmos y técnicas usadas junto a los resultados obtenidos en cada una de las fases del proyecto. El proyecto se divide en 4 fases donde todas son dependientes de la fase 0 y la fase 3 de las tres anteriores. La 1 y 2 no dependen entre si dado que son algoritmos que buscan cosas diferentes, pero sí que necesitan toda la información extraída en la fase 0 para analizarla y enviar los resultados a la fase 3 donde se trataran y se decidirá si ese tweet contiene o no una mentira.

4.1. Problemática lenguaje natural

El procesamiento del lenguaje natural (PLN o NLP en sus siglas anglosajonas) es una rama de la Inteligencia Artificial encargada de estudiar métodos de comunicación entre máquina y hombre a través del lenguaje natural, es decir, el lenguaje empleado de forma habitual en una conversación escrita u oral entre personas.

El lenguaje natural presenta muchas características que lo hacen un verdadero reto para las ciencias de la computación. Como suele pasar en el mundo de la Inteligencia Artificial, una tarea que puede parecer para una máquina. Aspectos como la ambigüedad, la espontaneidad, la falta de fluidez, las referencias y las abreviaturas son difíciles de procesar.

4.1.1. Homonimia

La homonimia es la cualidad de dos palabras, de distinto origen y significado por evolución histórica, que tienen la misma forma, es decir, la misma pronunciación o la misma escritura [34].

Un ejemplo estaría en la palabra **vela**:

1. Acción de velar; cilindro de cera con una mecha para iluminar. (Ambos sentidos relacionados con el verbo velar).
2. Tela grande que aprovecha la fuerza del viento, especialmente en un barco.

En castellano de homonimias tenemos diferentes clases:

1. **Homónimos lexicales:** los que pertenecen a la misma categoría gramatical: onda y honda, botar y votar, haya y aya, ojear y hojear.
2. **Homónimos gramaticales:** los que no pertenecen a la misma categoría grammatical: cabe verbo y cabe preposición, o los que perteneciendo a la misma categoría grammatical se diferencian en alguna marca morfemática: el pez, la pez; el orden, la orden.
3. **Homónimos léxico-gramaticales:** los que se han formado a través de un cambio de funciones: poder (verbo) poder (sustantivo)
4. **Homónimos morfológicos:** cuando se producen diferentes formas de una sola palabra: decía primera y tercera personas del pretérito imperfecto de indicativo; o se dan formas correspondientes de palabras diferentes: fui (de ser e ir); ve (de ir y de ver), etc.

4.1.2. Polisemia

La polisemia, en lingüística, se presenta cuando una misma palabra o signo lingüístico tiene varias acepciones o significados. Una palabra polisémica es aquella que tiene dos o más significados que se relacionan entre sí [34].

Cabe resaltar que la polisemia puede surgir por diversos motivos. Por un lado, el vocabulario figurado produce polisemia por medio de las metáforas y las metonimias. Por ejemplo: los brazos de un río, las patas de una mesa. La especialización y el lenguaje técnico también atribuyen un significado específico a ciertos términos (como en el caso del ratón en la informática).

La influencia extranjera y las modificaciones de aplicación son otras condiciones que favorecen la polisemia: una muestra de esto es el vocablo botón que nació con la indumentaria y luego pasó a utilizarse también en los artefactos electrónicos.

Un ejemplo estaría en la palabra **cabo**:

1. (masculino) Punta de tierra que penetra en el mar.
2. (masculino/femenino) Escalafón militar.
3. (masculino) Cuerda en jerga náutica.

4.1.3. Anáfora y elipses

La anáfora se puede definir como la palabra o palabras que asumen el significado de una parte del discurso (texto) que ya se ha mencionado antes [35].

Un ejemplo de anáfora sería:

- Estoy de paso por Madrid. (Madrid) Es maravillosa.

Cuando el elemento sustituido aparece después del sustituto, hablamos de catáfora.

Un ejemplo de catáfora sería:

- Quería estar con Jesús; pero no lo he visto en toda la tarde.

Por otro lado tenemos las elipses que es la omisión o supresión de una o varias palabras que ya se habían mencionado antes o que se pueden presuponer o sobreentender. Esta construcción evita el uso de repeticiones innecesarias.

Un ejemplo de elipse sería:

- El concierto fue genial, la comida (fue) regular.

4.1.4. Sintaxis no normalizada

Otro problema que nos plantea el análisis de lenguaje natural es el hecho de que su estructura no está normalizada, es decir, a la hora de utilizar el lenguaje en español, no se utiliza una sintaxis concreta y bien definida, si no que a la hora de expresar una misma idea, ésta se puede estructurar de diversas maneras.

- Me encantan las mañanas de domingo.
- Las mañanas de domingo me encantan.

Ambas oraciones expresan la misma idea utilizando un orden diferente en las distintas palabras que las componen.

4.1.5. Contexto

El contexto es el conjunto de circunstancias (materiales o abstractas) que se producen alrededor de un hecho, o evento dado, que hacen que una palabra pueda tomar varios sentidos según el momento en el cual se utiliza.

Un claro ejemplo de la importancia del contexto es la ironía, a un suceso malo se puede contestar con otra frase 'mala' como sería el caso de; *¡Otra vez no!*, con una frase buena del estilo; *¡Qué bien!* donde estaríamos siendo irónicos o con una frase como; *¿Porqué a mí?*.

4.1.6. Otras circunstancias

A parte de toda esta serie de fenómenos que se dan en la lengua formal, hay que tener que cuenta otros aspectos que se dan en ambientes más coloquiales como son las redes sociales.

En la siguiente tabla se muestran algunos ejemplos de cómo un mismo mensaje puede tener formas diferentes según el usuario y el medio en el que lo escriba.

Texto de ejemplo	Explicación
Messi es el mejor jugador de la historia del fútbol.	Frase correcta
Mesi es el mejor jugador de la istoria del futbol;	Faltas de ortografía
MesSi es el MeJor juGAdor de la hIStoriA del fúTBol :)	Incluir mayúsculas y símbolos innecesarios
Messiiii es el mejor jugador de la historia del futboooooool	Añadir repetición de letras para darle mayor énfasis a la palabra
Mssi s l mjour jugadqr d la histria dl futbol	Lenguaje SMS.
¡Vamos equipo!	Breveedad y ambigüedad del texto

4.2. Fase 0 - Extracción de la información

Esta es la fase cero es donde se diseña el proyecto y por eso es la fase que más veces ha tenido que ser modificada para ajustarse a las necesidades reales del proyecto descubiertas en las siguientes fases a lo largo de la implementación de estas. No se llama fase cero solo porque los informáticos estamos acostumbrados a contar desde cero, sino que también no solo es una fase previa de almacenar datos en el proyecto, sino que a lo largo que ha ido evolucionando el proyecto se ha tenido que ir modificando código y estructuras alocadas en esta fase. Podríamos decir que a parte de una fase previa también es una fase paralela a todo el proyecto.

Antes de nada lo que se decidió es que dada la acotación que se hizo de solo buscar trolls en el ámbito de los deportes este proyecto sea fácilmente escalable así que se creó una clase *Utils* con clases para definir los temas por variables como nombre, lista de palabras, id, etc y otra clase donde se hace una relación de un id numérico creciente con el nombre del tema así pues, todos los algoritmos tienen un bucle y solo tienen que buscar por cada uno de los id's sin necesidad de modificar nada del código únicamente se necesita incluir el nuevo tema con su id correspondiente en esa clase. Esto permite añadir una capa de abstracción de los temas que hay a los algoritmos y diccionarios dado que tanto los algoritmos como los diccionarios utilizan esa lista de temas sin saber cuántos hay por lo tanto si añades un nuevo tema sin darse cuenta llenara el diccionario con ese nuevo tema y clasificara el texto con ese tema si ese tweet trata sobre ese nuevo tema.

A continuación se puede ver como es la función para obtener los deportes que se buscarán y que por su estructura tiene un muy fácil crecimiento de los temas:

```

| def get_data_id(data_name):
|     data ={ 'Futbol':      0,
|             'Baloncesto': 1,
|             'Golf':        2,
|             'Boxeo':       3,
|             'Judo':        4,
|             'Balonmano':   5,
|             'Tenis':       6,
|             'Ciclismo':    7,
|             'Atletismo':   8,
|             'Voleibol':    9,
|             'Rugby':       10,
|             'Motociclismo': 11,
|             'Futbol americano': 12,}

|     return data[data_name.lower()]

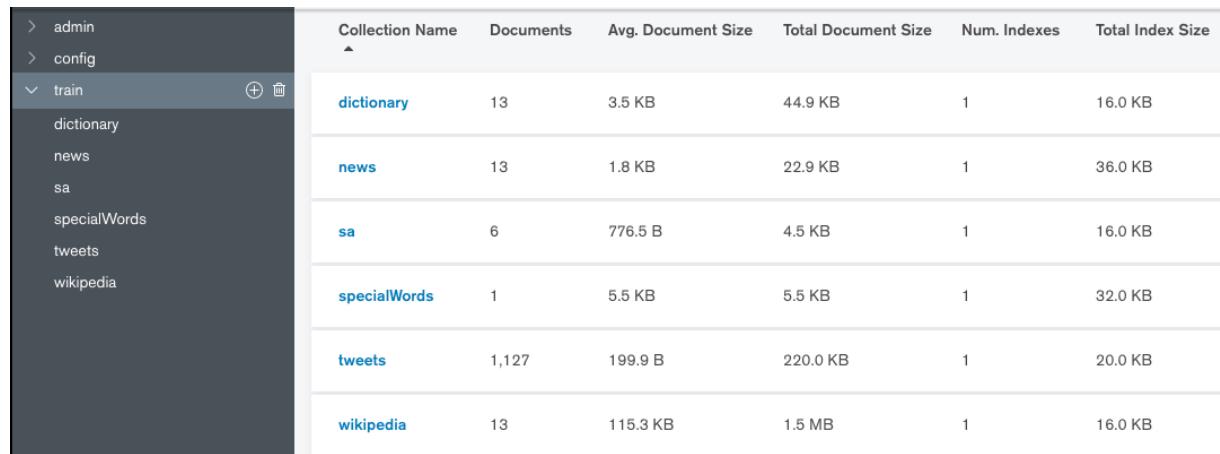
def get_data_name(data_id):
    data ={ 0: 'Futbol',
            1: 'Baloncesto',
            2: 'Golf',
            3: 'Boxeo',
            4: 'Judo',
            5: 'Balonmano',
            6: 'Tenis',
            7: 'Ciclismo',
            8: 'Atletismo',
            9: 'Voleibol',
            10: 'Rugby',
            11: 'Motociclismo',
            12: 'Futbol americano'}

    return data[data_id]

```

Figura 21: Función que devuelve el id o el nombre del tipo en esa posición.

La primera tarea en esta fase es la de guardar los diccionarios que no son más que una ontología propia y montar toda la base de datos en MongoDB dado que sin ellos no se puede hacer nada. Se decidió que la base de datos fuese en Mongo dado que es la idónea para guardar grandes cantidades de textos por su indexación. En esta base de datos se encuentran tweets de ejemplo, un diccionario con palabras que no aportan significado a las frases como es el caso de los artículos, un diccionario para cada deporte donde se encuentran palabras primarias, secundarias y excluyentes y por último un diccionario con palabras que relacionan directamente a cada uno de los sentimientos. En los apartados siguientes de experimentación en la fase uno y dos se explicará para que sirve cada uno de los diccionarios, en esta fase solo se usa el de palabras vacías.



	Collection Name	Documents	Avg. Document Size	Total Document Size	Num. Indexes	Total Index Size
	dictionary	13	3.5 KB	44.9 KB	1	16.0 KB
	news	13	1.8 KB	22.9 KB	1	36.0 KB
	sa	6	776.5 B	4.5 KB	1	16.0 KB
	specialWords	1	5.5 KB	5.5 KB	1	32.0 KB
	tweets	1,127	199.9 B	220.0 KB	1	20.0 KB
	wikipedia	13	115.3 KB	1.5 MB	1	16.0 KB

Figura 22: Base de datos del proyecto.

