INSTITUTO POLITÉCNICO NACIONAL

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

Detección de acoso en mensajes de *Twitter*

T E S I S

QUE PARA OBTENER EL GRADO DE

MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA

Ing. Juan Carlos Ramos Márquez

DIRECTORES DE TESIS

Dr. Francisco Hiram Calvo Castro

M. en C. Sergio Sandoval Reyes

Ciudad de México, Enero de 2017

SIP-14 bis

INSTITUTO POLITÉCNICO NACIONAL
SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México siendo las 10:00 horas del día 06 del mes de diciembre de 2016 se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación del:

Centro de Investigación en Computación

para examinar la tesis titulada:

"Detección de acoso en mensajes de Twitter"

Presentada por el alumno:

RAMOS	**MÁRQUEZ**	**JUAN CARLOS**
Apellido paterno	Apellido materno	Nombre(s)

Con registro: B 1 4 0 4 8 7

aspirante de: **MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN**

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***APROBAR LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Directores de Tesis

Dr. Francisco Hiram Calvo Castro

M. en C. Sergio Sandoval Reyes

Dr. Grigori Sidorov

Dr. Miguel Jesús Torres Ruiz

Dra. Olga Kolesnikova

M en C. Sandra Dinora Orantes Jiménez

PRESIDENTE DEL COLEGIO DE PROFESORES

Dr. Marco Antonio Ramírez Salinas

INSTITUTO POLITÉCNICO NACIONAL
CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN
DIRECCIÓN

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESIÓN DE DERECHOS

En la Ciudad de México el día 15 del mes de diciembre del año 2016, el que suscribe Ing. Juan Carlos Ramos Márquez alumno del Programa de Maestría en Ciencia de la Computación con número de registro B140487, adscrito al Centro de Investigación en Computación, manifiesta que es autor intelectual del presente trabajo de Tesis bajo la dirección del Dr. Francisco Hiram Calvo Castro y el M. en C. Sergio Sandoval Reyes y cede los derechos del trabajo intitulado Detección de acoso en mensajes de Twitter, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección juan.c.ramos.m@gmail.com. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

Juan Carlos Ramos Márquez

Nombre y firma

Resumen

El presente trabajo propone un estudio detallado sobre el contenido de los mensajes de la red social *Twitter*.

El objetivo principal es conocer el tipo de mensaje emitido por parte de los usuarios de la red social, esto es, si es un mensaje de acoso o no, entendiéndose por acoso todo aquel mensaje que perjudique la reputación de un usuario; por tanto se debe llevar a cabo un proceso de análisis minucioso de las palabras contenidas en el mensaje, ver cómo están relacionadas entre sí además de analizar el uso de símbolos especiales como emoticones y *emojis* y, de esta manera, poder categorizar dicho mensaje.

En el presente trabajo se define una metodología clara para abordar el problema en cuestión, desde obtener los datos hasta reportar la eficiencia de los algoritmos. Acerca del proceso de clasificación: se toman en cuenta los diversos clasificadores que se utilizan en la literatura para posteriormente hacer los diversos experimentos correspondientes para obtener la mejor calibración posible de acuerdo al problema en cuestión. Además, se propone unir los resultados obtenidos de dichos clasificadores y combinarlos en un método compuesto o ensamble. Utilizar un método compuesto para trabajar en el mismo problema tiene sentido porque no es muy costoso y potencia el resultado al compensar deficiencias de uno u otro método individual.

También se hará uso de las técnicas de procesamiento de lenguaje natural para representar los mensajes computacionalmente de forma eficiente; de esta manera se van a analizar y utilizar las que mejor se adapten al problema en cuestión.

Por último, se va a hacer un análisis de los resultados obtenidos empleando las métricas comunes a problemas de clasificación.

Abstract

This thesis proposes a detailed report on the content of the messages from the social network Twitter.

The main objective is to know the type of message sent by users of the social network. That is, if it is or not a message with harassment. We define harassment as a message that harms the reputation of a user. Therefore, a process of careful analysis of the words contained in the message should be carried and then classify them accordingly.

This work takes into account various classifiers used in the literature and evaluates them with an appropriate calibration according to the problem at hand. Also we propose to join the results of several classifiers and mix them in a compound or ensemble method. An ensemble method for this problem makes sense because it is not considerably more expensive and can improve the result by compensating deficiencies of either individual method.

This work also relies on natural language processing techniques to handle messages in a computationally efficient manner, so we will analyze and use techniques best suited to the problem at hand.

Finally, an analysis of the results obtained will be done using the metrics common to classification problems.

AGRADECIMIENTOS

Al Instituto Politécnico Nacional por haberme otorgado el privilegio de realizar mis estudios de posgrado y de pertenecer a una de las instituciones pilares de la educación en México.

Al Centro de Investigación en Computación por haberme dado la oportunidad de cursar este posgrado y por el apoyo académico y tecnológico para la elaboración de este trabajo.

Al Consejo Nacional de Ciencia y Tecnología (CONACYT) que como organización, me brindó el apoyo económico necesario a través de la beca de posgrado para la realización de este trabajo.

A mis directores de tesis: el Dr. Francisco Hiram y el M. en C. Sergio por sus enseñanzas, tiempo, dedicación y apoyo en la revisión del presente trabajo pero más que nada gracias por su paciencia y comprensión.

A mis sinodales por su apoyo y tiempo en las correcciones necesarias de este trabajo. A la gran mayoría de ustedes, gracias por compartir su conocimiento y experiencia que tuve como estudiante en el Centro de Investigación en Computación.

A mis maestros por todas las buenas enseñanzas que me han brindado.

A mis amigos de maestría que, aunque son pocos, pero por esa razón los hace más valiosos, gracias por compartir sus conocimientos así como esos momentos de convivencia y alegría, ustedes saben quiénes son.

DEDICATORIA

A JEHOVA JIREH

Mi Dios quien me ha sostenido, ha sido fiel conmigo y me ha dado más de lo que merezco, Él ha sido mi fuerza para seguir adelante y gracias a su bendito amor me ha dado los dones de la sabiduría y el entendimiento para culminar este proyecto.

A MI ESPOSA BRENDA

Por el apoyo que me ha dado y por ser quién cuidó de mis peques en esta travesía, este es el fruto de tu esfuerzo y del tiempo que no estuve contigo.

A MIS HIJOS JUAN CARLOS Y PAOLA

Por todas las veces que no pudieron tener un papá de tiempo completo.

A MIS PADRES WENCESLAO Y MARGARITA

Por ser un gran ejemplo de trabajo, esfuerzo y dedicación. Tal vez nunca pueda pagarles todo lo han hecho por mí, gracias por instruirme desde pequeño.

A MIS HERMANOS WENCESLAO Y ROBERTO

Por el apoyo que me han dado ya que han estado ahí cuando los he necesitado.

Índice general

Lista de figuras

Lista de tablas

Capítulo 1

Introducción: El acoso en las redes sociales

En este capítulo se presenta la introducción al acoso en las redes sociales; la cual toma como base los mensajes que hacen los usuarios en la red social de *Twitter*. Los mensajes deben ser analizados por cada una de sus palabras y ver si en conjunto forman una proposición positiva o negativa para de esta manera clasificarlo como mensaje de acoso o no acoso. Los tipos de usuarios comunes en esta red social son: usuario *bully* (bravucón), usuario *troll* (molestan a todos y no aportan nada) y usuario ordinario (no molestan a nadie, solo hace uso de los servicios de la red social).

A partir de ello se formulan el problema de tesis, los objetivos y contribuciones a alcanzar, y la organización del documento de tesis.

1.1 Antecedentes

México ocupa el primer lugar internacional de casos de *bullying* (intimidación o acoso) en educación básica, ya que afecta a 18 millones 781 mil 875 alumnos de primaria y secundaria tanto públicas como privadas, de acuerdo con un estudio de la Organización para la Cooperación y el Desarrollo Económicos (OCDE, 2013).

El análisis efectuado por la OCDE entre los países miembros reporta que 40.24 por ciento de los estudiantes declaró haber sido víctima de acoso; 25.35 por ciento haber recibido insultos y amenazas; 17 por ciento ha sido golpeado y 44.47 por ciento dijo haber atravesado por algún episodio de violencia verbal, psicológica, física y ahora a través de las redes sociales.

El *bullying* se ha convertido en un severo problema ya que, conforme a la Comisión Nacional de los Derechos Humanos (CNDH), el número de menores afectados aumentó en los últimos dos años 10 por ciento, al grado de que siete de cada diez han sido víctimas de violencia (Valadez, 2014).

Investigaciones del Instituto Politécnico Nacional y de la Universidad Nacional Autónoma de México detallan que de los 26 millones 12 mil 816 estudiantes de los niveles preescolar, primaria y secundaria, alrededor de 60 y 70 por ciento ha sufrido *bullying* y aún cuando se carece de registros certeros, la ausencia de políticas para prevenir la violencia y el acoso escolar han derivado en bajo rendimiento, deserción, así como en un incremento de suicidio (Valadez, 2014).

1.2 Justificación

El problema del acoso en las redes sociales se ha convertido en un problema de salud nacional por lo que varias organizaciones han sumado esfuerzos para en alguna medida detectarlo y erradicarlo.

En la actualidad en México sólo se han tomado medidas a nivel físico-social, pero no se han generado herramientas tecnológicas que proporcionen alguna ayuda, salvo lo mencionado por (Castro & Váldez, 2012) donde se genera una herramienta la cual analiza en tiempo real los mensajes que son intercambiados a través de *chats*.

Por lo tanto, para poder tener un buen plan de acción para combatir el *bullying,* es indispensable contar con herramientas tecnológicas que permitan analizar cómo se comportan los individuos en los medios sociales, y de esta manera tomar medidas tanto preventivas como correctivas.

El software propuesto permite analizar el contenido en los mensajes de la red social *Twitter* y detectar si hay algún índice de acoso para que de esta manera se pueda tomar alguna acción al respecto.

1.3 Objetivos

1.3.1 Objetivo general

Revisar y aplicar algoritmos para analizar el contenido de los mensajes de la red social *Twitter* y de esta manera clasificarlos como mensajes de acoso o no acoso, usando técnicas de aprendizaje automático y procesamiento de lenguaje natural para el idioma español.

1.3.2 Objetivos particulares

- Revisar el estado del arte de los sistemas que hacen detección de acoso y analizar su estructura y funcionamiento.
- Estudiar las diversas técnicas de aprendizaje automático para lograr el mejor ajuste de parámetros.
- Estudiar las diversas técnicas de procesamiento de lenguaje natural que permitan representar los mensajes de la mejor manera posible.
- Definir una metodología para la detección de acoso en mensajes de texto (*tweets*).
- Implementar un sistema de detección de acoso para comprobar la validez de la metodología.

1.4 Planteamiento del problema

Diversos sistemas han sido desarrollados para llevar a cabo la tarea de detección de acoso es las distintas redes sociales, sin embargo, en su mayoría están dados en el idioma inglés lo cual no permite aplicarlos a otros idiomas y por ende a otros contextos e idiosincrasias. Hoy en día el tema de clasificación de textos o análisis de sentimientos en los contenidos de dichos textos es un área de gran interés de investigación. Es por ello que se pueden aplicar estas técnicas a un tema en específico como lo es el detectar acoso y ver la madurez y eficacia de dichas técnicas. Este trabajo de tesis se va a enfocar en un tipo específico de *bullying* llamado simplemente acoso, la Tabla 1.1 muestra otros tipos de acoso que quedan fuera del alcance de este trabajo.

Tipo de acoso	Definición
Acoso	Significa que el agresor envía mensajes ofensivos y maliciosos a un individuo o a un grupo y es a menudo repetido varias veces.
Pelea	Peleas en tiempo real usualmente a través de salas de chat, mensajería instantánea, correo electrónico, donde mensajes molestos y rudos son intercambiados.
Exclusión	Es el acto intencional de dejar fuera de una plática en línea a un usuario y posteriormente dejar comentarios maliciosos acerca de él.
Revelación	Es cuando el agresor comparte información, videos, fotos personales y privados acerca de alguien de manera pública. Una persona ha sido expuesta cuando su información se ha propagado por todo el Internet.
Suplantación	Pretender ser alguien cuando se envía información falsa.

Tabla 1.1 Tipos de acoso

1.5 Contribuciones

Las contribuciones de este trabajo son las siguientes:

- Se definió una metodología para la detección de acoso en mensajes de texto utilizando una combinación de diferentes técnicas de aprendizaje automático.
- Se definieron reglas para etiquetar un tipo específico de acoso.
- Se crearon dos corpus etiquetados para detectar acoso.
- Se definió un método efectivo para clasificar mensajes (69% exactitud).

- Se implementó un sistema de detección de acoso que demuestra la validez de la metodología propuesta.

1.6 Organización de la tesis

El presente trabajo de tesis de maestría se encuentra organizado de la siguiente forma:

En el primer capítulo se encuentran los antecedentes de la investigación, la justificación, el objetivo general, los objetivos particulares y las contribuciones que el presente trabajo aporta al desarrollo tecnológico en México.

En el segundo capítulo se presenta el Estado del Arte, es decir las investigaciones relacionadas en la detección de acoso en las redes sociales, basadas en diferentes técnicas de aprendizaje automático así como procesamiento de lenguaje natural. A partir de ellas se formula la propuesta de solución.

En el tercer capítulo se presenta el marco teórico, que abarca los cimientos teóricos que respaldan a este trabajo de tesis, por lo que se muestran las diferentes técnicas para tratar con información textual, así como de las técnicas de aprendizaje automático para categorización de texto y de igual manera se presentan métricas para la evaluación de la clasificación hecha.

En el cuarto capítulo se presenta una metodología sistemática y se diseña el sistema de análisis automático de contenido de mensajes de probable acoso en la red social *Twitter*.

En el quinto capítulo se implementa la propuesta para solucionar la problemática relacionada con el trabajo de tesis y de esta forma cumplir los objetivos plasmados en la misma, además se realizan las distintas pruebas de funcionamiento del sistema. Finalmente se presentan los resultados de las distintas técnicas de aprendizaje automático así como de procesamiento de lenguaje natural aplicados al problema en cuestión así como la meta-información de los conjuntos de datos de estudio.

En el sexto capítulo se presentan las conclusiones del presente trabajo de tesis basadas en los resultados y los objetivos a lograr, recomendaciones, aportaciones hechas y los trabajos futuros que pueden emerger del trabajo desarrollado.

Al final se presentan las referencias bibliográficas así como un glosario y los apéndices recabados durante la elaboración de la tesis.

Capítulo 2

Estado del arte: Soluciones afines y propuesta de solución

En el presente capítulo se describen los diferentes trabajos relacionados que se han publicado en relación al tema de detección de acoso en las redes sociales. Se incluyen los trabajos más relevantes de los últimos años. Además, se abordan algunas técnicas así como las herramientas empleadas que son de uso común para llevar a cabo dichas tareas. A partir de ello se formula la propuesta de solución.

2.1 Soluciones afines

2.1.1 Herramienta de mensajería instantánea para detección de acoso

Este trabajo es pionero en la detección de acoso en los contenidos de los mensajes instantáneos en el idioma español. Fue desarrollado en la universidad de Sonora en México (Castro & Váldez, 2012).

Este trabajo crea un prototipo de software basado en la arquitectura de agentes *ARSEC-AMS*[1] para el diseño de sistemas de seguridad. Básicamente el sistema se enfoca en la extracción de datos en el intercambio de mensajes en sesiones de *chat* que son usados en los salones de clases, y emite mensajes de alerta cuando considera que puede haber contenido perjudicial.

La metodología seguida fue construir un diccionario el cual contiene palabras que representan amenazas así como palabras que por sí mismas no representan un ataque. Posteriormente los mensajes capturados se dividen en sus palabras y se pre procesan, esto con el objetivo de considerar aspectos de errores de escritura y abreviaciones. Más adelante con las palabras "limpias" se comparan con el diccionario y de estos resultados se pueden identificar patrones que posiblemente signifiquen un mensaje de acoso. Para llevar a cabo la evaluación de las frases se usa un módulo de conocimiento programado a través de predicados o reglas tipo *prolog* para determinar si hay patrones en los mensajes.

Como se mencionó anteriormente la herramienta para controlar el flujo de la aplicación está basada en la arquitectura *ARSEC-AMS* como se muestra en la Figura 2.1.

[1] Security Architecture for Multi-Agent System.
http://www.world-academy-of-science.org/worldcomp11/ws

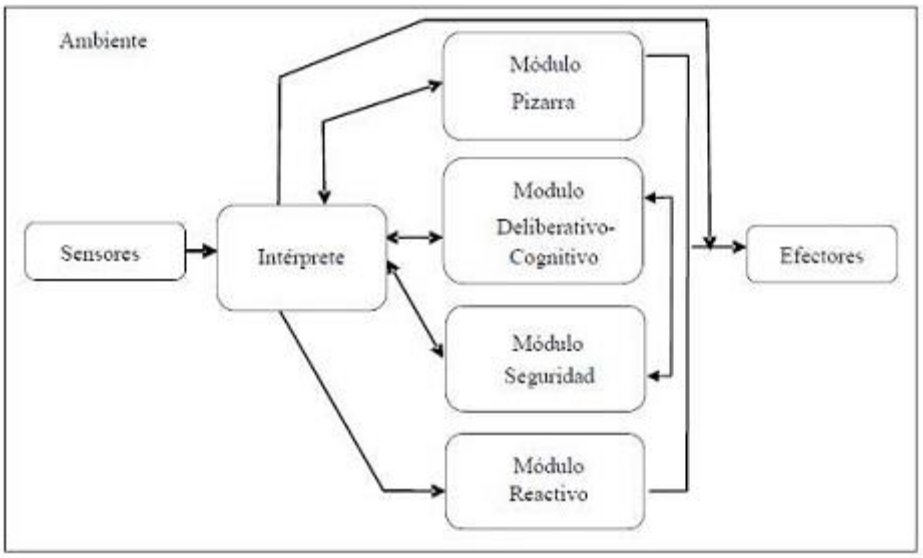

Figura 2.1 Arquitectura ARSEC-AMS

Esta arquitectura contempla el uso de cuatro módulos: reactivo, deliberativo-cognitivo, de seguridad y de control de pizarra, de tal forma que las señales que se reciben en el sistema de agentes a través de sensores, son filtradas por un intérprete que se encarga de actuar en forma coordinada, y dar un seguimiento al conjunto de acciones a realizar para alcanzar los objetivos correspondientes, aplicando esto en el modelo para detectar la peligrosidad de los mensajes intercambiados a través de un sistema de Mensajería Instantánea (MI).

Para llevar un buen seguimiento de las acciones que se realizan se implementan tres agentes, a continuación se describe la función de cada uno de ellos:

• Agente Coordinador: Es el agente software encargado del flujo de las transacciones que permiten aplicar niveles de seguridad en el sistema y en el trabajo de los agentes locales y de red. Además, se encarga de analizar el contenido de los mensajes, y en su caso registrar las alertas correspondientes para el administrador.

• Agente Local: Es el agente software encargado de analizar y revisar los mensajes de los usuarios de forma local.

• Agente de Red: Es el agente encargado de monitorear los mensajes enviados por el usuario y de hacerlos llegar al sistema ya cifrados.

Cabe mencionar que esta solución es de tipo caja negra, en el sentido que no se conocen las reglas o predicados tipo *prolog* para detectar si un mensaje contiene patrones de *bullying*. También es interesante notar que no usaron técnicas de aprendizaje automático, pero si se usó el idioma español como lengua para la detección del acoso.

2.1.2 Clasificación de *Tweets* usando un clasificador de texto para detectar acoso

En este artículo (Nalini & Sheela, 2015), los autores proponen un enfoque efectivo para detectar mensajes de *cyber bullying* de *Twitter* a través de un esquema ponderado de selección de características.

El método de detección identifica mensajes, acosadores y víctimas.

El método se divide en dos fases:

- Detectar mensajes de acoso: para realizar esto primero, se deben seleccionar qué características se van a usar para llevar a cabo la clasificación de si el mensaje es de acoso. Básicamente los autores usan dos tipos de características. El primer tipo son las características ponderadas o lo que coloquialmente se conoce como palabras de lenguaje obsceno. Para el segundo tipo se usa un algoritmo que extrae las características latentes de los documentos, para ello se usa la técnica *LDA* (*Latent Dirichlet Allocation*, Localización de Dirichlet Latente). Se pude decir que este es el núcleo del trabajo.
- Como segunda fase dado que ya se tiene la clasificación de los tweets, se crea un grafo para identificar a los acosadores y las víctimas.

Las aportaciones de este trabajo son tres: 1) Un nuevo enfoque el cual se basa en el esquema TF·IDF (*Term frequency - Inverse document frequency*,frecuencia de término - frecuencia inversa de documento) y el uso de *LDA* para identificar características latentes. 2) Se presenta un modelo gráfico para detectar a los usuarios más activos y agresores. 3) Los experimentos demuestran que es un método efectivo.

Como método de clasificación se usaron las máquinas de soporte vectorial (*Support Vector Machines*, un tipo de red neuronal) con una función de mapeo lineal.