Solo recordar que este es el estado final de la base de datos a la cual se le han ido añadiendo campos y clases según se han necesitado. El primer diccionario que se creo fue el de las palabras vacías para poder excluir todas aquellas palabras que no nos aportaban nada y así tener unos mejores resultados. La primera opción fue buscar cada uno de los deportes en Wikipedia con una conexión a través de Python y de ese texto obtenido eliminar preposiciones, artículos y demás palabras encontradas en el fichero y guardar el número de repeticiones de cada una de las palabras que se encontraban en ese fichero de palabras vacías. Con esto conseguimos tener una lista de todas las palabras que se encuentran en Wikipedia sobre cada uno de los deportes y el número de veces que aparece. A partir de este primero diccionario se fue trabajando en paralelo creando algoritmos y afinando un diccionario propio más preciso donde se busco por Internet vocabulario de cada uno de los deportes, así como sus personas, equipos y campeonatos más relevantes. Con el diccionario de Wikipedia se tenían unos resultados algo pobres en las siguientes fases que se vieron mejorados gracias al diccionario escrito a mano. La diferencia principal a parte de que en Wikipedia no están todas las palabras de un deporte es que en Wikipedia solo almacenábamos las palabras más repetidas y el porcentaje con el que aparecían y con el diccionario propio conseguimos hilar más fino gracias a tener palabras principales que definen en el deporte, palabras secundarias que son las que se usarían en la jerga de ese deporte y palabras excluyentes que no se dirían nunca en ese deporte.

dictionary					
	_id ObjectId	text String	word_list Array	top_words Array	top_words_percentages Array
1	Sada0f69ab930e243ae5e07b	"el fútbol o futbol del ing"	[] 4943 elements	[] 303 elements	[] 303 elements
2	Sada0f6bab930e243ae5e07d	"el baloncesto basquetbol o l"	[] 4134 elements	[] 293 elements	[] 293 elements
3	Sada0f6dab930e243ae5e07f	"el golf es un deporte de pr"	[] 2864 elements	[] 196 elements	[] 196 elements
4	Sada0f6fab930e243ae5e081	"el boxeo del inglés boxing "	[] 6462 elements	[] 443 elements	[] 443 elements
5	Sada0f71ab930e243ae5e083	"el judo o yudo del japoné"	[] 3796 elements	[] 277 elements	[] 277 elements
6	Sada0f73ab930e243ae5e085	"el balonmano handball o hán"	[] 1948 elements	[] 144 elements	[] 144 elements
7	Sada0f76ab930e243ae5e087	"el tenis también llamado te"	[] 2477 elements	[] 168 elements	[] 168 elements
8	Sada0f77ab930e243ae5e089	"el ciclismo es un deporte e"	[] 1271 elements	[] 117 elements	[] 117 elements
9	Sada0f7aab930e243ae5e08b	"el atletismo es considerado"	[] 6005 elements	[] 409 elements	[] 409 elements
10	Sada0f7cab930e243ae5e08d	"el voleibol vóleibol volib"	[] 3014 elements	[] 200 elements	[] 200 elements
11	Sada0f7dab930e243ae5e08f	"el rugby es un deporte de"	[] 4060 elements	[] 276 elements	[] 276 elements
12	Sada0f7fab930e243ae5e091	"el motociclismo es el uso d"	[] 514 elements	[] 50 elements	[] 50 elements
13	Sada0f81ab930e243ae5e093	"el fútbol americano en ingl"	[] 4000 elements	[] 241 elements	[] 241 elements

Figura 23: Diccionario de los deportes guardados obtenidos por Wikipedia.

dictionary				
	_id ObjectId	type String	keyWords Array	secondaryWords Array
1	Sae7312cab930e84ed995cb3	"Futbol"	[] 84 elements	[] 247 elements
2	Sae7312dab930e84ed995cb5	"Baloncesto"	[] 78 elements	[] 91 elements
3	Sae7312dab930e84ed995cb7	"Golf"	[] 84 elements	[] 77 elements
4	Sae7312dab930e84ed995cb9	"Boxeo"	[] 82 elements	[] 72 elements
5	Sae7312dab930e84ed995ccb	"Judo"	[] 79 elements	[] 51 elements
6	Sae7312dab930e84ed995cbd	"Balonmano"	[] 78 elements	[] 68 elements
7	Sae7312dab930e84ed995cbf	"Tenis"	[] 84 elements	[] 77 elements
8	Sae7312dab930e84ed995cc1	"Ciclismo"	[] 84 elements	[] 69 elements
9	Sae7312dab930e84ed995cc3	"Atletismo"	[] 81 elements	[] 103 elements
10	Sae7312dab930e84ed995cc5	"Voleibol"	[] 78 elements	[] 62 elements
11	Sae7312dab930e84ed995cc7	"Rugby"	[] 83 elements	[] 83 elements
12	Sae7312dab930e84ed995cc9	"Motociclismo"	[] 83 elements	[] 81 elements
13	Sae7312dab930e84ed995ccb	"Futbol americano"	[] 81 elements	[] 110 elements

Figura 24: Diccionario de los deportes propio.

dictionary		
	_id ObjectId	type String
1	5b2bb04eab930e42aaaf926bc	"alegría"
2	5b2bb04eab930e42aaaf926be	"amor"
3	5b2bb04eab930e42aaaf926c0	"enfado"
4	5b2bb04eab930e42aaaf926c2	"miedo"
5	5b2bb04eab930e42aaaf926c4	"sorpresa"
6	5b2bb04eab930e42aaaf926c6	"tristeza"

Figura 25: Diccionario de los sentimientos propio.

Una vez obtenido el diccionario de los deportes tocaba ponerlo en práctica para ver cómo funcionaba así que se probaron frases escritas a mano y cuando empezamos a comprobar que los algoritmos funcionaban se probó la conexión que habíamos hecho previamente a Twitter con la librería Tweepy que proporciona una API muy sencilla para obtener información sobre usuarios, así como sus tweets y hacer búsquedas por hashtags. Estos tweets fueron almacenados en MongoDB con la finalidad de comprobar siempre los algoritmos con los mismos tweet dado que un estudio de *Designly*.¹⁶ dice que se escriben 3.935 tweets cada segundo así que nuestros resultados variaban en cada ejecución.

Una vez están las conexiones con MongoDB y Twitter creadas y se tiene toda la información guardada se vio que las consultas en las siguientes fases se hacían de forma lenta así que para mejorar tiempos de consulta se decidió guardar los diccionarios en un AVL Tree.

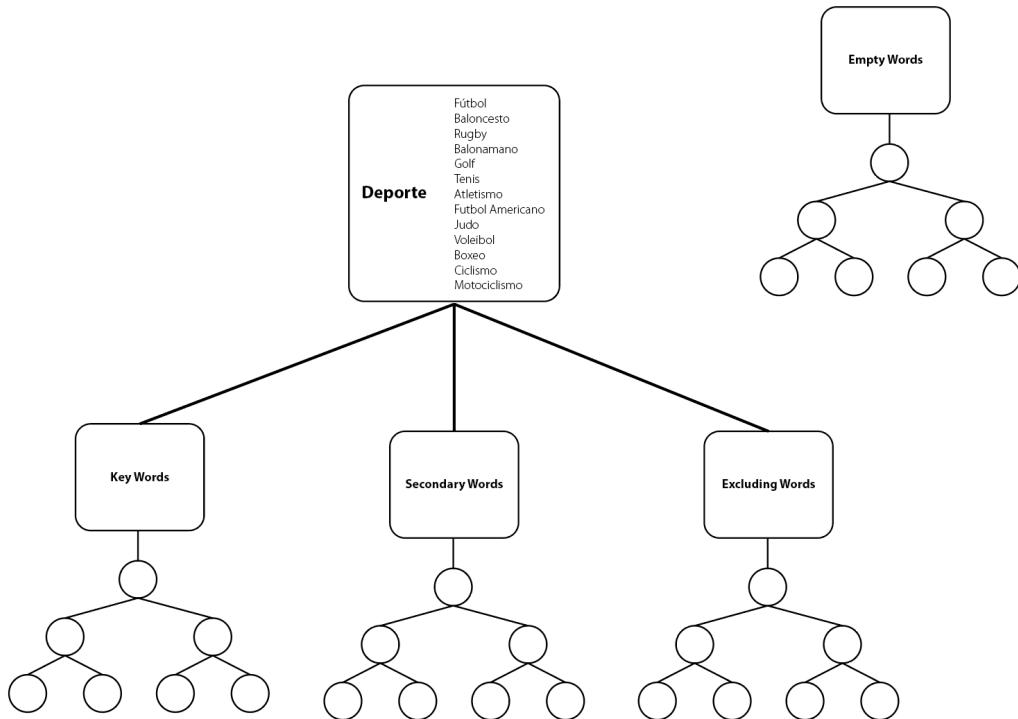


Figura 26: Esquema estructuración de los datos

¹⁶<https://www.infobae.com/2013/08/07/1500503-cuantos-tuits-se-escriben-segundo-el-mundo/>

Una vez el árbol esta creado y se tiene almacenado en él los diccionarios pertinentes se procede a tratar el tweet que se quiere analizar o en su defecto el texto introducido por el usuario. Cada tweet lo replicamos y guardamos el tweet original, aplicándole el *stemmer* para obtener la raíz de cada palabra y una ultima con el tweet sin las palabras vacías, todas las palabras separadas y en forma de *array*. Una vez teniendo el tweet original y sus variaciones se enviaran estas a las posteriores fases para analizarlas. Se quiso conservar en todo momento el texto original por si se necesitaba en algún algoritmo. El único problema que hubo con la librería de Tweepy es que solo proporciona tweets de cien en cien así que se tenía que ir guardando la fecha del último tweet y hacer peticiones siempre a partir de esa fecha hasta que tengamos el número de tweets que se querían.

Este análisis se aplica no solo al tweet a analizar sino que también recoge los últimos veinte tweets del usuario y si hizo algún retweet o mención se analizan los últimos diez tweets de ese otro usuario para tener un análisis más fiable, eso sí, se para en el segundo nivel, es decir solo se mira el usuario del retweet y como mucho si ese perfil tenía otro retweet se estudia ese usuario y no se adentrará a más perfiles.

A continuación, podemos observar dos ejemplos de como tratan los tweets y por lo tanto, de como reciben los textos las siguientes fases.

```
Example 1
Input: Jugando con mi sobrino nos dimos cuenta de lo divertido que es el escondite
Output stemmed: ['jug', 'con', 'mi', 'sobrin', 'nos', 'dim', 'cuent', 'de', 'lo', 'divert', 'que', 'es', 'el', 'escondit']
Output stemmed and without empty words: ['jug', 'sobrin', 'dim', 'cuent', 'divert', 'escondit']

Example 2
Input: Lionel Messi es el mejor jugador del mundo, su zurda es enviable
Output stemmed: ['lionel', 'messi', 'es', 'el', 'mejor', 'jugador', 'del', 'mund', 'su', 'zurd', 'es', 'envidi']
Output stemmed and without empty words: ['lionel', 'messi', 'jugador', 'mund', 'zurd', 'envidi']
```

Figura 27: Ejemplo de salida de la frase una vez tratada.

En la siguiente figura se pueden ver las clases implementadas más importantes:

```

class WikiData:
    def __init__(self, text, word_list, top_words, top_words_percentages, type_name):
        self.text = text
        self.word_list = word_list
        self.top_words = top_words
        self.top_words_percentages = top_words_percentages
        self.url = None
        self.type_name = type_name

    def getTopWordsString(self):
        text = ''
        for word in self.top_words:
            text += word + ' '
        return text

class Dictionary:
    def __init__(self, type, key_words, secondary_words, excluding_words):
        self.type_name = type
        self.key_words = key_words
        self.secondary_words = secondary_words
        self.excluding_words = excluding_words

class DictionaryTree:
    def __init__(self, type, key_words, secondary_words, excluding_words):
        self.type_name = type
        key_words_tree = Tree.AVLTree()
        self.key_words = key_words_tree.insert_array(key_words)
        secondary_words_tree = Tree.AVLTree()
        self.secondary_words = secondary_words_tree.insert_array(secondary_words)
        excluding_words_tree = Tree.AVLTree()
        self.excluding_words = excluding_words_tree.insert_array(excluding_words)

class Tweet:
    def __init__(self, text, length, date, source, likes, retweets, stemmed):
        self.text = text
        self.length = length
        self.date = date
        self.source = source
        self.likes = likes
        self.retweets = retweets
        self.sa = 'NEUTRAL'
        self.stemmed = stemmed
        self.theme = 'SPORT'

class Sentiment:
    def __init__(self, sentiment, words_list):
        self.sentiment = sentiment
        self.words_list = words_list

class User:
    def __init__(self, description, verified, followers_count):
        self.description = description
        self.verified = verified
        self.followers_count = followers_count

```

Figura 28: Clases propias más importantes.

En conclusión, es la encargada de crear y gestionar conexiones con MongoDB y Twitter así como de almacenar los datos obtenidos en esas conexiones y enviarlas a las siguientes fases para su posterior tratamiento.

4.3. Fase 1 - Clasificación de texto según temática

La tarea de esta fase es la de analizar cada uno de los tweets y decir cada uno de que tema hace referencia para finalmente saber sobre que X temas habla ese usuario donde X esta fijada en tres aunque en los ejemplos que se mostraran para hacerlo más claro solo se dice el tema sobre el que más habla.

Recordad que los temas sobre los que se centra el algoritmo son de los que hicimos diccionario:

- | | |
|----------------|----------------------|
| 0. Fútbol. | 7. Judo. |
| 1. Baloncesto. | 8. Voleibol. |
| 2. Rugby. | 9. Boxeo. |
| 3. Balonmano. | 10. Ciclismo |
| 4. Golf. | 11. Motociclismo. |
| 5. Tenis. | 12. Fútbol americano |
| 6. Atletismo | |

En la figura siguiente se muestra un esquema que representa la idea principal del funcionamiento de esta fase.

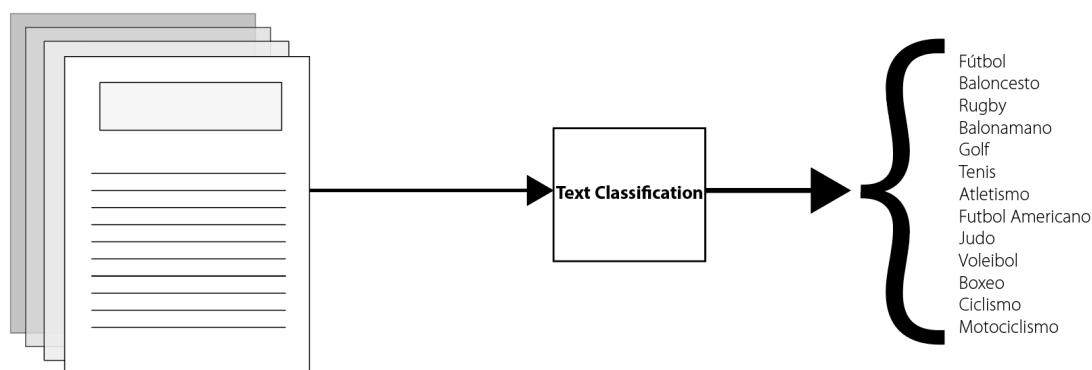


Figura 29: Idea principal del clasificador de texto

Para la clasificación de los Tweets según la temática lo primero que se hizo fue implementar unos algoritmos que se entrenaban mediante el mismo Twitter. Lo que se hizo fue pedirle a la fase cero que nos enviase X tweets donde el 33% de estos tweets se utilizaban para entrenar el algoritmo. X son el número de tweets totales que se tendrán para hacer las pruebas pero realmente se piden X' tweets donde X' corresponde a $(X/(N\acute{u}mero\ de\ Temas))$ por lo tanto, en nuestro caso, $X'=X/13$. Así pues se hicieron pruebas donde X tomaba el valor de: 13, 26, 52, 104, 208, 416, 832, 1.500, 3.000, 5.000 y 10.000. En el siguiente gráfico podemos observar los resultados de la accuracy de los algoritmos con los valores anteriores.