El conjunto de datos se obtuvo a través de la *API stream de Twitter*[2] la cual da acceso en tiempo real a los mensajes publicados.

La solución presentada por este trabajo se acerca mucho a la planteada en esta tesis, a que se usan conceptos de lenguaje natural así como de aprendizaje automático. En la solución que se establece en este trabajo de investigación, se van a usar junto a las máquinas de soporte vectorial, otros algoritmos de aprendizaje automático mezclados en una técnica de ensamble (véase sección 4.1.5) además de utilizar como idioma el lenguaje español.

2.1.3 Análisis de sentimientos para detección efectiva de *Cyber bullying*

En este trabajo (Nahar, Unankard, Li, & Pang, 2012) al igual se pretende detectar en los mensajes de Twitter si hay algún indicio de acoso.

[2] https://dev.twitter.com/

La metodología se divide en dos etapas, a saber:

- Detectar mensajes perjudiciales basados en forma híbrida, es decir, tener características comunes que son las palabras que representan obscenidades y características de sentimientos.
- Analizar redes sociales para identificar abusadores y víctimas a través de las interacciones presentando un modelo de grafo.

Las características se modelan con el enfoque de bolsas de palabras.

Como en el trabajo anterior se empleó un modelo generativo a saber *PLSA* (*Probabilistic Latent Semantic Analysis*, análisis probabilístico de semántica latente), a los mensajes de *bullying* para identificar las características de sentimientos.

También se ocupó una máquina de soporte vectorial para la clasificación usando la biblioteca *libsvm*[3] y una validación cruzada de 10 pliegues.

El conjunto de datos fue extraído de tres sitios: 1) *Kongregate*: es un sitio de juegos en donde se puede asumir que los jugadores usan palabras agresivas. 2) *Slashdot*: foro para difundir mensajes, y 3) *Myspace*, red social.

La solución presentada por este trabajo es muy parecida al trabajo anterior ya que al igual emplea un modelo generativo (*PLSA*) y también utiliza máquinas de soporte vectorial, pero al igual que la anterior el idioma sobre el que se aplica es el inglés. Además, sólo aplica una técnica de aprendizaje automático, mientras que en la solución que se presenta en esta investigación se emplean varias.

2.1.4 Detección de *bullying* en *Twitter*

En este trabajo (Sanchez & Kumar, 2012), el objetivo fue comprender el *bullying* en las redes sociales y para este propósito se usó la red social *Twitter*.

Como clasificador de texto se utilizó el algoritmo de *Naïve Bayes*[4]. Para entrenar el clasificador se usó supervisión lejana. La precisión fue del 70% cuando se usó un conjunto de datos que contenían términos comúnmente usados en la intimidación.

La contribución principal del artículo fue hacer uso de análisis de sentimientos para detectar el *bullying*. Además de analizar los mensajes, se visualiza cómo estas instancias de *bullying* evolucionan y se comportan. Las gráficas muestran grupos de bravucones a través del tiempo.

Como producto se creó un *software* que infiere y visualiza casos de intimidación en *Twitter*, pero sólo se aplicó al idioma inglés, y en específico en el estado de California en los Estados Unidos.

El trabajo se enfocó a *bullying* de género y para esto se usaron las palabras: *gay*, *homo* (maricón-joto), *dike* (machorro, lesbiana, tortillera), *queer* (marica, mariposón,

[3] http://www.csie.ntu.edu.tw/~cjlin/libsvm/

[4] http://es.wikipedia.org/wiki/Clasificador_bayesiano_ingenuo

rarito). Y la única palabra con polaridad positiva fue: *queer,* Esta última palabra aunque a primera instancia parece todo lo contrario, según (Committee for children, 2016) tiene en la actual generación una connotación de sentimiento positivo.

Se siguió el rastro de los *bullies* y sus seguidores ya que éstos se sienten más robustos cuándo están en grupo.

Metodología: Una vez que se detecta un *tweet* negativo se confirma con un servicio de Amazon llamado el "Turco Mecánico" (*Mechanical Turk*)[5]. Si se confirma se extraen datos del abusador así como de la víctima y además de los seguidores del bravucón, es decir se monitorea el grafo social del bravucón como de sus seguidores para ver si están acosando a la víctima. Por tanto, la salida del monitoreo serán varios grafos uno por bravucón. Se usó Gephi para mostrar los grafos.

Además, se muestra otro grafo entre el bravucón y la víctima para encontrar grupos de bravucones, y conexiones ocultas de víctimas.

Se recolectaron 5000 *tweets*, se construyó un *framework* usando *LingPipe* (*LingPipe* es un *kit* de herramientas para procesamiento de texto empleando lingüística computacional) y como línea base de clasificación se usó *Naïve Bayes*.

Se construyó un extractor de *tweets*, en el que un *framework* extrae y clasifica en *streaming*. De los *tweets* recolectados en base a las palabras de interés se quitaron los que no expresan opiniones como los de noticias.

Después se extrajeron las palabras del *tweet* y se almacenaron como un vector de características. Los valores para cada característica pueden ser binarios para indicar la presencia o ausencia o un entero para indicar la intensidad de la palabra. Asimismo, se incluyó el modelo de bolsas de palabras.

El extractor se construyó con las bibliotecas *open source Twitter4J* y la *Twitter streaming API*, y se configuró para que extrajera *tweets* de california en inglés conteniendo las 4 palabras.

El conjunto de entrenamiento se obtuvo con el extractor buscando las palabras *gay, homo, dike y queer* luego manualmente se etiquetaron los *tweets* junto con el servicio del Turco Mecánico de Amazon.

La solución presentada por este trabajo proporciona muchas pautas de cómo recolectar los datos al mencionar el uso de librerías de acceso a *Twitter*. También se puede apreciar que se trabaja con herramientas de inteligencia humana como el Turco Mecánico de Amazon para formar el conjunto de entrenamiento. Finalmente, cabe destacar que también se usan herramientas de aprendizaje automático como el algoritmo de *Naïve Bayes* aunque en el idioma inglés.

[5] https://www.mturk.com/mturk/welcome

2.1.5 Aprendizaje automático para la detección de perfiles bravucones en la red social *Twitter*: aplicación a un caso real.

En este trabajo (Galán-García, Gaviria, Laorden, Santos, & García, 2013). Se presenta una metodología para detectar un perfil falso en *Twitter* al analizar el contenido de los comentarios y asociarlo a una cuenta real.

Básicamente la metodología se basó en un caso real de acoso en una escuela de Bilbao España, donde un usuario con perfil falso empezó a hacer comentarios negativos en la red de *Twitter* acerca de las clases y los maestros.

Para la clasificación se usó la herramienta de *Weka* con diferentes algoritmos de aprendizaje automático como: árboles predictores, árboles de decisión, K-vecinos más cercanos, basados en el teorema de Bayes y *SMO* (*Sequential minimal optimization*, Optimización mínima secuencial). Al igual se usó validación cruzada.

Para modelar los mensajes se usó el modelo de espacio vectorial con TF·IDF, algo importante a destacar es que aquí se usó el idioma español y por tanto los *tweets* se filtraron con *stop words* en español.

2.2 Solución Propuesta

Con base en lo analizado en los artículos antes discutidos, la propuesta es analizar el contenido de los mensajes de la red social *Twitter* y determinar, si es un mensaje de acoso o no. Para llevar a cabo dicha clasificación se hará uso de los siguientes algoritmos de aprendizaje automático: *Naïve Bayes*, regresión logística y máquinas de soporte vectorial. Todos ellos se combinarán en un algoritmo de ensamble; dicho enfoque ayuda a mejorar los resultados del sistema, y más aún, en la literatura aún no se encuentra dicho enfoque aplicado específicamente a la detección de mensajes de acoso, el cual puede dar buenos resultados al combinar varias técnicas que en la literatura se tratan por separado. Además, cabe mencionar que se hará uso de técnicas de procesamiento del lenguaje natural para modelar, es decir, representar los mensajes eficientemente para que los algoritmos de aprendizaje automático puedan sacar el mejor provecho de las características de dichos mensajes. Finalmente, basta mencionar que estas técnicas que comúnmente se aplican al idioma inglés en este trabajo de tesis se aplican al idioma español y aunque cada idioma tiene sus propias características se espera que el proceso que se va a utilizar dé buenos resultados.

Por tanto, a grandes rasgos se puede concluir que este trabajo de tesis comparado con los otros mostrados anteriormente, se diferencia en que se usan varias técnicas de aprendizaje automático y se combinan los distintos resultados para dar lugar a uno solo. Además, la detección de mensajes de acoso, se aplica al idioma español con respecto a los otros que se basan en el idioma inglés; también se emplean técnicas de procesamiento del lenguaje natural para manejar las características propias de dicho idioma tales como ambigüedad en el uso de las palabras, errores gramaticales, omisión o mal uso de signos de puntuación, uso de barbarismos, etc.

Otra diferencia está en la representación de los mensajes para entrenar el clasificador; en otros trabajos se hace uso de la representación por bolsa de palabras o n-gramas clásicos, en este trabajo se hará uso de los n-gramas sintácticos, véase sección 3.4.2. En la Figura 2.2 se muestra gráficamente la solución propuesta.