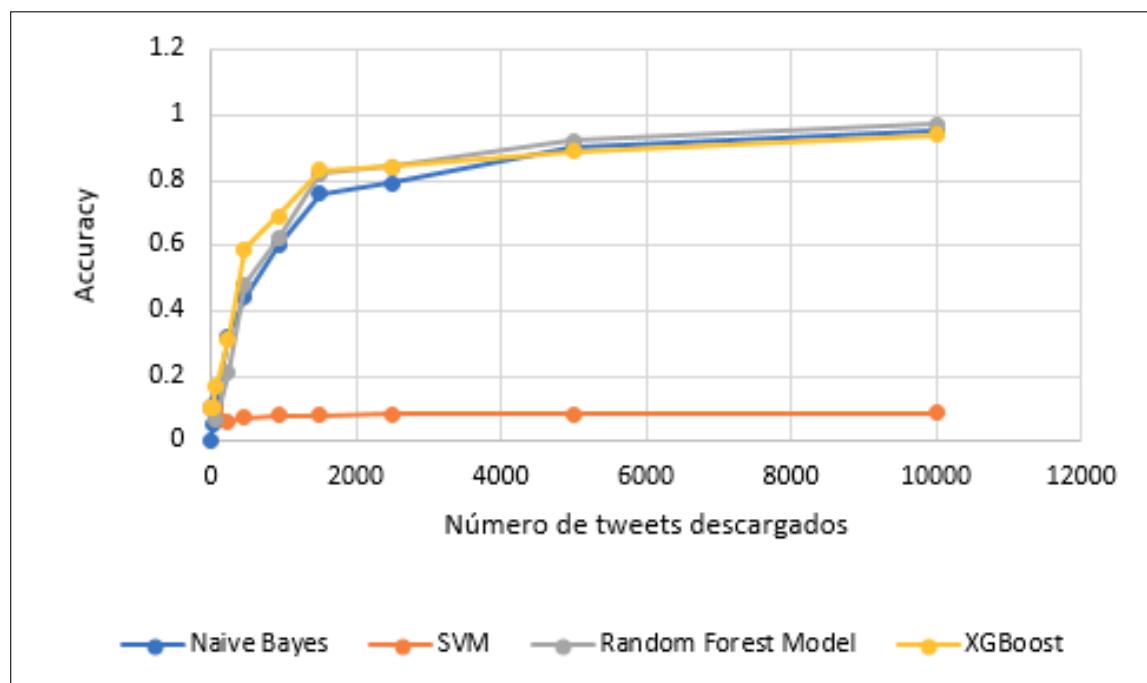


Figura 30: Resultados obtenidos en clasificación de texto sin diccionario.

En el diagrama anterior se puede observar como el peor algoritmo es el SVM dado que nunca se puede obtener un accuracy mayor al 0,1. Esto viene dado a que tiene una gran demanda de buenos tokenizadores (filtros) pero, el diccionario del conjunto de datos siempre tiene tokens sucios o ruido porque son los tweets de otros usuarios, este funcionaría mejor si se aplicará el diccionario creado ya que ese diccionario contiene palabras que definen cada deporte por separado y no verbos o adjetivos que se pueden usar indistintamente en cada uno de los deportes. También se puede apreciar que todos los algoritmos coinciden en que, por un juego de datos menor a 1000 tweets, ninguno obtiene un accuracy mayor al 0,5. Los algoritmos que obtienen mejor

accuracy con menos de 1500 tweets es XGBoost y y a partir de los 2000 tweets los que devuelven mejor accuracy es Random Forest Model acercándose al 0,9 a partir de los 5000 tweets donde de estos 5000 solo 1560 se están utilizando para entrenar el algoritmo.

Los resultados anteriores se podrían dar por satisfactorios y acabar la investigación, pero primero habría que analizarlos más detalladamente y darse cuenta de que tienen algo de trampa. Los resultados dados así son muy gratificantes, solamente descargas 2000 tweets y obtienes un accuracy del 0,8 donde cualquier analista te lo compraría y daría por cerrado el proyecto. Bien, no todo es tan fácil, descargar 2000 tweets es más difícil de lo que parece dado que la API que proporciona Twitter para Python solo devuelve un máximo de 100 tweets por petición, es decir, para pedir 2000 tweets hay que hacer 20 consultas las cuales no son rápidas y tiene que obtener la fecha del tweet número cien para en la siguiente petición pedir tweets a partir de esa fecha y así hasta que logres alcanzar los 2000 tweets. Esto tampoco parece un gran problema, en una media de 10 segundos consigues los tweets que quieras, pero si a esto le sumas que solo puedes hacer 100 peticiones cada media hora, tienes el resultado que cada 30 minutos solo puedes ejecutar tu algoritmo cinco veces.

Otro problema que se presenta es la proximidad temporal de los tweets. Esto se refiere a que cogemos los últimos tweets que se han escrito en Twitter y los usuarios normalmente comentan las últimas novedades y lo sucedido en las últimas horas y esto es una ventaja dado que usualmente compararemos tweets parecidos al del usuario, pero también puede pasar que un usuario hablé sobre lo que pasó ayer u otro día anterior y al hacer la búsqueda con estos algoritmos ninguno se parezca y lo clasifiquemos de forma errónea.

Para solucionar estos problemas más las restricciones de tiempos de los algoritmos se decidió utilizar Naive Bayes con un dataset de 700 tweets donde se consigue un accuracy del 0,4-0,5 y es el más rápido en cuanto a tiempo se refiere gracias a la sencillez del algoritmo. Con los resultados obtenidos se hace una media de que deporte puede tratarse entre los dos algoritmos y se acaba de afinar con un tercer algoritmo el cual es propio y se ejecuta mediante un diccionario también propio.

TODO: MATRIZ DE CONFUSION NAIVE BAYES CON 700

Este diccionario de cada deporte esta almacenado en un árbol AVL Tree el cual es una estructura de datos ordenada y los costes temporales de búsqueda son $O(\log(n))$. Otra ventaja que tiene este árbol es el poder clasificar por palabras compuestas, es decir que por ejemplo si buscas la falta de baloncesto 'campo atrás', gracias a la búsqueda que se ha implementado en este árbol, se encontrará. En la siguiente figura se puede observar que se tarda 1,32 segundos en tratar sin el árbol y 0,33 analizando con el árbol AVL implementado.

```
..... Tweet .....
lionel messi es un autentico crack, tiene un disparo enviable, es un autentico killer, le amo
Tweet a tratar: ['lionel', 'messi', 'autentico', 'crack', 'disparo', 'enviable', 'autentico', 'killer', 'amo']

----- Dictionary -----
Segun palabras primarias: Futbol con las palabras repetidas: ['messi']
Segun palabras secundarias: Futbol con las palabras repetidas: ['killer', 'disparo']
Aplicando la formula final: Futbol con 1.5 puntos
Ha tardado: 1.3204758167266846 seg

----- Dictionary&Tree -----
Segun primary words: Futbol ['lionel', 'messi']
Segun secondary words: Futbol ['disparo', 'killer']
Ha tardo: 0.331298828125 seg
```

Figura 31: Comparativa entre guardar diccionario en una *array* o en un árbol AVL.

Una vez este algoritmo recibe el tweet previamente tratado en la fase 0, es decir que tengo el tweet separado por palabras y sin artículos ni preposiciones, actuará el algoritmo. Este algoritmo principalmente lo que hace es buscar apariciones de las palabras del tweet en los arboles creados, uno por cada deporte y además uno que contiene palabras primarias las cuales si se encuentra sumara 1 punto a ese deporte, palabras secundarias las cuales si las encuentra sumara 0,25 puntos sobre ese deporte y palabras excluyentes las cuales si se encuentran se restaran 1,5 puntos en ese deporte. Una vez ya no hay más palabras obtenemos los dos deportes con mayor puntuación y se contrastan con los obtenidos en la evaluación anterior del algoritmo Naive Bayes y .

En la siguiente figura se pueden observar cuatro ejemplos de como actúa este algoritmo descrito anteriormente que hace uso de los diccionarios de cada uno de los deportes.

```
..... Tweet .....  

El portero la cogió fuera del área, debio ser falta y amarilla  

Tweet a tratar: ['portero', 'cogió', 'área', 'debio', 'falta', 'amarilla']  

----- Dictionary&Tree -----  

Ninguna key word encontrada  

Segun palabras secundarias: Futbol con las palabras repetidas: ['amarilla', 'falta', 'portero']  

Aplicando la formula final: Futbol con 0.75 puntos  

Ha tardado: 0.935478925704956 seg  

..... Tweet .....  

Ese jab le dejó besando la lona y a Myweather eufórico  

Tweet a tratar: ['jab', 'besando', 'lona', 'myweather', 'eufórico']  

----- Dictionary&Tree -----  

Segun palabras primarias: Boxeo con las palabras repetidas: ['lona']  

Segun palabras secundarias: Boxeo con las palabras repetidas: ['jab']  

Aplicando la formula final: Boxeo con 1.25 puntos  

Ha tardado: 0.7490377426147461 seg  

..... Tweet .....  

Canastó de tres en el ultimo minuto, los aficionados de los Lakers no podian ocultar su furia  

Tweet a tratar: ['canastó', 'ultimo', 'minuto', 'aficionados', 'lakers', 'podian', 'ocultar', 'furia']  

----- Dictionary&Tree -----  

Segun palabras primarias: Baloncesto con las palabras repetidas: ['canastó', 'lakers']  

Ninguna secondary word encontrada  

Aplicando la formula final: Baloncesto con 2.0 puntos  

Ha tardado: 1.265803575515747 seg  

..... Tweet .....  

Ese flanker fué a por uvas  

Tweet a tratar: ['flanker', 'fué', 'uvas']  

----- Dictionary&Tree -----  

Ninguna key word encontrada  

Segun palabras secundarias: Rugby con las palabras repetidas: ['flanker']  

Aplicando la formula final: Rugby con 0.25 puntos  

Ha tardado: 0.48641204833984375 seg  

..... Tweet .....  

Ese tio no vale nada  

Tweet a tratar: ['tio', 'vale', 'nada']  

----- Dictionary&Tree -----  

Ninguna key word encontrada  

Ninguna secondary word encontrada  

El texto no se pudo clasificar.  

Ha tardado: 0.47912073135375977 seg
```

Figura 32: Ejemplos de ejecución del algoritmo por diccionario almacenado en un árbol AVL en análisis por temática.

También se hicieron pruebas con el algoritmo de Naive Bayes entrenándolo, no solo con tweets, sino que también con las palabras que se encontraban en el diccionario como palabras primarias. Esto mejoró los resultados respecto a entrenarlo solo con tweets, pero no a los resultados obtenidos con el diccionario anterior. Por último, se intentó entrenarlo también con las palabras secundarias pero los resultados fueron parecidos.

Debido al uso del castellano como lenguaje a analizar todos los análisis se han complicado, un ejemplo de ello es el intentar de utilizar un *stemmer* para obtener la raíces de las palabras ya que, en castellano, es un proceso muy complicado por la riqueza del mismo. En la siguiente figura se puede observar un ejemplo de este problema dónde confunde la preposición para que coge como raíz 'par' con el verbo parar dado que en el diccionario de balonmano y fútbol hay la palabra parada y la define también con la raíz 'par'.

```
..... Tweet .....
Ese tio no vale para nada
Tweet a tratar: ['tio', 'vale', 'para', 'nada']

----- Dictionary&Tree -----
Ninguna key word encontrada
Segun palabras secundarias: Balonmano con las palabras repetidas: ['para']
Aplicando la formula final: Balonmano con 0.25 puntos
Ha tardado: 0.6524112224578857 seg
```

Figura 33: Ejemplos de error en el uso de un *stemmer* en clasificación de texto.

Para concluir, recordar que se usan los algoritmos NaiveBayes junto al diccionario propio y no solo que estos algoritmos se aplican no solo a un tweet sino, a todos últimos 20 tweets del usuario y los 10 de los perfiles retuiteados si los hay. Una vez hecho esto se envía a la fase tres los resultados y está queda a la espera de tener los resultados de la fase dos.

4.4. Fase 2 - Análisis del sentimiento

El análisis de sentimiento de textos en las redes sociales es el proceso que determina el tono emocional que hay detrás de unas palabras determinadas, si una frase contiene una opinión positiva o negativa sobre un producto, marca, institución, organización, empresa, evento o persona.

Los sentimientos se clasifican en positivos, negativos o neutros. Sin embargo, el lenguaje natural es complejo y ambiguo por lo que enseñar a una máquina a que analice los diferentes matices gramaticales, variaciones culturales, jergas, expresiones coloquiales o a distinguir faltas de ortografía, la sinonimia o la polisemia dentro de un contexto que determina el tono de la conversación es francamente difícil. Así, por ejemplo, ante un comentario sarcástico, la máquina tomaría la frase como algo positivo en vez de algo negativo o expresiones como “LOL, OMG, estuvo geeeeeeeeniaaaaaaaal” son dificilísimas de procesar.