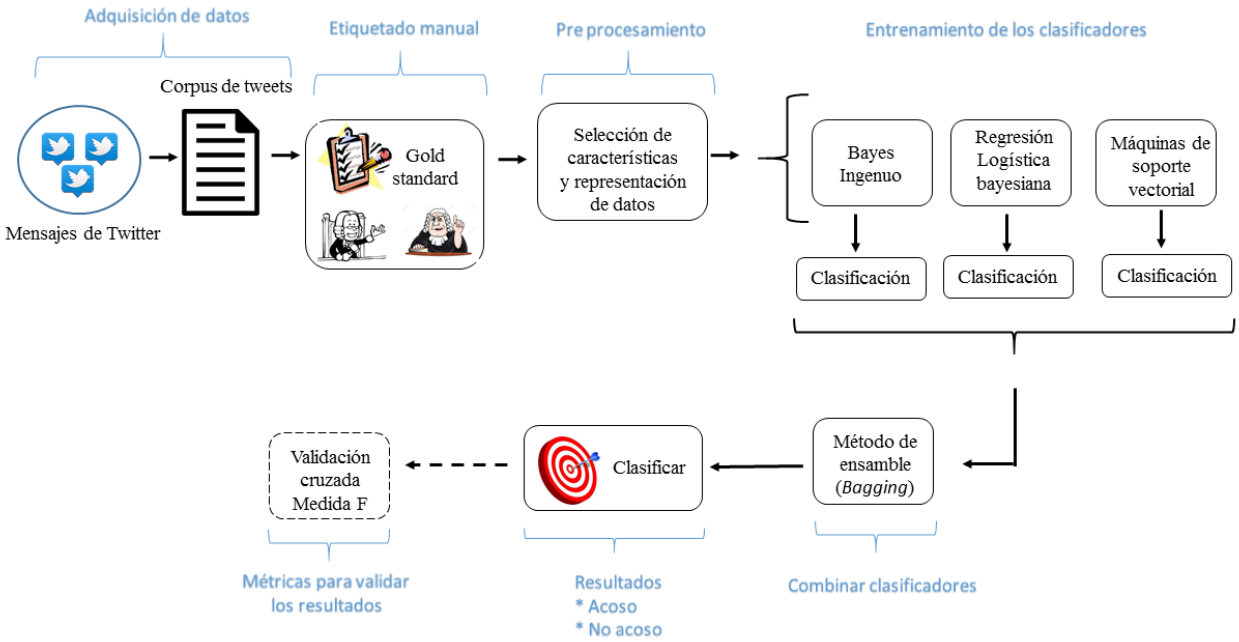

Figura 2.2 Solución propuesta

Grosso modo se pueden apreciar en la Figura 2.2 las distintas etapas las cuales se describen más abajo de manera breve, ya que en el próximo capítulo se profundizará en cada una de ellas:

- Adquisición de datos: Recolectar los mensajes de *Twitter*.
- Etiquetado manual: Tres jueces clasificarán los mensajes como acoso o no acoso.
- Pre procesamiento: Quitar ciertos elementos como por ejemplo signos de puntuación los cuales no aportan información útil al clasificador además de elaborar una representación útil y entendible para pasarlo como entrada a los clasificadores.
- Entrenamiento de los clasificadores: Dada una representación de los mensajes, pasarlos como entrada a los 3 clasificadores y entrenarlos para que generen un modelo que posteriormente será usado para clasificar.
- Combinar clasificadores: Dada la salida de cada clasificador juntarlas y producir una única salida.

- Resultados: Dada una entrada como un mensaje de *Twitter* clasificarlo como mensaje de acoso o no acoso. Cabe destacar que es imprescindible medir la validez de dichos resultados, para esto se usarán las medidas de precisión, especificidad y medida F.

2.3 Resumen del capítulo

En este capítulo se revisaron varias técnicas para llevar a cabo la clasificación de un ejemplo (mensaje) ya sea de acoso o no acoso y se pudo apreciar que se ha hecho un uso extensivo del área de aprendizaje automático. Dentro de los algoritmos más utilizados están el de *Naïve Bayes* y las máquinas de soporte vectorial. También es común llevar a cabo un pre procesamiento de los datos para dejarlos limpios o normalizados y en este tenor entra en juego el área de procesamiento del lenguaje natural para representar los datos de una mejor manera.

Por último, en el siguiente capítulo se presentan los fundamentos teóricos en los que descansa esta tesis, con base en el problema planteado al principio de la misma. El valor de este marco de referencia es que apoya en las formulaciones iniciales, además da pie para aventurarse en plantear hipótesis y sus probables resultados así como servir de guía en la presente investigación.

Capítulo 3

Marco Teórico: Conceptos y definiciones

En este capítulo se abordan los conceptos y técnicas que dan sustento a este trabajo de investigación. Primeramente, se abordarán los modelos comunes para representar mensajes de texto a vectores numéricos que son en definitiva los elementos de entrada a un clasificador. Después se verán las herramientas de construcción de los n-gramas y n-gramas sintácticos. Antes de construir un n-grama sintáctico debe tenerse un árbol sintáctico para, a partir de éste, generar el correspondiente n-grama; por ende, también se revisan las herramientas para construir estos elementos, así como los fundamentos de los algoritmos de aprendizaje supervisado.

3.1 Modelos de representación de información textual

Antes de poder hacer uso de un algoritmo de aprendizaje automático es importante poder representar la información de una manera que el algoritmo la pueda entender. A continuación se describen algunos de estos conceptos.

3.1.1 Modelo de bolsa de palabras

El modelo bolsa de palabras (*Bag of Words*) es un método popular que se utiliza en el procesado del lenguaje natural para representar documentos sin tener en cuenta la gramática e incluso el orden de las palabras. En este modelo, cada documento parece una bolsa que contiene algunas palabras. Por lo tanto, este método permite un modelado de las palabras basado en diccionarios, donde cada bolsa contiene unas cuantas palabras del diccionario. Las principales ventajas de utilizar este modelo son su facilidad de uso y su eficiencia computacional.

En este modelo el procesamiento de los documentos consta de las siguientes etapas:

- Pre procesado de los documentos: consiste fundamentalmente en preparar los documentos para su parametrización, eliminando aquellos elementos que se consideran superfluos.
- Parametrización: es una etapa de complejidad mínima una vez que se han identificado los términos relevantes. Consiste en realizar una cuantificación de las características (es decir, de los términos) de los documentos.

Grosso modo, el modelo bolsa de palabras aprende un vocabulario (diccionario) de todos los documentos y después modela cada documento asignando un valor a cada una de estas palabras cuantificándolas.

Se clarifican estas ideas mediante el siguiente ejemplo:

- **Documento 1**: A Juan le gusta ver películas. A María también le gusta.
- **Documento 2:** A Juan además le gusta ver partidos de futbol.

Basado en estos dos documentos se puede construir el siguiente vocabulario o diccionario: {A, Juan, le, gusta, ver, películas, María, también, además, partidos, de, futbol}. Este vocabulario tiene 12 distintas palabras. Ahora, usando los índices del diccionario, cada documento es representado por un vector de 12 elementos.

Para obtener los valores para cada elemento de los vectores, pueden aplicarse diferentes tipos de cuantificación sobre los términos, por ejemplo uno (tipo de cuantificación) sería obtener el número de veces que aparece el término en el documento TF (*Term Frecuency*, frecuencia del término). Para el ejemplo anterior pueden construirse los siguientes dos vectores:

Documento 1: {2, 1, 2, 2, 1, 1, 1, 1, 0, 0, 0, 0}

Documento 2: {1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1}

Cada entrada de las listas se refiere al conteo de la correspondiente entrada del diccionario. Por ejemplo en el primer documento las primeras dos entradas son “2, 1”. La primera entrada corresponde a la palabra “A” que es la primera entrada en el diccionario y tiene valor 2 porque “A” aparece 2 veces en el primer documento. Similarmente, la segunda entrada corresponde a la palabra “Juan” la cual es la segunda palabra en el diccionario y su valor es 1 porque “Juan” aparece una vez en el primer documento. Estas representaciones de vectores no preservan el orden de las palabras en el documento original, lo cual es la principal característica de este modelo. Este tipo de representación tiene varias aplicaciones satisfactorias, por ejemplo, el filtrado de correo electrónico.

Sin embargo, la cuantificación sobre las características (términos) que se aplicó arriba (frecuencia de término) no es necesariamente la mejor representación para el texto. Palabras como “a”, “el”, “que” son mayormente comunes y por ende tienen las frecuencias de término más altas, pero tener un conteo alto no necesariamente significa que la palabra correspondiente es más importante. Para manejar este problema, uno de los caminos más populares es normalizar la frecuencia de término, o utilizar alguna otra medida de cuantificación como IDF (*Inverse Document Frequency,* Frecuencia Inversa de Documento).

3.2 Técnicas de pre procesamiento de la información

En esta sección se tratarán algunas técnicas comunes para la limpieza del texto utilizadas en el área de procesamiento del lenguaje natural: *stemming*, eliminación de *stop words*, lematización.

3.2.1 *Stemming*

El *Stemming* es la técnica de eliminar las distintas flexiones de una palabra tales como sufijos o prefijos de tal manera que la fracción restante de la palabra después de este proceso se le conoce como raíz o lexema (*stem*).

En dicha tarea de stemming se aplican un conjunto de reglas que permiten esta simplificación, por lo que esta actividad de encarga de obtener la raíz de la palabra, la cual no necesariamente tiene un significado. Por ejemplo, en la Tabla 3.1 se tiene el caso para las distintas flexiones que se pueden dar para la raíz cas.

Flexiones	Raíz
casa, casas, casitas	cas

Tabla 3.1 Ejemplo de stemming

3.2.2 Lematización

La lematización es una técnica que lleva a cabo un análisis morfológico de tal manera que lleva la palabra a su forma base (lema en español, o lemma en inglés) (Manning et al., 2009). A diferencia del stemming, esta palabra sí contiene un significado y se encuentra como entrada en los diccionarios. Un ejemplo de este proceso es el mostrado en la Tabla 3.2:

Flexiones	Lema
trabajo, trabajaste, trabaje	trabajar

Tabla 3.2 Ejemplo de Lematización

Ambos procesos tanto el stemming como el de lematización son útiles en áreas como recuperación de la información, ya que llevando la palabra a su lema o raíz es más fácil que se encuentre una coincidencia. Una desventaja es la reducción en la eficiencia de los sistemas, la cual se ve afectada debido a los recursos requeridos para aplicar estos procesos. Existen diversos algoritmos para hacer el stemming; sin embargo, uno de los más populares es el de Porter (Porter, 1980), el cual utiliza un conjunto de reglas con el propósito de eliminar determinados sufijos de las palabras. Dicho algoritmo es popular por su rapidez y buenos resultados.

3.2.3 Eliminación de palabras de no contenido (*stop words*)

Otra técnica común en el área de procesamiento de lenguaje natural es la eliminación de las palabras de no contenido (*stop words*). Están palabras hacen referencia a palabras que aparecen muchas veces en el documento tales como las palabras conectoras, artículos, pronombres, preposiciones, etc. pero que no aportan información relevante. El apéndice 1 tiene una lista de dichas palabras de no contenido en español usadas en este trabajo.

3.3 Aprendizaje automático

En Ciencias de la Computación, el aprendizaje automático es una rama de la Inteligencia Artificial cuyo objetivo es desarrollar técnicas que permitan a las computadoras aprender. De forma más concreta, se trata de crear programas capaces de generalizar comportamientos a partir de una información no estructurada suministrada en forma de ejemplos. Es, por lo tanto, un proceso de inducción del conocimiento, es decir, un método que permite obtener por generalización un enunciado general a partir de enunciados que describen casos particulares.

Una definición formal de aprendizaje automático es propuesta por Tom Mitchell (Mitchell, 1997), y ésta dice que una máquina aprende si es capaz de utilizar su experiencia tal que su rendimiento mejore en experiencias similares en el futuro.