Como los algoritmos que usaran en esta fase se aplicarán en textos (o tweets) escritos en castellano, los resultados son bastante pobres dado que no se tiene un *dataset* tan amplio como en la anterior fase. Además las posibilidades de decir un adjetivo que describa un sentimiento son mucho mayores dado qu en un deporte tienes palabras clave como 'jab' o 'lona' en boxeo, 'gol' y 'fuera de juego' en fútbol, etc. En cambio en los sentimientos una palabra como por ejemplo la palabra 'amor' se puede dar con diferentes variantes como 'amo', 'amada', 'amor', 'amante', etc. Por ese motivo se ha decidido hacer uso de un *stemmer* para encontrar la raíz de las palabras pero, como pasaba en la fase anterior, encontrar la raíz de una palabra en castellano es mucho más difícil y ninguno funciona del todo bien aunque ayudan en algunos casos como en no diferenciar masculino y femenino.

```
Palabras originales: ['amor', 'amar', 'amo', 'amada', 'amado', 'amante', 'avaría', 'enamorar', 'desamor', 'amorio']
Palabras stemmed:   ['amor', 'amar', 'amo', 'amad', 'amad', 'amant', 'avari', 'enamor', 'desamor', 'amori']
```

Figura 34: Ejemplos de error en el uso de un *stemmer* con derivaciones de la palabra amor.

Esta fase servirá al proyecto para acabar de afinar si es o no mentira y para detectar a usuarios agresivos, envidiosos o enfadados. Primero se aplicarán dos librerías, VADER Sentiment y TextBlob, que dan tres resultados posibles sobre si la frase es: positiva, neutral o negativa. Se asignará un valor de -1, 0 y 1 respectivamente a cada una de las frases analizadas. Si no da el mismo resultado con las dos librerías se cogerá el valor medio entre los dos resultados, es decir si uno devuelve como resultado neutral y el otro negativo se contará como 0,5 y si uno devuelve positivo y el otro negativo se contabilizará con 0.

Una vez se haya asignado ese valor se analizará mediante el diccionario propio que detectará si el tweet es sobre alegría, amor, enfado, miedo, sorpresa u otro donde se asignan los valores -1, -1, 2, 1, 0 y 0. Una vez asignados los valores, si el valor es negativo o cero significará que no es un tweet negativo y, por lo contrario, si es positivo significará que el texto es agresivo o de enfado. En estos algoritmos a diferencia de los anteriores se eliminarán únicamente los determinantes para hacer más rápida la clasificación sin modificar las palabras que podrían ser clave. Esto se hace porque cada palabra encontrada en uno de los diccionarios si, se sumará con un 1, en ese diccionario pero si se encuentra algún adverbio de cantidad como 'mucho' o 'muy' se sumará con un 2 y si son adverbios como 'poco' o 'algo' se sumará únicamente 0,5.

El método del diccionario es el mismo algoritmo que en la fase anterior, lo único que cambia son los *datasets* que se guardarán en el árbol ya que se tratará de un juego de datos con palabras relacionadas a los sentimientos y no a los deportes como en la primera fase.

A continuación podemos observar unos ejemplos de la aplicación del algoritmo:

```
..... Tweet .....  
CR7 es un creido, me da asco  
***** Sentiment Analysis *****  
----- VADER -----  
NEGATIVE  
----- TextBlob -----  
NEGATIVE  
----- Dictionary -----  
Frase clasificada con el sentimiento -> enfado <- encontrada/s {'asco'}  
Is furious  
  
..... Tweet .....  
Canastó de tres en el ultimo minuto, los aficionados de los Lakers no podian ocultar su furia  
***** Sentiment Analysis *****  
----- VADER -----  
NEGATIVE  
----- TextBlob -----  
NEGATIVE  
----- Dictionary -----  
Frase clasificada con el sentimiento -> enfado <- encontrada/s {'furia'}  
Is furious  
  
..... Tweet .....  
Ese jab le dejó besando la lona y a Myweather eufórico  
***** Sentiment Analysis *****  
----- VADER -----  
POSITIVE  
----- TextBlob -----  
NEUTRAL  
----- Dictionary -----  
Frase clasificada con el sentimiento -> alegría <- encontrada/s {'eufórico'}  
POSITIVE  
  
..... Tweet .....  
Ese flanker fué a por uvas  
***** Sentiment Analysis *****  
----- VADER -----  
NEUTRAL  
----- TextBlob -----  
NEUTRAL  
----- Dictionary -----  
Ninguna palabra encontrada dentro del diccionario  
NEUTRAL  
  
..... Tweet .....  
Nadal no pudo ocultar su asombro al perder contra Federer  
***** Sentiment Analysis *****  
----- VADER -----  
NEGATIVE  
----- TextBlob -----  
NEUTRAL  
----- Dictionary -----  
Frase clasificada con el sentimiento -> sorpresa <- encontrada/s {'asombro'}  
Is furious
```

Figura 35: Resultados obtenidos en clasificación de sentimientos.

En la figura anterior se puede observar como el algoritmo acierta en los dos primeros textos diciendo que son de gente furiosa. También acierta en el siguiente que es positivo pero, falla en los dos últimos dado que el penúltimo es algo malo y lo detecta como neutral y el ultimo es neutral de sorpresa y lo detecta como de furia.

4.5. Fase 3 - Resultado del análisis

Está última fase es la que toma todas las decisiones finales. En ella se acaba de decidir si es un *bot* que hace *SPAM* en su muro, un usuario agresivo o si el tweet se trata de una mentira o *fake news*. Todo esto se decide mediante a los datos que nos proporcionan las fases anteriores, lo único que hay que hacer es tratar esa información de manera correcta y cuidadosa. En concreto nos llega información del usuario de Twitter, el tema relacionado y el sentimiento del tweet a analizar y sus últimos tweets así como el texto original de ellos.

Para hacer este análisis más sencillo para el usuario se ha decidido hacer un menú con seis opciones según que quieras analizar en ese momento.

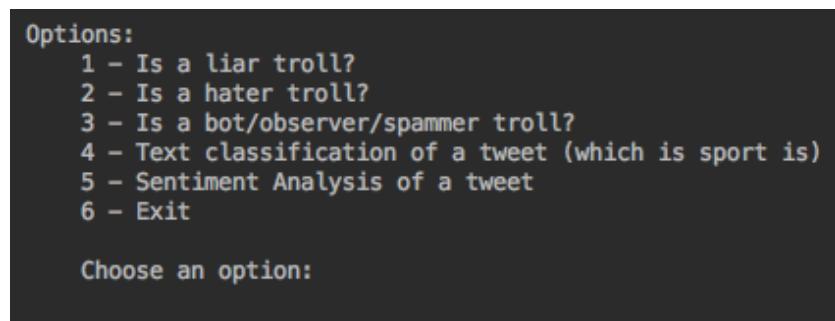


Figura 36: Menú principal del programa.

En la primera opción se pretende saber si ese tweet es un *fake* y por lo tanto es un *liar troll*. Para ello se tienen en cuenta diferentes variables como sobre que trata ese tweet y si es sobre ese tema el que escribe el usuario. También se mira si el tweet introducido tiene alguna similitud con los últimos mil tweets escritos sobre ese tema por los usuarios de Twitter, sobre esto hay que comentar que solo se detectaran fakes de últimas noticias. Por último, se averigua si ese perfil se trata de un usuario verificado o no. La formula quedaría como: $is_fake = 0,15*A + 0,25*B + 0,2*C + 0,4*D$. Donde:

1. $A = 1$ si el tweet trata sobre los temas que normalmente escribe ese usuario sino $A = 0$.
2. $B = 1$ si la cuenta es verificada sino $B = 0$.
3. $C = 1$ si el usuario se trata de un *spammer*, *bot* o *observer* sino $C = 0..$
4. $D = 1$ si existen tweets parecidos entre los últimos 1000 sino $A = D$.

Finalmente si con la formula anterior se obtiene un resultado mayor o igual 0,6 se dirá que es una mentira.

```
Choose an option:  
1  
  
-> Option is fake?  
    Write the tweet:  
    Futbol: Modric gana el the best  
    Write the user name:  
    jorgejmt94  
The tweet is about: Futbol  
There are more users writing about the same( 14 similar tweets)  
Is not a verified user.  
This user sames a normal user.  
  
Theres no reason to think it's a fake.
```

Figura 37: Ejemplo de un usuario no verificado que publica una verdad.

```
-> Option is fake?  
    Write the tweet:  
    Futbol: Modric gana el the best  
    Write the user name:  
    jPedrerol  
The tweet is about: Futbol  
There are more users writing about the same( 15 similar tweets)  
Is a verified user.  
The user writes about: Futbol  
This user sames a normal user.  
  
Theres no reason to think it's a fake.
```

Figura 38: Ejemplo de un usuario verificado que publica una verdad.

```
-> Option is fake?  
    Write the tweet:  
    Futbol: Modric gana el the best  
    Write the user name:  
    jPedrerol  
The tweet is about: Futbol  
There are more users writing about the same( 15 similar tweets)  
Is a verified user.  
The user writes about: Futbol  
This user sames a normal user.  
  
Theres no reason to think it's a fake.
```

Figura 39: Ejemplo de un usuario verificado que publica una verdad.

```
Choose an option:  
1  
  
-> Option is fake?  
    Write the tweet:  
La chilena de Cristiano es el autentico Puskas  
    Write the user name:  
jpedrerol  
The tweet is about: Futbol  
There are more users writing about the same( 5 similar tweets)  
Is a verified user.  
The user writes about: Futbol  
This user sames a normal user.  
  
Theres no reason to think it's a fake.
```

Figura 40: Ejemplo de un usuario verificado que publica un tweet que no es verdad pero hay más gente opinando igual.

```
Choose an option:  
1  
  
-> Option is fake?  
    Write the tweet:  
Messi merecia el the best  
    Write the user name:  
jPedrerol  
The tweet is about: Futbol  
There are no more users writing about the same.  
Is a verified user.  
The user writes about: Futbol  
This user sames a normal user.  
  
It seems an opinion
```

Figura 41: Ejemplo de un usuario verificado que publica una opinion.

```
Choose an option:  
1  
  
-> Option is fake?  
    Write the tweet:  
    Salah gana el the best!  
    Write the user name:  
    jorgejmt94  
    The tweet is about: Futbol  
    There are no more users writing about the same.  
    Is not a verified user.  
    This user sames a normal user.  
  
Attention!  
This user is probably a liar troll.
```

Figura 42: Ejemplo de un usuario no verificado que publica una opinion.

```
Choose an option:  
1  
  
-> Option is fake?  
    Write the tweet:  
    Luka Modric  
    Write the user name:  
    jorgejmt94  
    The tweet is not about any sport analyzed.
```

Figura 43: Ejemplo de tweet demasiado corto que no se encuentra sobre que tema trata.

Para encontrar los usuarios que sean *hater troll*, lo que se hace es analizar los últimos 10 tweets del usuario con la fase 2, si más de la mitad de los tweets que ha escrito son clasificados con un sentimiento de 'enfado' se dirá que ese usuario podría ser un *hater troll*.

```

Choose an option:
2

-> Option is an agresive user?
    Write the user name:
Pavone_CR
-> Last tweets:
Pq mierda Maxi Romero y Heinze no subsistieron en el mismo espacio y tiempo
CR7 es un creido, me da asco
A Cristiano le gusta llegar a casa y hacer 150 abdominales y a mi prender fuego para un asado
Modric es un falso asqueroso, no merece ese premio!!! #ascoDeFIFA
RT @guidomdebella: Primer sinsabor que la vida le devuelve... Vendrán más. Todo vuelve.
Te amo Heinze, pero como me rompe que no te importe el resultado y las tablas
Que envidia me da Atl Tucuman lpm

Attention!
The user seems an hater troll, better not to follow him

```

Figura 44: Análisis de un usuario considerado *hater troll*.

```

Choose an option:
2

-> Option is an agresive user?
    Write the user name:
jPedrerol
-> Last tweets:
¿Qué me pongo? ¿Qué chaqueta azul me pongo?☀️ #TheBestEnMega #Atresmedia en Madrid, Spain https://t.co/V7Tzz0GMxa
¿Habrá SORPRESA? https://t.co/L3RJJ4AV2Y
En MEGAAAAAAA https://t.co/5ZCm6pa06u
iMe gustan los domingos! Ayer @elchiringuitotv volvió a ser el LÍDER indiscutible del Late night en TDT con un FANT...
Yo soy del que diga el Loco Gatti... https://t.co/3kdpn0fAjq
Ufffff Ahora haced lo que queráis pero a las doce... @elchiringuitotv https://t.co/HqzYvqawV2
¿Intereses? https://t.co/ikmtI1gyHh

The user jPedrerol does not seem an hater troll

```

Figura 45: Análisis de si el periodista Josep Pedrerol es o no un *hater troll*.

Si escogemos la opción 3 se estudiara un usuario que puede ser **bot troll**, **observer troll** o **spammer troll**. Para hacer este análisis únicamente se pedirá al usuario que escriba el nombre del perfil a analizar.

En cuanto a un usuario *bot troll* o un *observer troll* no los distinguimos entre ellos y se clasifican como aquellos usuarios que tengan menos de un cuarto de seguidores respecto a los que ellos siguen y también no tiene tweets publicados (tendrá todos los papeles para ser un usuario espía o *observer troll*) o bien los tweets que ha publicado son clasificados como SPAM (se parecería más a un *bot troll*). Un ejemplo sería el usuario '@Mandeep20016':



Figura 46: Bot o observer troll.

Por lo tanto si escogemos la tercera opción y escogemos analizar el usuario anterior obtendremos:

```
Choose an option:
3

-> Option is a bot, spammer or observer troll.
Write the user name:
Mandeep20016
User name: Mandeep20016

Attention!
This user could be a bot or an observer troll, have 0 tweets, 0 followers, and follows 26 users.
```

Figura 47: Bot or observer troll detectrado por trolls detector.

Un perfil clasificado como *spammer troll* será aquel usuario que sus últimos 20 tweets se parecen mucho entre ellos y por lo tanto tengan el mismo significado o vengan a decir lo mismo. Para ello analizaremos la diferencia entre los mismos. Un ejemplo sería¹⁷:

¹⁷Se trata de un usuario inventado donde 'original text' serían sus últimos cinco tweets y 'modified text' los tweets tratados sin palabras vacías.

```

Original text 0 -> Nadal no pudo ocultar su asombro al perder contra Federer
Original text 1 -> Nadal asombrado al perder contra Federer
Original text 2 -> Nadal no pudo ocultar su asombro al perder
Original text 3 -> Federer ganó a Nadal
Original text 4 -> Nadal pierde contra Federer

Modified text 0 -> ['nadal', 'ocult', 'asombr', 'perd', 'feder']
Modified text 1 -> ['nadal', 'asombr', 'perd', 'feder']
Modified text 2 -> ['nadal', 'ocult', 'asombr', 'perd']
Modified text 3 -> ['feder', 'gan', 'nadal']
Modified text 4 -> ['nadal', 'pierd', 'feder']

Attention!
This user could be a spammer troll, always publishes the same.

```

Figura 48: Resultados obtenidos con un usuario clasificado como *troll spammer* (usuario inventado).

Un ejemplo real de un usuario *spammer* seria el usuario 'BangtanPromoARG' cuyos últimos tweets son:

The image shows three consecutive tweets from the user 'Beyond The Promo Argentina' (@BangtanPromoARG). Each tweet is identical in content and structure, asking users to vote for #IDOL, #FakeLove, or #DNA respectively, to get a higher ranking on #Ranking47. The tweets are timestamped at 10 hours ago.

- Tweet 1:** [✉️❤️] Voto por #IDOL de @BTS_twt para que suba en el #Ranking47 de #RDArgentina #RankingRD @RadioDisneyLA
- Tweet 2:** [✉️❤️] Voto por #FakeLove de @BTS_twt para que suba en el #Ranking47 de #RDArgentina #RankingRD @RadioDisneyLA
- Tweet 3:** [✉️❤️] Voto por #DNA de @BTS_twt para que suba en el #Ranking47 de #RDArgentina #RankingRD @RadioDisneyLA

Each tweet has the following engagement counts: 45 replies, 25 retweets, 24 likes, and 1 message. The profile picture for the user is 'PROMO BTS ARGENTINA'.

Figura 49: Clasificación suario 'BangtanPromoARG' por *trolls detector*.

```

Choose an option:
3

-> Option is a bot, spammer or observer troll.
    Write the user name:
BangtanPromoARG

Attention!
This user could be a spammer troll, always publishes the same.

```

Figura 50: Usuario 'BangtanPromoARG' clasificado como *troll spammer*.

Por último, también puede darse la posibilidad de que el usuario no sea ninguna de las tres cosas como por ejemplo un usuario que sus últimos 5 tweets sean¹⁸:

```

Original text 0 -> El portero la cogió fuera del área, debio ser falta y amarilla
Original text 1 -> Canastó de tres en el ultimo minuto, los aficionados de los Lakers no podian ocultar su furia
Original text 2 -> Ese jab le dejó besando la lona y a Myweather eufórico
Original text 3 -> Ese flanker fué a por uvas
Original text 4 -> Ese tio no vale nada

Modified text 0 -> ['porter', 'cog', 'are', 'debi', 'falt', 'amarill']
Modified text 1 -> ['canast', 'ultim', 'minut', 'aficion', 'lakers', 'podi', 'ocult', 'furi']
Modified text 2 -> ['jab', 'bes', 'lon', 'myweath', 'eufor']
Modified text 3 -> ['flank', 'fue', 'uvas']
Modified text 4 -> ['tio', 'val', 'nad']

This user sames a normal user.

```

Figura 51: Resultados obtenidos con un usuario que no es no es un bot, spammer o observer troll.

Este menú también permite clasificar solo un tweet mediante clasificación de texto (opción 4) y sería aplicar la fase 1 y la **clasificación o análisis del sentimiento** del tweet (opción 5) y correspondería a la fase 2.

```

4

-> Option Text classification of a tweet.
    Write the tweet:
Los All Blacks toman nota: "Los Pumas mejoraron mucho y saben explotar nuestras fallas"...
Text classified like::: Rugby with 1.0 points

```

Figura 52: Ejemplo opción 4, analizar un tweet sobre rugby.

¹⁸Se tratá de un usuario inventado donde 'original text' serían sus últimos cinco tweets y 'modified text' los tweets tratados sin palabras vacías.

```
Choose an option:  
4  
-> Option Text classification of a tweet.  
Write the tweet:  
Que alguien diga al arbitro que deje de contar, el @fcoscu ya no se levanta de la lona tras el "jab", los dos "crochet", el "swing" y el definitivo "uppercut".  
Text classified like:: Boxeo with 3.0 points
```

Figura 53: Ejemplo opción 4, analizar un tweet sobre boxeo.

```
Choose an option:  
5  
-> Option Sentiment Analysis of a tweet.  
Write the tweet:  
hacia años que no disfrutaba tanto viendo este real Madrid, parezco un crío con tanta euforia  
Frase clasificada con el sentimiento: alegría
```

Figura 54: Ejemplo opción 5, analizar un tweet alegre.

```
Choose an option:  
5  
-> Option Sentiment Analysis of a tweet.  
Write the tweet:  
Quique Saitén: "Están con tensión, les quema el balón. Tienen miedo a fallar, no se sienten apoyados". #Vamos  
Frase clasificada con el sentimiento: miedo
```

Figura 55: Ejemplo opción 5, analizar un tweet sobre miedo.

5. Costes del proyecto

5.1. Costes temporales

TODO: pensa quant has dedicar quant a revisar tema, implementar, recollir dades, experimentació, escriure memòria. Deixo al teu criteri els blocs. I ho podries pintar en forma de quesitos.

5.2. Costes económicos

TODO: si haguesis de monetitzar el què has fet quan costaria? Preu hores / analista, Preu/hores programador, etc, software (si és gratuit dir-ho), hardware, caldrà manteniment? el que crequis.

6. Conclusiones y líneas de futuro

6.1. Conclusiones

TODO

6.2. Líneas de futuro

Este ha sido un trabajo muy ambicioso dado que quería acaparar un gran volumen de conceptos y no se ha podido llevar a cabo todas las ideas que se tenían puesto que el tiempo para su realización era acotado. Con el tiempo que se ha tenido para su implementación se ha tenido un resultado satisfactorio, pero como todos los proyectos, se empieza con una idea de cómo se quiere hacer y al largo de su codificación se van teniendo más y más ideas de cómo podría ser mejor o en qué más ámbitos se podría aplicar con unos simples cambios.

Sin ir más lejos, en esta misma universidad en el mes de junio se entregó un trabajo final de máster que de recogía información de Twitter y cuya finalidad era predecir los resultados de los equipos de fútbol en la Premier League. Bien, él sólo cogía la información de las cuentas oficiales de los equipos, con el trabajo presentado anteriormente se podría reorganizar un poco para que, únicamente ampliando los diccionarios, se busque por todo Twitter quien está hablando de fútbol y después detectar sobre qué equipo hablan ampliando así el dataset del trabajo y como todos sabemos, cómo más grande son los datasets (si los datos son útiles) más probabilidad tenemos de que nuestro algoritmo tenga buenos resultados dado que se entrena con un número mayor de datos.

La primera gran restricción de este trabajo fue el tiempo, este factor hizo que se acotara la búsqueda de trolls a solo el mundo de los deportes, pero con más tiempo y personal, lo primero que se podría hacer es aumentar el número de diccionarios con la finalidad de poder abarcar más temas como accidentes o el tiempo e incluso dentro de los deportes separarlos por clubes y jugadores para hacer un análisis aún más exhaustivo. Gracias a que esto ya se sabía desde un principio, se realizó una implementación que permite fácilmente la ampliación de los diccionarios y con lo único que se tendría que pelear es con las hipótesis como que quien habla mucho de fútbol raramente hablará de rugby puesto que son deportes que no se combinan demasiado bien.

Otra gran barrera de este trabajo fue el idioma. El idioma con el que trabajan todos los algoritmos ya implementados es el inglés y este trabajo lo hace con el español. Esto implica a no poder usar muchos de los algoritmos previamente implementados en librerías y tener que hacer uso de una implementación propia. Se pretende en un futuro traducir los diccionarios a otros idiomas y con la codificación propia de los algoritmos, no tendría

que haber demasiados problemas en poder usar esas traducciones y así incluso, se podrá resolver el problema de los usuarios que por ejemplo escriben en castellano, pero utilizan expresiones en inglés.

Enlazado con el punto anterior, también se pretende hacer en un futuro un crecimiento en el análisis de los emoticonos que escriben los usuarios y que en muchas ocasiones te pueden dar más información que el texto que ha escrito dado que los emoticonos tienen un alto grado de información.

Otra posible ampliación para mejorar el análisis de los algoritmos es la corrección automática de los twits dado que en Twitter se habla con un lenguaje vulgar y se cometen tanto faltas de ortografía como abreviaciones que no existen como sería el caso de tk que significa te quiero.

Con el alto grado de estudio de cada usuario, si se ampliaran los temas de los diccionarios, también se podría usar para ayudar al algoritmo de Twitter de sugerir gente para seguirla dado que se conoce sobre qué temas ha escrito y por lo tanto sus intereses.

A partir del análisis del sentimiento se podría sugerir a los usuarios de dejar de seguir a gente considerada agresiva, que habla con malos modos o que siempre están enfadados dado que en las redes sociales nadie quiere tener problemas.

Este trabajo también nos da la facilidad de recopilar información de los usuarios por temáticas, es decir, se podría usar, añadiendo nuevamente nuevos temas en los diccionarios, para poder hacer extracciones de datos tanto como de fútbol, como del tiempo o de cualquier tema que seamos capaces de crear un diccionario.

Como hemos podido ver este trabajo ofrece un abanico muy grande tanto para mejorarlo como para hacer de base en nuevos proyectos como serían sugerir gente a quien seguir, la extracción de información de twitter, avisar a la gente que siguen a gente agresiva o incluso revolucionar el mundo de las apuestas.

7. Bibliografia

Referencias

- [1] Miguel Silvestre, “Como detectar a una cuenta falsa en Twitter,” *ABC*, 2014. Disponible en: “<http://www.abc.es/tecnologia/redes/20140320/abci-faketwitter-saber-twitter-falsos-201403191326.html>”
- [2] @elentir, “Twitter: 5 formas de identificar a un troll y 5 consejos para librarte de él,” *outono*, 2013. Disponible en: “<http://www.outono.net/elentir/2013/02/02/twitter-5-formas-de-identificar-a-un-troll>”
- [3] Alana Moceri, “Cómo distinguir las noticias falsas en redes sociales,” *entrepreneur*, 2015. Disponible en: “<https://www.entrepreneur.com/article/292342>”
- [4] Tecnoesfera, “Manual para identificar noticias falsas y evitar su propagación,” *El tiempo*, 2018. Disponible en: “<http://www.eltiempo.com/tecnosfera/novedades-tecnologia/como-identificar-noticias-falsas-en-redes-sociales>”
- [5] TICBEAT.COM, “Cómo detectar una noticia falsa en internet,” *ABC*, 2013. Disponible en: “<https://www.abc.es/tecnologia/redes/20131210/abci-como-detectar-noticia-falsa-201312092125.html>”
- [6] Ruben Sanchez, “Los quince tipos de troll que se ocultan en las redes sociales,” *El confidencial*, 2015. Disponible en: “https://blogs.elconfidencial.com/tecnologia/elclubdelalucha/2015-05-09/troll-redes-sociales-ciberacoso_790558/”
- [7] Izabela Pecherska, “Día del Troll: nueve tipos de ‘trolls’ que puedes encontrar en redes sociales,” *El mundo*, 2017. Disponible en: “<http://www.elmundo.es/f5/comparte/2017/10/19/59e77fb22601db82d8b459f.html>”
- [8] “PyMongo” *Documentación*, 2018. Disponible en: “<https://api.mongodb.com/python/current/>”
- [9] Jose Luis Moreno, “Características de un tweet confiable,” *paper*, 2012.
- [10] Gabriela Gonzalez, “7 herramientas para detectar si tienes seguidores falsos en Instagram y Twitter,” *genbeta*, 2018. Disponible en: “<https://www.genbeta.com/redes-sociales-y-comunidades/7-herramientas-para-detectar-si-tienes-seguidores-falsos>”