Ahora bien todo proceso básico de aprendizaje puede ser dividido en 4 componentes básicos como lo menciona Lantz (Lantz, 2015):

- Almacenamiento de datos: utiliza observación, memoria, y recuerdos para proveer un fundamento basado en hechos para el razonamiento a futuro.
- Abstracción: Involucra la translación de datos almacenados en representaciones y conceptos más amplios.
- Generalización: Usa datos abstractos para crear conocimiento e infiere qué acción tomar en nuevos contextos.
- Evaluación: Provee un mecanismo de retroalimentación para medir la utilidad del conocimiento adquirido e informar de mejoras posibles.

Aplicar aprendizaje automático a un conjunto de información incluye seguir un proceso compuesto por las siguientes etapas como lo menciona Lantz (Lantz, 2015):

- Obtener los datos. Consiste en capturar y reunir los datos en un formato que permita su análisis.
- Limpiar los datos. Para limpiar los datos, éstos se deben explorar y estudiar, y luego pre procesarlos (aplicar diferentes técnicas para eliminar o transformar datos “sucios”) para mejorar su calidad.
- Entrenar un modelo de aprendizaje. Consiste en adaptar un modelo en particular al conjunto de datos de entrenamiento, evitando problemas como el sobre entrenamiento (*overfitting*).
- Predecir o clasificar con el modelo. Implica utilizar el modelo previamente entrenado y aplicarlo sobre un conjunto de datos de prueba.
- Evaluar el modelo. Involucra evaluar su desempeño y su exactitud, utilizando métodos estadísticos.
- Mejorar el desempeño del modelo. Para mejorar el desempeño del modelo hay que aplicar estrategias avanzadas, realizando otras preparaciones a los datos originales, o eligiendo otro modelo.

3.3.1 Tipos de algoritmos de aprendizaje automático

Aprendizaje supervisado. Se genera una función que establece una correspondencia entre las entradas y las salidas deseadas del sistema, donde la base de conocimientos del sistema está formada por ejemplos etiquetados a priori (es decir, ejemplos de los que se sabe la clasificación correcta). Como se puede apreciar en la Figura 3.1, se tiene una tabla de aprendizaje, donde a partir de ésta se debe generar un modelo o función de correspondencia entre un ejemplo y su etiqueta. Ahora cada ejemplo es un registro de dicha tabla y cada columna es un atributo o característica. Para obtener dicho modelo se aplica un algoritmo de aprendizaje que genere dicho modelo. Ya con este modelo generado se puede evaluar qué tan "eficiente" es evaluándolo con datos no vistos por el modelo.

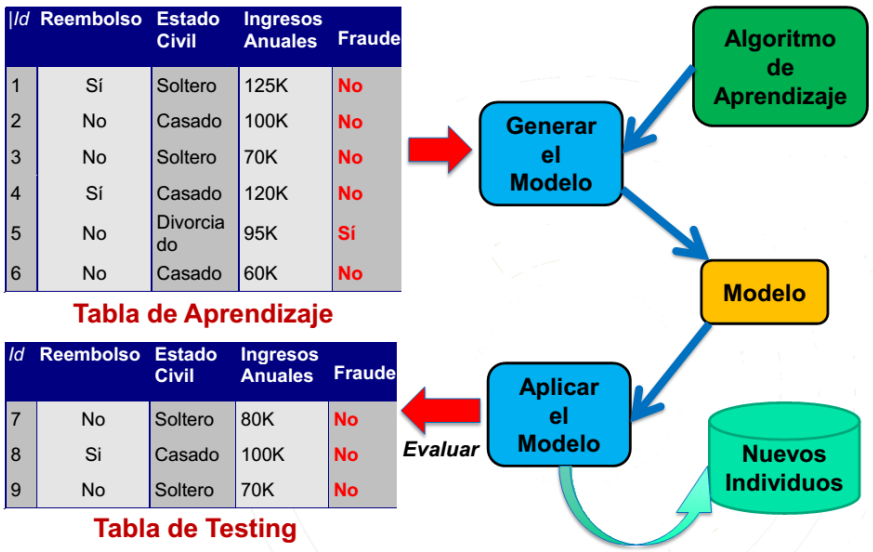

Figura 3.1 Aprendizaje supervisado

Aprendizaje no supervisado. El proceso de modelado se lleva a cabo sobre un conjunto de ejemplos formados únicamente por entradas al sistema, sin conocer su clasificación correcta. Lo que se busca es que el sistema sea capaz de reconocer patrones para poder etiquetar las nuevas entradas. Como se puede apreciar en la Figura 3.2, en la parte izquierda se tienen datos sin clasificar, es decir sólo se tienen los ejemplos y sus características, que en este caso específico corresponderían a un vector de dos dimensiones. La tarea aquí consiste en aplicar algún algoritmo y agrupar estos ejemplos. En la parte derecha de la Figura 3.2 puede apreciarse que se han creado 4 grupos, por lo que se tendrían 4 clases. A partir de aquí se pueden obtener ejemplos etiquetados.

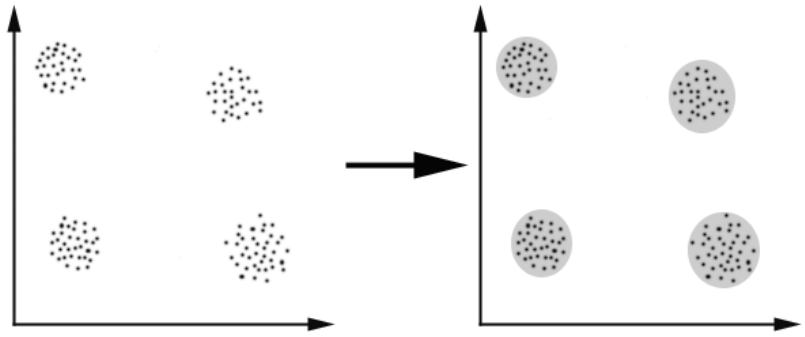

Figura 3.2 Aprendizaje no supervisado

Ya con los datos limpios y en una representación adecuada para que el clasificador los pueda entender se va a hacer uso de los siguientes clasificadores: *Naïve Bayes*, Máquinas de soporte vectorial y regresión logística.

3.3.2 Clasificador *Naïve Bayes*

En teoría de la probabilidad y minería de datos, un clasificador *Naïve Bayes* es un clasificador probabilístico fundamentado en el teorema de Bayes y algunas hipótesis simplificadoras adicionales. Es a causa de estas simplificaciones, que se suelen resumir en la hipótesis de independencia entre las variables predictoras, que recibe el apelativo de ingenuo.

En términos simples, un clasificador de *Naïve Bayes* asume que la presencia o ausencia de una característica particular no está relacionada con la presencia o ausencia de cualquier otra característica, dada la clase variable. Por ejemplo, una fruta puede ser considerada como una manzana si es roja, redonda y de alrededor de 7 cm de diámetro. Un clasificador *Naïve Bayes* considera que cada una de estas características contribuye de manera independiente a la probabilidad de que esta fruta sea una manzana, independientemente de la presencia o ausencia de las otras características.

La fórmula para calcular la mayor probabilidad de que dados ciertos valores para las características de un objeto pertenezca a cierta clase se muestra en la Ecuación 3.1. En dicha fórmula se puede apreciar cómo se tienen en base al teorema de Thomas Bayes, probabilidades distintas para cada clase a la que podría pertenecer un conjunto de características de un solo ejemplo, según esta fórmula se toma la mayor probabilidad. Se utilizaran las librerías de *NLTK*[6] para implementar el clasificador bayesiano.

[6] http://www.nltk.org/

$$v_{MAP} = argmax_{v_j \in V} \left(P(v_j \mid a_1, \ldots, a_n) \right)$$

Ecuación 3.1 Formula de Bayes Ingenuo

3.3.3 Máquinas de soporte vectorial

Las *SVM* (*Support Vector Machines*, máquinas de soporte vectorial o máquinas de vectores de soporte) son un conjunto de algoritmos de aprendizaje supervisado desarrollado por Vladimir Vapnik y su equipo en los laboratorios *AT&T* (Cortes & Vapnik, 1995).

Estos métodos están propiamente relacionados con problemas de clasificación y regresión. Dado un conjunto de ejemplos de entrenamiento (de muestras) pueden etiquetarse las clases y entrenar una *SVM* para construir un modelo que prediga la clase de una nueva muestra. Intuitivamente, una *SVM* es un modelo que representa a los puntos de muestra en el espacio, separando las clases por un espacio lo más amplio posible. Cuando las nuevas muestras se ponen en correspondencia con dicho modelo, en función de su proximidad pueden ser clasificadas a una u otra clase.

Más formalmente, una *SVM* construye un hiperplano o conjunto de hiperplanos en un espacio de dimensionalidad muy alta (o incluso infinita) que puede ser utilizado en problemas de clasificación o regresión. Una buena separación entre las clases permitirá una clasificación correcta. Se utilizó la librería *libsvm*[7], para implementar las máquinas de soporte vectorial. En la Figura 3.3 Se puede apreciar como la línea diagonal h es el plano que permite el mayor espacio de separabilidad entre las clases -1 y +1, además también se puede apreciar cómo las clases redondeadas por líneas intermitentes son los vectores de soporte.

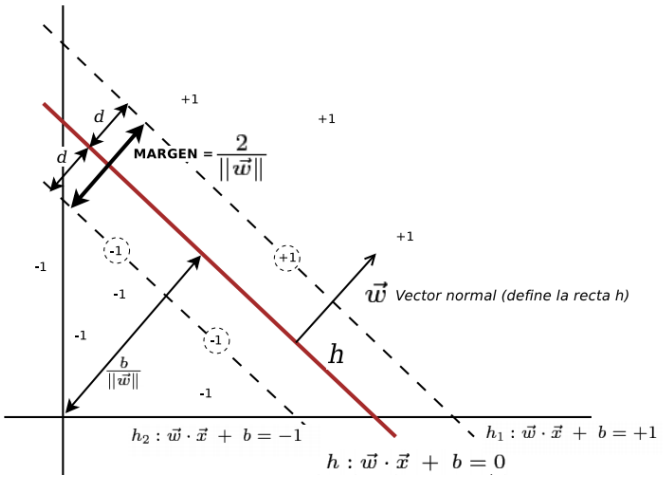

Figura 3.3 Plano de máximo margen

[7] https://www.csie.ntu.edu.tw/~cjlin/libsvm/

3.3.4 Regresión Logística Bayesiana

En estadística, la regresión logística es un tipo de análisis de regresión utilizado para predecir el resultado de una variable categórica (una variable que puede adoptar un número limitado de categorías) en función de las variables independientes o predictoras. Es útil para modelar la probabilidad de un evento ocurriendo como función de otros factores. El análisis de regresión logística se enmarca en el conjunto de Modelos Lineales Generalizados (*GLM* por sus siglas en inglés) que usa como función de enlace la función *logit.* Las probabilidades que describen el posible resultado de un único ensayo se modelan, como una función de variables explicativas, utilizando una función logística.

La regresión logística es usada extensamente en las ciencias médicas y sociales. Otros nombres para regresión logística usados en varias áreas de aplicación incluyen modelo logístico, modelo *logit*, y clasificador de máxima entropía. La Figura 3.4 muestra la forma que toma la función logística.

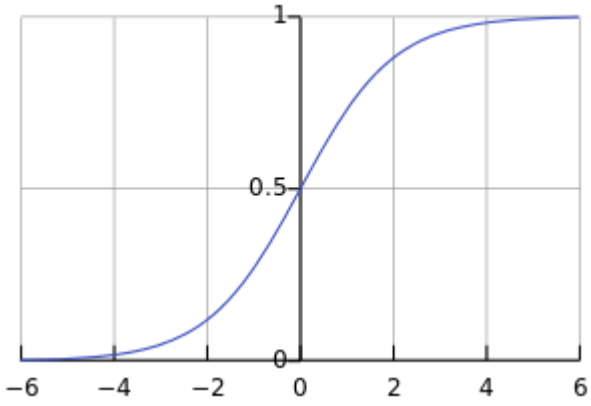

Figura 3.4 Función logística.

3.4 N-gramas y n-gramas sintácticos

En este apartado se presentan los n-gramas tradicionales y los n-gramas sintácticos, mencionando sus características y usos.

3.4.1 N-gramas

Una definición formal de n-gramas es la siguiente: un n-grama es una secuencia de *n* palabras de una expresión en lenguaje natural tal como aparecen en la expresión, y por lo general, se utilizan en aplicaciones de Procesamiento de Lenguaje Natural tales como corrección automática, reconocimiento de voz, traducción automática, etiquetado de categoría gramatical, generación de lenguaje natural, semejanza entre palabras, identificación de autoría entre otras (Jurafsky y Martin, 2009). En este caso *n* indica cuantos elementos deben tomarse, es decir, la longitud de la

secuencia o de n-grama. Por ejemplo, existen bigramas, trigramas, cuatrigramas (2-gramas, 3-gramas, 4-gramas), etc. En la Tabla 3.3 se muestra un ejemplo de las secuencias de 2 y 3 palabras (bigrama y trigrama) para la frase: “El árbol de mi casa está muy grande”.

N-grama	Secuencias
2 palabras – Bigrama	El árbol, árbol de, de mi, mi casa, casa está, está muy, muy grande.
3 palabras – Trigrama	El árbol de, árbol de mi, de mi casa, mi casa está, casa está muy, está muy grande.

Tabla 3.3 Ejemplos de ngramas obtenidos a partir de una oración.

3.4.2 N-gramas sintácticos

El enfoque de este trabajo de tesis está basado principalmente en el concepto de n-gramas sintácticos (*syntactic n-grams/sn-grams*) los cuales son n-gramas o secuencias de palabras (u otros elementos) obtenidas a partir del orden en el cual aparecen los elementos en los árboles de análisis sintácticos (dependencias o constituyentes) de una oración; más específicamente, los n-gramas sintácticos son construidos a partir de la secuencia de nodos que pueden ser alcanzados en cualquier camino de longitud n en un árbol de análisis sintáctico. Este tipo de n-gramas sintácticos es conocido como n-gramas sintácticos continuos (Sidorov et al., 2012).

Considérese la siguiente frase de ejemplo: “*El doctor Ferguson se ocupaba desde hacía mucho tiempo de todos los pormenores de su expedición*” tomada de (Sidorov, 2013). Para este ejemplo se construyó el árbol sintáctico de manera manual aunque existen programas que se encargan de esta tarea automáticamente (*parsers*). La Figura 3.5 muestra el árbol sintáctico creado para la frase anterior.

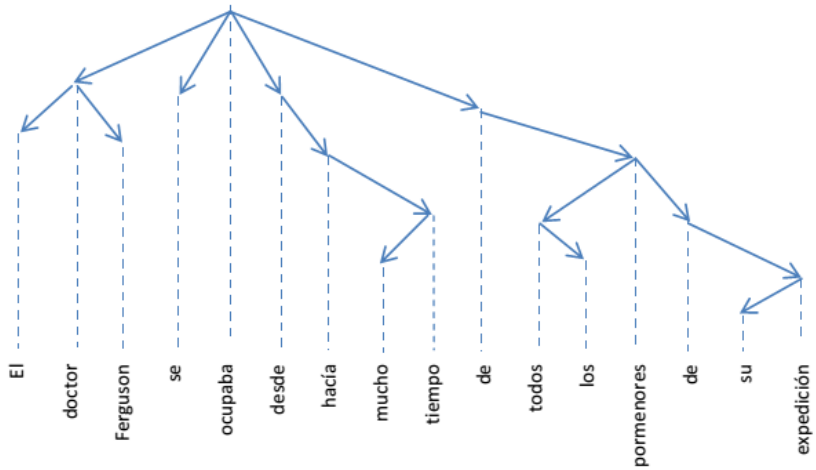

Figura 3.5 Ejemplo de un árbol sintáctico.

Ahora, del árbol sintáctico obtenido se pueden generar los correspondientes n-gramas como se muestra en la Tabla 3.4.

N-grama	Secuencias
2 palabras – Bigrama	ocupaba se, ocupaba doctor, ocupaba desde, ocupaba de, desde hacía, el Ferguson, etc.
3 palabras – Trigrama	ocupaba desde hacía, ocupaba de pormenores, ocupaba doctor el, etc.

Tabla 3.4 ejemplos de n-gramas sintácticos

3.5 Resumen del capítulo

En este capítulo se ha presentado un panorama con los fundamentos teóricos que respaldan a esta tesis.

Se han descrito conceptos relativos a la representación de datos, el peso de cada componente de los vectores. Se han mostrado algunas bases necesarias para entender al aprendizaje automático (particularmente los algoritmos de *Naïve Bayes*, Máquinas de Soporte Vectorial y regresión logística) como herramientas que pueden ayudar en la clasificación de acoso en los mensajes de Twitter. Finalmente, se revisaron las técnicas de los n-gramas y n-gramas sintácticos como características para los vectores de entrada a los clasificadores y con esto tener un poco más de información contextual que pueda ayudar en la precisión de los clasificadores.

A continuación se muestra la metodología utilizada para el desarrollo de este trabajo, es decir, el diseño del sistema y los procedimientos que se han considerado adecuados para sistematizar la investigación científica y el desarrollo del sistema propuesto, con el objetivo de cumplir con las metas planteadas en un principio.

Capítulo 4

Diseño del sistema de análisis y detección automática de mensajes de acoso en Twitter

En este capítulo se presenta la metodología empleada para la solución del problema afrontado en la presente tesis, es decir, la serie de pasos a seguir basados en la investigación científica con la finalidad de cumplir con los objetivos propuestos inicialmente. A continuación se presenta el desglose por etapas de las actividades que conducirán el trabajo, estudiando de manera independiente cada proceso necesario para lograr desde la etapa de la adquisición de los mensajes de la red social Twitter, pasando por el etiquetado de los mensajes, procesamiento, aplicación de los algoritmos de aprendizaje, hasta el proceso de clasificación y validación.

4.1 Etapas de la metodología propuesta

Como se mostró en la Figura 2.2, la cual representa la propuesta de solución las etapas que se van a tratar para dar solución al problema propuesto son: adquisición de datos, etiquetado de los datos de forma manual, pre procesamiento de los datos y representación de los mismos, entrenamiento de los clasificadores, combinar clasificadores y resultados. Estas etapas se detallan en las siguientes secciones.

4.1.1 Etapa de adquisición de datos

El primer paso a realizar es la obtención de un *corpus* con el cual se entrenarán los clasificadores puesto que se utiliza aprendizaje supervisado. Buscando en la literatura no se encontró un corpus previo en el idioma español ya etiquetado por lo que será necesario crear un conjunto de varios mensajes de distintos autores en diferentes situaciones que potencialmente puedan ser acoso. Para lograr esto se hará uso de la red social *Twitter* debido a las siguientes características que presenta, que pueden ser de gran utilidad:

- Dominio: El contenido de los mensajes están relacionados a una amplia variedad de tópicos.

- Datos disponibles. Una gran cantidad de datos pueden ser fácilmente recolectados debido a que *Twitter* provee una *API*[8] para recolectar *tweets*.

Ahora bien, la *API* de *Twitter* puede regresar mensajes de tres formas distintas dependiendo de la forma en la que se acceda:

- El *Streaming API* proporciona un subconjunto de *tweets* en casi tiempo real. Se establece una conexión permanente por usuario con los servidores de *Twitter* y mediante una petición http se recibe un flujo continuo de *tweets* en formato *json*. Se puede obtener una muestra aleatoria (*statuses/sample*), un filtrado (*statuses/filter*) por palabras claves o por usuarios.
- La *Search API* suministra los *tweets* con una profundidad en el tiempo de 7 días que se ajustan a la consulta solicitada. Es posible filtrar por cliente utilizado, lenguaje y localización. No requiere autenticación y los *tweets* se obtienen en formato *json* o *atom*.
- La *REST API* ofrece a los desarrolladores el acceso al nucleo de los datos de *Twitter*. Todas las operaciones que se pueden hacer vía *web* se pueden realizar desde la *API*.

Para este trabajo se utilizó la *API de Streaming*. También se diseñaron los filtros que se mencionan más abajo para obtener información útil al problema planteado en esta tesis, además se hizo una combinación de estos dependiendo del *corpus* a generar.

1. Lista de palabras altisonantes: es una la lista de palabras que potencialmente pueden incluirse en mensajes de acoso. Esta lista fue elaborada según un estudio de la UNICEF[9] acerca del acoso en Chile, las palabras se adaptaron al contexto en México; estas palabras caen dentro de grupos como son: relacionado al aprendizaje (retrasado, tonto, etc.), nivel socioeconómico (pobre, indigente, etc.), relacionado al comportamiento sexual (facilota, homosexual, etc.).
2. Coordenadas geográficas: Los mensajes cuentan con información de geolocalización y sólo se tomaron en cuenta los que estaban dentro de la latitud y longitud de la república mexicana. Esto se hizo para evitar mensajes en otro idioma o con otras formas de hablar el español.
3. Mensajes que contengan referencias a usuarios, esto para que potencialmente se dirija el ataque a una persona y no a un ente o idea. Se puede manejar esta situación con detectar como parte del mensaje el símbolo @. Además, se verificó que solo hubiera referencia a sólo un usuario

[8] https://dev.twitter.com/streaming/overview
[9] http://www.unicef.cl/pdf/PPTLaVozDiscriminacion2011.pdf

ya que si hay más de uno se puede perder el sentido de a quién va dirigido el ataque.