- [11] Lauren Goode, “Google just made its troll-detecting software available to developers,” *theverge*, 2017. Disponible en: “<https://www.theverge.com/2017/2/23/14713496/google-jigsaw-perspective-software-ai-machine-learning-developers>”
- [12] Jane Wakefield, “Fake news detector plug-in developed,” *BBC*, 2016. Disponible en: “<http://www.bbc.com/news/technology-38181158>”
- [13] Patxi GalAn-GarcIa, JosE Gaviria de la Puerta, “Supervised machine learning for the detection of troll profiles in twitter: application to a real case of cyber-bullying,” *Oxford*, 2015. Disponible en: “<https://academic.oup.com/jigpal/article-abstract/24/1/42/2893010?redirectedFrom=fulltext>”
- [14] IPTC, News Topics, 2018. Disponible en: “<http://show.newscode.org/index.html-newscode=medtop&lang=en-GB&startTo>Show>”
- [15] Wikilengua, Glosario de deportes, 2015. Disponible en: “http://www.wikilengua.org/index.php/Glosario_de_deportes”
- [16] Tweepy, Documentación, 2018. Disponible en: “<http://www.bibtex.org/Using/>”
- [17] GUILLERMINA TORRESI, , “¿Quién se esconde detrás de un ‘troll’ en la red?,” *La Vanguardia*, 2015. Disponible en: “<https://www.lavanguardia.com/tecnologia/20170404/421446008040/trolls-twitter-haters-redes-sociales.html>”
- [18] PyWiki, 2018. Disponible en: “<https://pypi.org/project/wikipedia/>”
- [19] Steve Kramer, “Identifying viral bots and cyborgs in social media, 2017. Disponible en: “<https://www.oreilly.com/ideas/identifying-viral-bots-and-cyborgs-in-social-media>”
- [20] Silvia Camargo, “Cómo detectar mentiras en internet, 2011. Disponible en: “<https://www.fucsia.co/opinion/blogs/entrada-blog/como-detectar-mentiras-internet/35060>”
- [21] Gabriela Ulloa , “3 señales para detectar cuándo te mienten en Facebook o WhatsApp, 2013. Disponible en: “<https://www.biobiochile.cl/noticias/2013/09/16/3-senales-para-detectar-cuando-te-mienten-en-facebook-o-whatsapp.shtml>”

- [22] Rocío P. Benavente, “Pheme, un detector de mentiras para las redes sociales”, ” *El confidencial*, 2014. Disponible en: “https://www.elconfidencial.com/tecnologia/2014-02-27/pheme-un-detector-de-mentiras-para-las-redes-sociales_94300/”
- [23] Anónimo, “¿Hasta dónde llegarán las mentiras en internet?”, ” *Semana*, 2017. Disponible en: “<https://www.semana.com/vida-moderna/articulo/mentiras-en-las-redes-sociales/521862>”
- [24] “Comparación R y Python ” *Club de la tecnología*, 2016. Disponible en: “<http://www.clubdetecnologia.net/blog/2017/python-vs-r-cual-es-mejor-para-la-ciencia-de-datos/>”
- [25] Alex Rayón, “Analítica R y Python”, ” *Deusto Data*, 2015. Disponible en: “<https://blogs.deusto.es/bigdata/eliendo-una-herramienta-de-analitica-sas-r-o-python/>”
- [26] “MongoDB vs MYSQL ” *DA-14*, 2017. Disponible en: “<https://da-14.com/blog/mongodb-vs-mysql-comparison-which-database-better>”
- [27] “NaiveBayes ” *scikit-learn*, 2018. Disponible en: “http://scikit-learn.org/stable/modules/naive_bayes.html”
- [28] “SVM ” *scikit-learn*, 2018. Disponible en: “<http://scikit-learn.org/stable/modules/svm.html>”
- [29] Niklas Donges, “The Random Forest Algorithm”, ” *Towards Data Science*, 2018. Disponible en: “<https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd>”
- [30] “XGBoost ” *Medium Corporation*, 2018. Disponible en: “<https://medium.com/syncedreview/tree-boosting-with-xgboost-why-does-xgboost-win-every-machine-learning-competition-ca8034c0b283>”
- [31] “VADER Sentiment ” *Documentación*, 2018. Disponible en: “<https://github.com/cjhutto/vaderSentiment>”
- [32] “TextBlob’ ” *Documentación*, 2018. Disponible en: “<https://textblob.readthedocs.io/en/dev/>”
- [33] “NLTK ” *Documentación*, 2018. Disponible en: “<https://www.nltk.org/>”

- [34] “Homonimia y Polisemia” *Wikilengua*, 2018. Disponible en:
[“http://www.wikilengua.org/index.php/Homonimia_y_polisemia”](http://www.wikilengua.org/index.php/Homonimia_y_polisemia)
- [35] “Anafora y elipses” *Razonamiento verbal*, 2018. Disponible en:
[“http://tenemosgrandesideas.blogspot.com/2012/07/anafora-catafora-y-elipsis.html”](http://tenemosgrandesideas.blogspot.com/2012/07/anafora-catafora-y-elipsis.html)
- [36] “Ontología Deportes” *Git Hub*, 2018. Disponible en:
[“http://tenemosgrandesideas.blogspot.com/2012/07/anafora-catafora-y-elipsis.html”](http://tenemosgrandesideas.blogspot.com/2012/07/anafora-catafora-y-elipsis.html)
[https://github.com/Tobion/Sports-Ontology”](https://github.com/Tobion/Sports-Ontology)

8. Anexos

8.1. Manual MongoDB

Para instalar MongoDB hay dos maneras:

```
Download package from:  
$ https://www.mongodb.com/download-center#production  
In the same directory execute the commands via Terminal:  
$ tar -zxf mongodb-osx-ssl-x86_64-4.0.2.tgz  
$ export PATH=<mongodb-install-directory>/bin:$PATH  
Or via Brew execut commands on Terminal:  
$ brew update  
$ brew install mongodb  
$ brew install mongodb --devel
```

Figura 56: Instalar MongoDB.

Se puede ejecutar haciendo las siguientes comandas:

```
Now you can run mongoDB by executing command:  
$ mongod  
On the terminal you can:  
# CREATE the database  
    use train  
# Be sure you are in the database  
    db  
# CREATE the collections  
    db.createCollection('sports')  
# Be sure you create it  
    show collections
```

Figura 57: Ejecutar MongoDB en nuestro ordenador.

En la siguiente figura se pueden observar las principales funciones de Connect, Get e Insert.

```
## Functionalites via python ##
#Connect to DB #
def connectDB(name_db, name_collection):
    client = MongoClient('localhost', 27017)
    return client[name_db], client[name_db][name_collection]

# INSERT in a collection #
def INSERT_json_toDB(db_name, cl_name, data_json):

    db, cl = connectDB(db_name, cl_name)
    result = cl.insert_one(data_json)

# GET collections #
def GET_dictionary_from_DB():
    db, cl = connectDB('train', 'dictionary')
    cursor = cl.find({})
    data = []
    for document in cursor:
        data.append(Utils.Dictionary(
            document['type'],
            document['keyWords'],
            document['secondaryWords'],
            document['excludingWords']
        ))
    return data
```

Figura 58: Funcionalidades MongoDB.

8.2. Manual Tweepy

Para la conexión con Twitter, Tweepy necesita pedir las credenciales en:

<https://apps.twitter.com/>.

Una vez se obtienen las credenciales se tendrán las siguientes 'keys': 'CONSUMER_KEY', 'CONSUMER_SECRET', 'ACCESS_TOKEN' Y 'ACCESS_SECRET' que se necesitan para hacer la conexión con la API de Twitter.

```
# API's setup:  
def twitter_setup():  
    # Authentication and access using keys:  
    auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)  
    auth.set_access_token(ACCESS_TOKEN, ACCESS_SECRET)  
    # Return API with authentication:  
    api = tweepy.API(auth, wait_on_rate_limit=True)  
    return api
```

Figura 59: Conexión con Twitter desde Python.

Una vez obtenida la conexión se pueden hacer diferentes funciones como obtener los últimos 'n' tweets por hashtag:

```
def get_tweets_by_hashtag(word, n_tweets, date, show):  
    api = twitter_setup()  
    word='#'+word  
    tweets = api.search(q=word, since=date, count=n_tweets)  
    if show:  
        print('Ya tenemos tweets sobre:', word)  
    return tweets
```

Figura 60: Obtener tweets por Hashtag.

Pedir los últimos ' n ' tweets de un usuario:

```
def get_last_tweets(user_name, n_tweets):
    # create an extractor object:
    extractor = twitter_setup()

    # create a tweet list as follows:
    twts = extractor.user_timeline(screen_name=user_name, count=n_tweets)
    tweets=[]
    for tweet in twts:
        #tweet = clean_tweet(tweet)
        tweets.append(get_tweet_data(tweet))
    return tweets
```

Figura 61: Obtener últimos X tweets.

Hacef la llamada para obtener la información de un usuario:

```
def get_user_data(user_name, n_tweets):
    import Utils
    # create an extractor object:
    extractor = twitter_setup()
    user = extractor.get_user(user_name)
    return Utils.User(user.description, user.verified, user.followers_count)
```

Figura 62: Obtener información del usuario.

8.3. Ontología de los deportes

La siguiente figura muestra una ontología para definir y separarlos deportes [36].

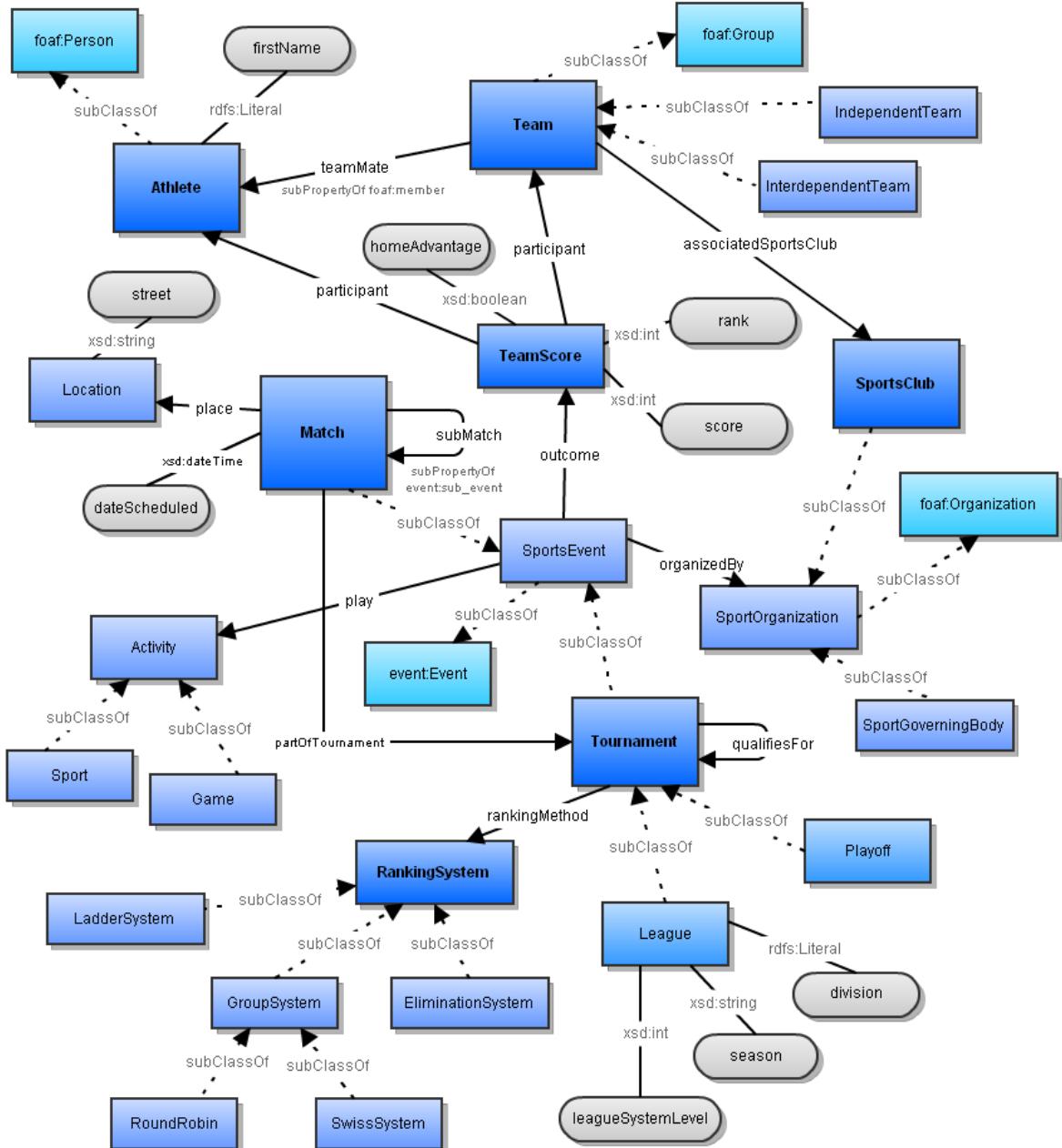


Figura 63: Ontología de los deportes.

8.4. Diccionarios

8.4.1. Palabras clave

8.4.1.1 Atletismo

Wgx, jogging, maratón, maratoniano, iron man, pentatlón, pértiga, skipping, bolt starting blocs, steeple-chase, tartán, fondista, runner, water jump, relevista, marcha, marchador, jabalina, javelin, lanzador de jabalina, lanzador de disco, lanzamiento de jabalina, lanzamiento de disco, lanzamiento de martillo, salto de longitud, saltador de longitud, triple salto, salto con pértiga, pertiguista, atleta, atletismo, pentathlete, pentatleta, usai, phelps, michael phelps, ruth betia, carl lewis, yelena isinbayeva, haile gebrselassie, michael johnson, jesse owens, Hicham El Guerrouj, Kenenisa Bekele, jamaica, jamaicano, Javier Sotomayor, Paquillo Fernández, Maria Lourdes Mutola, 110m, 110m vallas, javier Gómez Noya, Ashton Eaton.