4. Mensajes con *emojis* y emoticonos para analizar si de alguna manera estos símbolos ayudan al aprendizaje del clasificador y su correcta clasificación.

4.1.1.1 Conjuntos de datos recopilados

Con base en los filtros definidos en la sección pasada se crearon dos conjuntos de datos (*corpus*), el primero con un contenido de 301 mensajes y el segundo con un conjunto de 556 mensajes. A continuación se describen los filtros aplicados a cada uno así como su filosofía de diseño:

Corpus 1: Inicialmente se pensó en obtener la mayor cantidad de mensajes potenciales de acoso, por ende, para este corpus se aplicó el filtro de palabras altisonantes comunes en los mensajes de acoso, además como se requería que los mensajes solo pertenecieran a la república mexicana también se aplicó el filtro de coordenadas geográficas.

Corpus 2: Con la experiencia de creación del *corpus* anterior, se vio que se necesitaba delimitar aún más los mensajes, debido a la complejidad de abarcar varios tipos de acoso ya que para esto se necesita analizar a profundidad las características que detectan un tipo de acoso de otro, para esto se hizo uso de una taxonomía de acoso (véase Tabla 1.1) y se enfocó en el tipo “acoso”; primeramente se incluyó el filtro de obtener mensajes con *emojis* y emoticonos para hacer un análisis posterior del uso de estas entidades, también se usó el filtro que obtiene mensajes con referencias a usuarios. Además, para tener un *corpus* equilibrado en cuanto al número de elementos (*emojis*, emoticonos, palabras altisonantes, referencias a usuarios y mensajes “libres”, es decir cualquier mensaje que cachara el extractor) se definieron los siguientes porcentajes para cada combinación. Véase Tabla 4.1

Palabra obscena	Emoticono / *Emoji*	Referencia a usuario	Porcentaje	
✓	✓	✓	25%	+
✓		✓	30%	
✓	✓		5%	
	✓	✓	5%	
✓			20%	
	✓		5%	
		✓	5%	
			5%	-

Tabla 4.1 Cantidad de mensajes por tipo de filtro

4.1.2 Etapa de etiquetado manual de los mensajes.

Ya que se tienen los dos *corpus* recabados lo siguiente es etiquetarlos de manera manual, esto es, asignar a cada mensaje una clase.

Para el primer conjunto de 301 mensajes se definieron las siguientes clases:

- Acoso directo – Indica que se está humillando de manera directa y tajante a otra persona.
- Acoso moderado – Puede tener palabras altisonantes y de alguna manera se avergüenza a la otra persona.
- Acoso leve – Pude contener alguna palabra altisonante pero a veces son las formas de hablar de las personas.
- Nada de acoso – Simplemente no es acoso.

La elección de estas categorías fue porque ni aun la inteligencia humana que es en definitiva la que hace el etiquetado “perfecto” puede a veces diferenciar entre solo dos clases como: acoso y no acoso, por ende, se pensó en dar un margen de decisión a los jueces y que no se estuviera atado solo a dos. Las situaciones que se identificaron por las cuales a los jueces se les dificulto el etiquetado son:

1) Falta de contexto: es decir el mensaje viene dentro de una plática la cual aporta la información necesaria para saber exactamente el significado de dicho mensaje;
2) Idiosincrasia del evaluador: esto es a veces el evaluador considera que ciertas frases o formas de hablar son comunes y no amerita propiamente una forma de acoso;
3) Poca información: debido a que los mensajes están limitados a sólo 140 caracteres y en su mayoría tienen enlaces que explicitan más el mensaje.

Como se mencionó anteriormente, en el segundo conjunto de datos se acotó más la forma de clasificar los mensajes y se creó un tipo de taxonomía de acoso (Véase Tabla 1.1), la cual enfoca a un tipo de acoso más específico. Esto porque se notó que si se quería abarcar todos los tipos de acoso y que el clasificador tuviera un buen desempeño se hubiera necesitado integrar aún más características al clasificador.

La clasificación de los mensajes se realizó por tres jueces para disminuir la posibilidad de caer en subjetividad; por ende, se tenían 3 categorías por mensaje, una por cada juez.

Para elegir la clase final para cada mensaje en el *corpus* uno, primero se seleccionaron los mensajes que por lo menos dos jueces le dieron la misma categoría. Con esto quedaron 292 mensajes, después las clases de acoso directo y acoso moderado se unieron en una sola clase, esta es: “acoso” y las clases de acoso leve y nada de acoso se unieron en la clase: “no acoso”. Por tanto, al final se generó un *corpus* con 292 mensajes y dos clases.

De igual manera se procedió para elegir la clase final para un mensaje en el *corpus* dos.

4.1.2.1 Acuerdo de inter-anotación.

Dado que un problema inherente en la categorización es la variabilidad entre los observadores, es decir, de un anotador a otro puede variar cómo se clasifica un mensaje, se tiene que usar una medida de acuerdo. Para determinar el grado de acuerdo se hace uso de una herramienta estadística usada frecuentemente: el coeficiente de Kappa, es decir hasta qué punto coinciden las mediciones de los observadores. El índice Kappa es el más común ya que considera en su cálculo el papel que pudo tener el azar en la coincidencia de las observaciones.

La fórmula para obtener el coeficiente de Kappa es la siguiente. Ver ecuación 4.1

$$k = \frac{(Co - Ce)}{1 - Ce}$$

Ecuación 4.1 Fórmula para coeficiente de Kappa

Donde:

Co = Concordancia observada

Ce = Concordancia esperada por azar

Aunque siempre es una escala subjetiva, Landis y Koch (Landis & Koch, 1977) propusieron unos límites para el grado de acuerdo estimado con el resultado del cálculo de Kappa. La Tabla 4.2 muestra la posible interpretación.

Valor de Kappa	Fuerza de concordancia
< 0	Pobre
0 a 0.20	Leve
0.21 a 0.40	Mediana
0.41 a 0.60	Moderada
0.61 a 0.80	Sustancial
0.81 a 1.00	Casi perfecta

Tabla 4.2 Valores de referencia de Kappa

El enfoque que se utiliza en esta tesis, cuenta con tres anotadores, por lo que dicha fórmula no serviría ya que sólo funciona para dos, por lo que se usara la fórmula extendida de Fleiss (Fleiss, 1981) para más de dos anotadores:

Fleiss generalizó la aplicación del índice Kappa de Cohen para medir el acuerdo entre más de dos codificadores u observadores para datos de escala nominal y ordinal. Ver ecuación 4.2.

$$k = \frac{\bar{P} - \bar{P_e}}{1 - \bar{P_e}}$$

Ecuación 4.2 Formula para Kappa de Fleiss

Donde:

$\bar{P}$ Representa la suma de la proporción de pares de los jueces que están de acuerdo en su evaluación sobre el tema *i.*

$\bar{P_e}$ Representa la suma de los errores.

4.1.3 Etapa de pre procesamiento de los datos y selección de características

La siguiente etapa es la de pre procesamiento de los datos. En esta etapa se hizo una limpieza de los mensajes debido a que hay cierta información que debe ser cambiada para mejorar la precisión del clasificador. La información eliminada fueron los signos de puntuación, ya que éstos comúnmente no aportan información útil, también se sustituyeron los enlaces de la forma “https://t.co/iqYkYT53xt“con la palabra “enlace” simplemente para indicar que el mensaje contiene un *link* a cierto recurso. El objeto de hacer esto es porque dentro de los alcances de esta tesis se definió clasificar el mensaje sólo con base en su contenido, sin tomar en cuenta otros recursos que estén por fuera de dicho mensaje aunque estén referenciados dentro de él.

Ahora que ya se han limpiado los mensajes quitando signos de puntuación y cambiado los enlaces, es necesario tener algún modelo para representar estos datos de manera adecuada para algún método de clasificación. Como primer acercamiento a dicha representación de los datos se usó el proceso conocido como “bolsa de palabras”.

Tal como se vio en la sección 3.1.1 el modelo bolsa de palabras es una representación simplificada usada en procesamiento del lenguaje natural y en recuperación de información. En este modelo, un texto (por ejemplo, una frase o un documento) se representa como la bolsa (conjunto múltiple) de sus palabras, sin tener en cuenta la gramática e incluso el orden de palabras, pero manteniendo la multiplicidad.

El modelo de bolsa de palabras se usa comúnmente en métodos de clasificación de documentos y como en este trabajo de tesis se pretende llevar a cabo la tarea de clasificación de mensajes ya sea en acoso o no acoso ésta es la línea base.

Posteriormente se hace uso de los n-gramas clásicos y finalmente se prueba con los n-gramas sintácticos. Primeramente, para obtener el árbol sintáctico como el que se muestra en la Figura 4.3, se tiene que utilizar un *parser* (software encargado de encontrar las relaciones sintácticas de una frase), para ello, se utiliza el *parser Freeling*[10] del cual posteriormente se extraen los n-gramas. A manera de ejemplo se observa cómo dicho software crea el árbol sintáctico. Para el siguiente mensaje: *"dulcee88 es de naco ser aficionado del América",* véase Figura 4.1.