8.4.1.2 Baloncesto

Baloncesto, basketball, básquetbol, basquetbol, baloncestista, basket, básquet, basquet, canasta, basquetbolista, dos puntos, tres puntos, nba, acb, fiba, lakers, boston celtics, ccleveland cavaliers, san antonio spurs, golden state warrios, philadelphia 76ers, chicago bulls, toronto raptors, michael jordan, miami heat, knicks, pau gasol, Kobe Bryant, Magic Johnson, Larry bird, o'neal, saquille o'neal, allen iverson, lebron james, tim duncan, kevin garnet, stephen curry, steve nash, kevin durant, derrick rose, ricki ru-bio, rudy fernández, josé calderoón, marc gasol, sergio llull, felipe reyes, jorge garbajosa, fernando romay, víctor claver, final four, mate, alley loop.

8.4.1.3 Balonmano

balonmano, handball, hándol, Nikola Karabatić, Talant Dujshebaev, vano Balic, IHF, ahf, cahb, pathf, ohf, ehf, asobal, joan cañellas, nikola karabatic, mikkel hansen, domagoj duvnjak, laszlo nagy, timur dibirov, dean bombac, arpad sterbik, julken aginagalde, kiril lazarov, daniel narcissse, filip jícha, stawomir szmal, andreas wolff, cristina neagy, eduarda amorin, andrea lekic, alexandra do nascimiento, heidi lekic.

8.4.1.4 Boxeo

Boxeo, boxeen, boxing day, boxing, crochet, cuadrilátero, cup protector, noquear, noqueo, punch, punching bag, punching ball, lona, ring, ring side, sparring, peso pesado, pesos pesados, peso ligero, pesos lijeros, peso mosca, peso gallo, peso wélter, peso

semipesado, peso semipesados, peso medio, peso superpesado, muhammad ali, rocky, rocky marciano, emilie griffith, sugar ray robinson, joe louis, julio cézar chávez, floyd, mayweather, floyd mayweather, título mundial.

8.4.1.5 Ciclismo

Ciclismo, ciclista, ciclistas, Eddy Merckx, Bernard Hinault, Fausto Coppi, Jacques Anquetil, Gino Bartali, Miguel Indurain, Nairo Quintana, Alberto Contador, Chris Froome, Alejandro Valverde, Peter Sagan, Vincenzo Nibali, Tom Dumoulin, Mark Cavendish, Greg Van Avermaet, bici, bicis, ruedas, rueda, bicicleta, bicicletas, vuelta españA, vuelta de españA, tour de francia, france tour, tour francia, Giro d' Italia, Giro de Italia, Giro Italia, 101 kilómetros de Ronda, Volta a Portugal, dos ruedas.

8.4.1.6 Fútbol

Futbol, fútbol, football, soccer, futbolista, larguero, paradinha, cola de vaca, A.F.A, F.E.F, fifa, F.I.F.A, aneff, uefa, balonpie, balonpié, bundesliga, Premier League, liga bbva, liga santander, liga de campeones, champions legue, panenka, hat trick, hat-trick, Diablos rojos, Los Blues, boixos nois, naranja mecánica, submarino amarillo, eje de la zaga, tarjeta amarilla, tarjeta roja, balon de oro, balón de oro, bota de oro, entre los tres palos, cabecear, línea de cal, canaletas, cartulina, casión, espinillera, espaldinha, goleador, efiCacia goleadora, equipo ascensor, folha seca, gambeta, killer, keeper, media luna, coutinho, var, pichichi, proroga, minuto 92, minuto 93, tiempo de descuento, puntapié, racha goleadora, messi, lionel messi, ronaldo, maradona, pele, pelé, ronaldinho, diego armando maradona, zidane, zinedine zidane, di stefano, cruyff, beckenbauer, gErard pique, busquets, cr7, d10s, arbeloa, alvaro arbeloa, roncero, neymar, de gea, bellerín, ramos, sergio ramos, pogba, arturo vidal, hazard.

8.4.1.7 Fútbol Americano

NFL, futbol americano, fútnol americano, american football, super bowl, quarterback, qb, touchdown, touch down, td, yac, Barry Sanders, Jerry Rice, Walter Payton, Joe Montana, Peyton Manning, Lawrence Taylor, Johnny Unitas, Reggie White, Jim Brown, Joe Greene, ncaa.

8.4.1.8 Golf

Hoyo, golf, golfista, hole in one, hoyo en uno, dougle eagle, madera 1, approach, backspin, tiger woods, seve ballesteros, jon rahm, roy mcllroy, jordan spierh, josé maría

olazábal, dustin johnson, masters de augusta, the us masters, pga, bola wound de núcleo líquido, bola wound con el núcleo sólido, bola múlticapas, chaqueta verde.

8.4.1.9 Judo

Chui, dan, doyo, hayime, ipon, ippon, judo, kachi, cinturón negro, koka, kuzushi, matel, oseakomi, randori, sono mama, sore made, suri-ashi, tatami, toketa, tori, uke, ukemi, waza ar, yudo, yudogui, quimono, yudoca, judoca, yuko,yamashita yoshiaki, isogai hajime, suichi nagaoka, kelmendi majilinda, kuziutina, miranda erika, nakamura misato, shishime, siuffrida, cohen gili, krasniq, ma yingnan, kata.

8.4.1.10 Motociclismo

Motorista, motociclista, motociclismo, motocicleta, gp, supErbike, supersport, superstock, motocross, trial, enduro, supermoto, rally raid, loasail, marc márquez, Andrea Dovizioso, Cal Crutchlow, lorenzo, Dovizioso, Johann Zarco, Maverick Viñales, Jack Miller, Danilo Petrucci, Valentino Rossi, rossi, Alex Rins, Andrea Iannone, Iannone, Esteve Rabat, Dani Pedrosa, Álvaro Bautista, Jorge Lorenzo, Aleix Espargaró, nicky hayden, dos ruedas, mundial de motociclismo.

8.4.1.11 Rugby

Rugby, rugbi, tackle, placaje, ruck, melé espontánea, melé, maul, mêlée, scrumensayo de castigo, rugbier, talonador, rugbista, medio melé, scrum-half, Springboks, All Blacks, pumas, Les Bleus, nations cup, 6 naciones, seis naciones, Beauden Barrett, Daniel Carter, Brodie Retallick, Kieran Read, Thierry Dusautoir, Richie McCaw, Shane Williams, Bryan Habana, Jonny Wilkinson, jonah lomu, Sebatien Chabal, ma'a nonu, Juan Martin Hernandez, Kees Meeuws.

8.4.1.12 Tenis

Tenis, tenista, raqueta, Match ball, match point, rafael nadal, rafa nadal, roger federer, novak djokovi, andy murray, del potro, david ferrer, marin cilic, tomas berdych, serena williams, hermanas williams, fernando verdasco, feliciano lopez, david goffin, john isner, maria sharapova, simona Halep, richard gasquet, carlos moyá, juan carlos ferrero, doble falta, set en blanco, ojo de halcón, alley, atp, punto de partida, punto de rotura, open australia, roland garros, wimbledon, us open.

8.4.1.13 Voleibol

Voleibol, volleyball, vóleibol, volibol, voleybol, voleyball, vóley, balonvolea, fivb, Gilberto Godoy Filho, giba, Wilfrido León, Ivan Zaytsev, Mariusz Wlazly.

8.4.2. Palabras secundarias

8.4.2.1 Atletismo

Carrera, salto, flat, liso, plano, obstáculos, recorrido, record, deportista, élite, miler, pista, pistas, velocidad, lisos, 200, 1500, natación, oro, plata, bronce, curb, aceleración, acelerar, sprint, final, miller, ría, relevos, walker, cross, disco, swing, stop board, throw, lanzar, lanzamiento, martillo, peso, saltar, triplista, pole, ACLIMATACIÓN, esróbico, agilidad, arena, articulaciones, batida, carrera continua, circuito, colchoneta, cuerda, fondo, photo finish, photo-finish, jaula, intervalo, s, m, resistencia, serie, tabla, justo, juegos, olimpiada, hinvierno, campeon, campeón, 800, 5000, 5.000, 10000, 10.000, 100, triple salto, vallas, metros, 4x100, YOHAN BLAKE, WAYDE VAN NIEKERK, mundial, físico, juegos de río, juegos atenas, juegos pekín, CATERINE IBARGUEN, campeona olímpica, campeon olímpico, olimpiadas, olímpico, olímpica, jjoo, estadounidense, ganador, heptalón, disciplina, subcampeón, prueba, distancia, campeonato, marca, record del mundo, campeonato de la PGA.

8.4.2.2 Baloncesto

Balon, balón, equipo, jugador, cinco, tiro, puntos, liga, olímpicos, draft, falta, control, pívot, entrenador, spurs, cancha, arbitraje, tiro a canasta, bote, defensa, falta en ataque, asistencia, bloqueo, canasta limpia, contraataque, pase de pecho, pase picado, pasos, rebote, penetración, tiro lible, tiro en suspensión, base, ala-pívot, play-off, fase final, oro, plata, bronce, drafteado, rookie, cheerleader, triple, tres, dos, más uno, más uno, zona press, marcate, zona, zonal, lob, fly, flight, guard, escolta, conductor, campo atrás, campo atras, cesta, 10 minutos, aro, juez principal, umpire, aero, defensa al hombre, falta personal, jugada de cuatro puntos, jugada de tres puntos, salto entre dos, pivotar, pantalla, gancho, buzzer beater, cross-over, bandeja, tablero, área restringida, slam dunk, cesta limpia, tocó el aro, tapon, tapón, chapa, anotó, venda, vendaje, entró, sustituir, defensor.

8.4.2.3 Balonmano

Fly, vuelo, marcar, pívot, time out, falta, defensa, ala, portero, fundéu, jugadores, línea, balón, lanzar, gol, central, tiempos, guardameta, descanso, banda, goles, siete contra siete, línea de 7, línea de 9, metros, línea de limitación, línea de gol, línea discontinua, 7 metros, 9 metros, amonestado, tarjeta amarilla, exclusión, sancionado, sanción, ángulo corto, ángulo largo, finta, feint rectificado, lanzamiento, golpe franco, lanzamiento recti-

ficado, shot, fallaway, hip throw, jump shot, bloqueo blocale, barrera, pasos, yugoslava, a dos manos, muñeca, wrist, juego, pista, anotar, equipo, cinco, cancha, contrataque, amistoso, entró, parada, sustituir, defensor.

8.4.2.4 Boxeo

Arbitro, golpe bajo, break, clinch, agarrar, agarre, coquille, protección, cross, cros, fighter, golpe, golpe cruzado, k.o, ko, gong, jab, gancho, kao, nocaut, knock out, knock down, speed bag, straight, superwelter, swing, uppercut, welter, cinturón, cinturón de campeon, cinturon, pugil, pugilismo, puño, puñetazo, golpear, nuca, peso, asalto, titulo, título, derribo, derribar, victoria, juez, victoria por decisión, victoria por puntos, esquina, cuerda, contra las cuerdas, campana, round, bucal, protegerse, calcertas, bata, guantes, entrenamiento, saco, todos por ko, todos por ko, derrota, puntos, caer, combate, guante, venda, vendaje, cuerdas, lanzar, lanzó.

8.4.2.5 Ciclismo

Coll, col, esprín, sprint, esprinter, grimpeur, malla, malla rosa, maglia, maillor, maiyót, mailót, pelouse, routier, stayer, surplace, volata, abanico, peloton, abrirse, abre, abrir, autobús, badana, biela, buje, cabra, cadencia, cazaetapas, etapa, chichoera, corona, cuernos, culotte, demaraje, desarollo, diirección, corredor, frenos, hacer un recto, hirerro, llantazo, manguito, mina, ppiano, grupo, rebufo, rodador, regular, vampiro, tirón, zapata, vuelta, tour, giro, pista, ruta, prueba modalidad, montaña, bajada, subida, carrera, obstáculos, km, distancia, manillar, rueda, rodar.

8.4.2.6 Fútbol

Penalti, autogol, calcio, chut, centro, falta, mano, forward, portería, porteria, patada, manotazo, chutar, tiro, disparo, arbitro, arbitrar, alirón, amago, amistoso, barça, chilena, contraataque, contragolpe, filial, vaselina, escorpion, espuela, Merengue, culé, vikingos, galo, Hooligans, tifos, tifosis, poste, madera, plancha, lateral, pivote, extremo, defensa, portero, centrocampista, volante, delantero, delantero centro, carrilero, lateral derezho, lateral izquierdo, tanto, marcar, anotar, anotó, autobús, autobus, media parte, segunda parte, primera parte, primera mitad, balón dividido, banderín, banderin, barrera, bota, brazalete, capitán, césped, clasificación, control, cuerpo técnico, defensa adelantada, derby, derechazo, zurdazo, desbordar, desborde, desmarque, despejar, despeje, triplete, doblete, elástica, derrota, escudo, estadio, eurocopa, eliminatoria, encarrilar, expulsar, fair play, farolillo rojo, fichar, gol de oro, hueco, pase, imbatido, imbatibilidad, lider, líder,

liderazgo, juego aéreo, juego limpio, jugada ensayada, jugón, jugadearazo, goal, mánager, mejora de contrato, mojar, mundialito, obús, obstrucción, omnipresente, o'rey, palco, paquete, partido, pase de la muerte, pase en profundidad, pase largo, pase raso, pay per view, peinar, pelota parada, penalty, perdonar, canaletas, neptuno, cibeles, play off, potencia, presentacion, presentación, prima, primera vuelta, segunda vuelta, puntuar, puerta grande, puerta pequeña, a puerta, a puerta vacía, quiebro, recogepelotas, rechace, reconocimiento médico, regular, rectangulo de juego, regatear, remontada, remontar, renovar, replegar, replegarse, revulsivo, rival, rivalidad, clásico, robo, rondo, rotación, rueda de premsa, saque de banda, saque de esquina, salvación, seleccionador, semifinal, final, once, sombrero, subcampeón, superioridad, suplente, suspensión cautelar, tanda de penaltis, tanda de penalties, teleespectador, bicampeón, tijera, tirarse de plancha, tiro libre, offside, esferico, el cuero, fuera de juego, pena maxima, alineación indebida, abrir la lata, abre la lata, auto pase, cambio de orientación, ariete, entrar, entró, túnel de vestuario, tunel de vestuarios, vaca sagrada, velocidad, veterano, colores, la roja, vuelta de honor, zamora, zamorana, rabona, zaga, cristiano, manos, millones, confederación, campo, madrid, barcelona, zaragoza, historia, sede, hombro, codo, antebrazo, àngulo corto, àngulo largo, palo largo, finta, golpe franco, corner, còrner, catenaccio, escuadra , cancerbero, linier, fondo de la red, red, pito, marco, volea, adelantad, punto de panalty, quiniela, venda, vendaje, botas, tacos, pies, rueda el balón, sustituido, defensor.