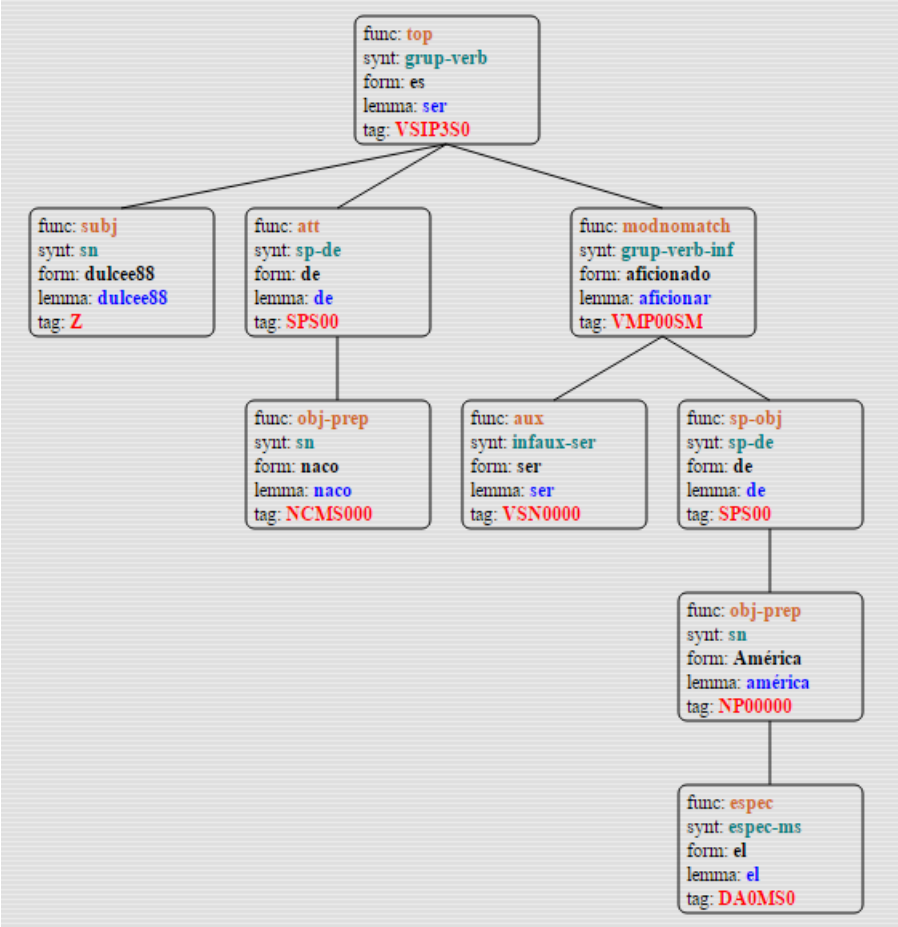

Figura 4.1 Árbol sintáctico dado un mensaje

Para obtener los n-gramas correspondientes, pueden utilizarse los programas desarrollados en el lenguaje Python por Grigori Sidorov disponibles públicamente en su página Web[11]. En la Tabla 4.4 se muestran algunos ejemplos de *bigramas* y *trigramas* de acuerdo al árbol sintáctico antes mostrado.

[10] http://nlp.lsi.upc.edu/freeling/
[11] http://www.cic.ipn.mx/~sidorov/

Bigramas	Trigramas
es dulcee88, es de, de naco, es aficionado, etc…	es de naco, es aficionado ser, es aficionado de, aficionado de América, etc..

Tabla 4.4 Ejemplo de n-gramas sintácticos obtenidos a partir del árbol mostrado en la figura 4.3

Mediante estos elementos (n-gramas) se forma el vector de características que es alimentado como entrada a los algoritmos de aprendizaje automático. Es importante destacar que se pueden formar varios tipos de vectores dependiendo de qué información se tome de los n-gramas sintácticos.

4.1.4 Etapa de entrenamiento de los clasificadores

En esta etapa se crean programas en *Python* por lo que se utilizara así mismo una biblioteca que ya tenga implementados los algoritmos de aprendizaje automático, esto con el fin de no reinventar la rueda y hacer más rápido el proceso de codificación. También se van a crear para cada *corpus* dos subconjuntos, uno que va a servir para entrenar al clasificador y otro para hacer las respectivas pruebas.

4.1.5 Etapa de combinación de resultados

Las actividades que se cubren en esta etapa se basan en el uso de una técnica de combinación de resultados, la cual de manera básica indica que después de usar varios clasificadores con una muestra supervisada, la clase final dada a un mensaje es aquella con mayor frecuencia, es decir aquella con mayor número de aparición en las respuestas individuales de cada clasificador.

La técnica de combinación se tiene que programar, el lenguaje empleado es *Python* debido a su amplia documentación en línea y por su rápida puesta en marcha de las aplicaciones.

4.1.6 Etapa de resultados

En esta etapa se utiliza la validación cruzada de *k* pliegues, primeramente, se separarán los datos en dos conjuntos: uno de entrenamiento y otro de pruebas. Con el conjunto de entrenamiento se entrenan los clasificadores y con el de prueba se hacen las validaciones. Como salida final se tendrá la clasificación de un mensaje como acoso o no acoso.

4.1.6.1 Validación de los resultados

Existen algunas medidas utilizadas para cuantificar el desempeño obtenido en tareas de clasificación. Tales medidas son: exactitud (*accuracy*), precisión (*precision*), especificidad (*recall*) y medida F (*F-measure*) las cuales son descritas a

continuación. En el caso específico de esta investigación, la tarea de clasificación consiste en saber si un mensaje es de acoso y si es así se dice que es un caso positivo, de lo contrario es un caso negativo.

Las medidas de desempeño anteriormente mencionadas están definidas con base en los siguientes términos, que se obtienen al comparar el resultado del clasificador con el valor real:

- TP (*True Positive*): Representa el caso cuando el clasificador etiqueta un mensaje como positivo correctamente.
- TN (*True Negative*): Representa el caso cuando el clasificador etiqueta un mensaje como falso correctamente.
- FP (*False Positive*): Representa el caso cuando el clasificador etiqueta un mensaje como positivo erróneamente.
- FN (*False Negative*): Representa el caso cuando el clasificador etiqueta un mensaje como negativo erróneamente.

De lo anterior se ve claramente que pueden surgir dos tipos de errores:

- Error tipo 1: El clasificador dice que un mensaje es positivo pero no lo es (falso positivo), de aquí surge la medida de precisión que se define como:

  Precisión: $\frac{tp}{tp+fp}$

  Esta medida expresa el porcentaje de mensajes clasificados como positivos que realmente eran positivos, es decir mide qué tan acertada es la clasificación con respecto a una etiqueta específica. También se puede ver como la proporción de datos recuperados que es relevante en el contexto del sistema. Si su valor es 1, se interpreta que todos los datos recuperados son relevantes.

- Error tipo 2: El clasificador dice que un mensaje no es positivo pero en realidad sí lo es (falso negativo), de aquí surge la medida de especificidad que se define como:

  Especificidad: $\frac{tp}{tp+fn}$

  Esta medida indica la cobertura del clasificador, es decir indica qué porcentaje de los casos positivos fueron detectados al realizar la clasificación. También se puede ver como la proporción de datos relevantes que fueron recuperados. Si su valor es 1, entonces se recuperaron todos los datos relevantes.

Para combinar precisión y especificidad en una métrica que permita evaluar una clasificación, se acostumbra calcular la media armónica de ambas cantidades en la

medida F. Cuando pondera de igual manera precisión y especificidad, se le conoce como F1.

F1: $2 * \frac{precisión \cdot especificidad}{precisión + especificidad}$

Su valor oscila entre 0 y 1, si es cero indica que el sistema es totalmente inexacto y conforme va aumentando a 1 indica el porcentaje de efectividad.

Por último, la medida más clásica es la exactitud (*accuracy*). Mide el porcentaje de resultados correctos totales obtenidos, por lo que, si se clasifican correctamente 50 pares de 100, entonces su exactitud será de 50%.

La fórmula siguiente muestra la manera de calcular dicho valor:

Exactitud: $\frac{tp+tn}{tp+fp+fn+tn}$

4.2 Resumen del capítulo

En este capítulo se han definido las etapas de la metodología a seguir durante el desarrollo de la tesis, desde la adquisición de los datos, el análisis y procesamiento de los mismos, hasta la clasificación de los mensajes, mostrando cada una de las actividades a realizar durante cada etapa. Dentro de una de estas etapas, se explicaron los criterios para decidir los factores a utilizar para la predicción con las herramientas de aprendizaje automático.

En el siguiente capítulo se detallan las implementaciones realizadas en cada etapa dada la metodología elegida para esta investigación, así como el uso de distintos paquetes de *software*. También se mencionan a detalle los ajustes realizados al modelo de aprendizaje, y la explicación de los resultados preliminares obtenidos.

Capítulo 5

Implementación, pruebas y resultados

En este capítulo se describe la parte de desarrollo de la tesis. En primera instancia se describe la forma en cómo se desarrollaron los procesos, algoritmos, y actividades que sirvieron para probar y validar las diferentes etapas de la metodología propuesta. Posteriormente se va desarrollando el caso de estudio para entre otras cosas determinar o predecir si un mensaje nuevo es de acoso o no. Posteriormente se detalla cómo fueron utilizados los diferentes métodos de aprendizaje automático para la predicción y se discuten los argumentos para configurar los parámetros para dar un rendimiento adecuado.

5.1 Herramientas y algoritmos utilizados para cada etapa.

Para las necesidades de las distintas etapas de la metodología se han utilizado una serie de bibliotecas de código libre así como las propias desarrolladas que han permitido probar y validar la metodología propuesta. El lenguaje de programación utilizado fue *Python* en su versión 2.7. Para las distintas librerías utilizadas que más adelante se van a ir mencionando propiamente se usó el gestor de paquetes Anaconda[12], el cual contiene alrededor de 720 paquetes ya con las distintas herramientas y algoritmos listos para ser usados. Como entorno de desarrollo integrado se utilizó *Pycharm Community edition* versión 4.5.

5.1.1 Adquisición de datos

En el proceso de recolección de mensajes se hizo uso de la librería *Tweepy* la cual, va a ayudar a extraer los mensajes de *Twitter*. Con esta librería se puede usar el lenguaje de programación *Python* y además su curva de aprendizaje es relativamente rápida. Posteriormente los mensajes se almacenaron en un archivo de texto con codificación *UTF-8* para de esta manera conservar los *emojis*.

En esta etapa se desarrolló un programa el cual se conecta a la *API de Streaming* de Twitter. Para poder conectarse a esta API se debe contar con 4 piezas de información: *API key*, *API secret*, *Access token* y *Access token secret*. Éstas se obtienen siguiendo los siguientes pasos:

1. Crear una cuenta de *Twitter* si no se tiene una ya creada.
2. Ir a la página https://apps.twitter.com/ e ingresar con el usuario y contraseña previamente creado.

[12] https://www.continuum.io/anaconda-overview

3. Seleccionar “crear nueva aplicación”.
4. Llenar el formulario, aceptar la licencia y dar clic en ”crea tu aplicación *Twitter*”.
5. En la siguiente página dar clic en la pestaña *API Keys* y copiar la *API Key* y *API Secret*.
6. Ir abajo en la misma página y dar clic en *Create my access token* y copiar el *Access token* y el *Access token secret*.

Ya con esta información se empezó a recolectar la información teniendo en cuenta los filtros comentados en la sección 4.1.1 y se generaron los siguientes conjuntos de datos.

- Conjunto 1: Se recolectaron 301 mensajes en un periodo de una semana, los filtros que se aplicaron aplicados fueron: geolocalización y palabras altisonantes.
- Conjunto 2: 556 mensajes, los filtros que se aplicaron aplicados fueron: geolocalización, palabras altisonantes, referencias a usuario, detección de *emojis* y emoticonos.

La Tabla 5.1 muestra algunos ejemplos de los mensajes recolectados mediante el extractor para los dos conjuntos:

Conjunto 1	