8.4.2.7 Fútbol Americano

Cheerleader, cheerleaders, equipo, All Pro, Artificial turf, Assistant coaches, coach, Audible, Back, ball, balón, Beat, Blitz, carga, Blindsdie, Bootleg, Complete pass, Pase completo, Esquinero, Cornerback, Cut back, cut, dead ball, Depth chart, defensiva, defense, Draft, Eligible receiver, Fair catch, receptor, pañuelo, flag, casco, helmet, protecciones, football, interceptar, interceptado, pase, pass rush, presión, pateador, playbook, libro de jugadas, posesion, posesión, bowl, quarter, periodo, red flag, pañuelo rojo, arbitro, entrenador, tackle, placaje, anotar, anotación, conversion, dos puntos, West Coast Offense, Wide receiver, Wildcard, Wildcat offense, ala, ofensiva wildcat, receptor, huddle, runningback, fullback, cuarto down, hole, pases, pase completo, pase incompleto, snap, scrimmage, línea de scrimmage, balón suelto, fumble, Umpire, Head Linesman, back judge, field judge, side judge, falta personal, partido, juego, final, falta, equipo, victoria, campo, suplente, mano, manotazo, venda, vendaje, botas, tacos, entre palos, sustituir, defensor, palos, deribo, derribar, chute, chutar, lanzar, lanzó.

8.4.2.8 Golf

Agujero, calle, driving range, campo de prácticas, fairway, green, verde, hierba fina, búnker, arenero, arena, obstáculo, hazard, water hazard, regata, charco, rough, matorral, tee box, zona de salida, course, recorrido, link, marshal, supervisor, supervisor de juego, supervisor de pista, encargado, hole, albatross, eagle, birdie, par, doble bogey, driver, madera, hierro, híbrido, putter, chip, dogleg, drive, slice, top spin, golpe, golpe con efecto, golpe largo, torpar, putt, golpe corto, open, master, major, sergio garcia, campo, hoyos, bogey, swing, mango, tecnica, ega, hook, loft, match play, medal play, stroke play, juego por golpes, eclectic, establefor, mulligan, us open, the u.s open, guantes, spike, spikes, bola, palos.

8.4.2.9 Judo

Victoria, amonestado, amonestación, arbitro, falta, falta leve, árbitro, interrupción, caída, kimono, delgado angelica, técnica, cuello, maestro, rojo, azul, amarillo, judogui, grado, japonés, olímpico, piernas, caer, cayó, antebrazo, potencia, demostración, naranja, combate, oro, plata, bronce, peso, grand slam.

8.4.2.10 Motociclismo

Velocidad, velocidades, freno, caer, tiro, motro, distancia, raid, circuito, montmeló, piloto, vehiculos, vehiculo, asfalto, curva, 125, 250, campeonato, jerez, termas de río hon-doo, las américa, le mans, mugello, assen, sachsenring, brno, red bull ring, silverstone, misano, chang international circuit, motegi, phillip island, sepang, mono, casco, manillar, pnéumatico, pnématicos, rueda, rodar, boxes, slick, liso, neumático, neumáticos, cochera, neumáticos mixtos, neumáticos de lluvia, safety car, pit wall, pole position, pit, parrilla, parrila de salida, penalización, stop and go, team radio, paddock, motorhome, warm up, grip, chattering, wheelie, caballito, feeling, rookie, motohome, piano, pianos, salida, yamaha, suzuki, honda, pit-lane, pit lane, qualifying, entrenamientos libres, en-trenamiento libe, marca.

8.4.2.11 Rugby

Avant, fuera de juego, offside, onside, pase forward, pase adelantado, ofensivo, defen-sa, mark, golpe, golpear, puntapié, franco, touch, zonas laterales, line-out, ensayo, try, free-kick, kick-off, foul, foul play, zona de ensayo, zona de marca, knock-on, adelantado, penalty, catigo, penal, drop goal, conversion goal, transformación, conversión, delantera, primera linea, pilar, pilar izquierdo, pilar derecho, prop, lock, segunda linea, hooker,

second row, tercera línea, flanker, ala, centro, línea de tres cuartos, backs, fly-half, apertura, wing, zaguero, contacto, 15, 22, balón, ovalado, infracción, infracciones, naciones, Numero 8, anotar, pasar, robar, equipo, rival, venda, vendaje, botas, tacos, entre palos, sustituir, defensor, palos, touche, ruck, derribo, derribar, chute, chutar, lanzar, lanzó.

8.4.2.12 Tenis

Ball, pelota, break, saque, servicio, cannon ball, cañonazo, cortada, efecto, bola con efecto, deuce, drive, juego, game, spin, recogepelotas, liftado, lob, globo, bola rápida, golpe liso, net, red, passing shot, set ball, set point, punto, smash, esmach, esmachar, swing, balanceo, timing, individual, dobles, hierba, tierra, dejada, stop volley, court, pist, indoor, iguales, 30, 15, 40, falta, peloteo, individuales, bote, juez, master, mejor de tres, mejor de cinco, volea, juez de línea, dura, ladrillo, polvo, césped, in, out, tie-break, ace, reves, set, tie-break, batida, grand slam, tierra batida, final, puntos, eliminatoria, partido, venda, vendaje, sube.

8.4.2.13 Voleibol

Ball in, ball out, bolea, lok out, power service, servicio, esmachado, mate, set average, tie-break, Adelantado, alcance, árbitro, auxiliar, ataque, bandas, bola muerta, zona cambio, rotación, zaguera, anotaciones, anotación, bancos, zona entrenador, red, fuera, diagonal, envío, engaño, pasadores, pasador, central, libre, puesto, atacadores, auxiliares, finta, formacion, invasión, mas-menos, net, quinto set, recibidor, señal, atrasado, variillas, voleo, zaga, k-i, k-ii, punto, ser decisivo, derrota, final, remontar, potencia, vuelo, pista, anotar, anotación, sustituir, defensor.

8.4.3. Palabras excluyentes

8.4.3.1 Atletismo

Neumático, portería, pelota, balón, placaje, boxes.

8.4.3.2 Baloncesto

Raqueta, guantes, rueda, portería, volea, césped, boxes, hierro, madera, chute, chutar, neumático.

8.4.3.3 Balonmano

Raqueta, canasta, césped, boxes, chute, chutar, neumático.

8.4.3.4 Boxeo

Neumático, ruedan, rueda, portería, portero, canasta, red, césped, placaje, boxes, madera, chute, chutar.

8.4.3.5 Ciclismo

Lanzó, pelota, jugador, jugadores, portería, canasta, volea, balón, lona, placaje, árbitro, madera, chute, chutar.

8.4.3.6 Fútbol

Raqueta, guantes, boxes, neumático.

8.4.3.7 Fútbol Americano

Raqueta, red, rueda, portería, portero, canasta, boxes, neumático.

8.4.3.8 Golf

Pelota, neumático, balón, portería, canasta, portero, placaje, raqueta, boxes, madera, equipo, balón, chute, chutar.

8.4.3.9 Judo

Neumático, pelota, balón, raqueta, boxes, chute, chutar.

8.4.3.10 Motociclismo

Portero, lanzó, pelota, jugador, jugadores, portería, canasta, volea, balón, lona, placaje, árbitro, Juegos olímpicos, hierro, chute, chutar.

8.4.3.11 Rugby

Raqueta, red, rueda, portería, canasta, boxes.

8.4.3.12 Tenis

Guantes, rueda, portería, canasta, placaje, boxes, madera, chute, chutar, portero, pneumático.

8.4.3.13 Voleibol

Raqueta, rueda, portería, canasta, boXes, portero, chute, chutar, madera.

8.4.4. Palabras vacías

el él ésta éste éste éstos últimas últimas último último a añadió aún actualmente adelante además afirmó agregó ahí ahora al algún algo alguna algunas alguno algunos alrededor ambos ante anterior antes apenas aproximadamente aquí así aseguró aunque ayer bajo bien buen buena buenas bueno buenos cómo cada casi cerca cierto cinco comentó como con conocer consideró considera contra cosas creo cual cuales cualquier cuando cuento cuatro cuenta da dado dan dar de debe deben debido decir dejó del demás dentro desde después dice dicen dicho dieron diferente diferentes dijeron dijo dio donde dos durante e ejemplo el ella ella ello ellos embargo en encuentra entonces entre era eran es esa esas ese esos está están esta estaba estaban estamos estar estará estas este esto estos estoy estuve ex existe existen explicó expresó fin fue fuera fueron gran grandes ha había habían haber hAbrá hace hacen hacer hacerlo hacia haciendo han hasta hay haya he hecho hemos hicieron hizo hoy hubo igual incluso indicó informó junto la lado las le les llegó lleva llevar lo los luego lugar más manera manifestó mayor me mediante mejor mencionó menos mi mientras misma mismas mismo mismos momento mucha muchas mucho muchos muy nada nadie ni ningún ninguna ningunas ninguno ningunos no nos nosotras nosotros nuestra nuestras nuestro nuestros nueva nuevas nuevo nuevos nunca o ocho otra otras otro otros para parece parte partir pasada pasado pero pesar poca pocas poco pocos podemos podrá podrán podría poner por porque posible próximo próximos primer primera primero primeros principalmente propia propias propio propios pudo pueda puede pueden pues qué que quedó queremos quién quien quienes quiere realizó realizado realizar respecto sí sólo se señaló sea sean según segunda segundo seis ser será serán sería si sido siempre siendo siete sigue siguiente sin sino sobre sola solamente solas solo solos son su sus tal también tampoco tan tanto tenía tendrá tendrán tenemos tener tenga tengo tenido tercera tiene tienen toda todas todavía todo todos total tras trata través tres tuvo un una unas uno unos usted va vamos van varias veces ver vez y ya yo la en A B C D E F G H I J K L M N Ñ O P Q R S T U V W X Y Z.

8.4.5. Palabras en análisis del sentimiento

8.4.5.1 Alegría

Gozo, contento, deleite, diversión, placer, gratificación, satisfacción, euforia, éxtasis, emoción, jovialidad, felicidad, euforia, contento, triunfo, fascinación, dicha, alegría, júbilo, entusiasmo, estímulo, impaciencia, alivio, alborozo, deleite, jolgorio, satisfacción, esperanza, regocijo, humor, brío, agradecimiento, excitación, orgullo, embeleso, emocionado, sensual, energético, alegre, creativo, esperanzado, atrevido, estimulante, divertido, optimista, agrado, aleluya.

8.4.5.2 Amor

Adoración. atracción. sentimentalismo. añoranza. afecto. cuidado. deseo. amor. ternura. pasión. cariño. compasión. capricho. simpatía, adonis, adoración, apego, ardor, arrebato, celos, compromiso, coqueteo, cortejar, cortesía, desear, encandilar, excitación, fervor, flechazo, galán, frenesí, gustar, idolatrar, idilio, ligar, noviazgo, predilección.

8.4.5.3 Enfado

Rabia, enojo, resentimiento, furia, exasperación, indignación, animosidad, irritabilidad, irritación, hostilidad, odio, violencia, malhumor, amargura, venganza, desprecio, exasperación, furia, odio, desagrado, envidia, inquietud, frustración, cólera, aversión, resentimiento, fastidio, enfado, hostilidad, menospicio, repugnancia, aspereza, ira, violencia, rencor, distante, sarcástico, frustrado, celoso, escéptico, herir, hostil, egoísta, odioso.

8.4.5.4 Miedo

Angustia, alarma, aprensión, fobia, pánico, preocupación, desasosiego, incertidumbre, ansiedad, inquietud, terror, nerviosismo, aflicción, shock, pánico, tensión, pavor, miedo, histerismo, desasosiego, susto, humillación, preocupación, horror, ansiedad, horrorizado, humillado, preocupado, nervioso, inquietado, incierto, angustiado, aterrado.

8.4.5.5 Sorpresa

Asombro, sorpresa, pasmo, confuso, desconcertado, chasco, commoción, estupefacción, estupor, exclamación, extrañeza, impresión, pasmo, sobresalto, susto, asombrado, confusión, confundido, conmocionado, estupefacto, impresionado, pasmado, consternación, desorientado, desconcierto, trauma, embelesamiento, anomalía, singularidad, peculiaridad, impacto.

8.4.5.6 Tristeza

Auto compasión, soledad, desaliento, melancolía, depresión, aflicción, pena, desconsuelo, pesimismo, desesperación, tormento, pesimismo, pesar, decepción, remordimiento, rechazo, bochorno, sufrimiento, congoja, disgusto, alineación, humillación, depresión, suplicio, culpa, aislamiento, derrota, insulto, tristeza, melancolía, vergüenza, abandono, desánimo, lástima, desesperación, infelicidad, dolor, desaliento, arrepentimiento, soledad, inseguridad, condolencia, pesadez, arrepentido, estúpido, inferior, aislado, apático, soñoliento, guilty, avergonzado, Deprimido, solitario, aburrido, cansado.

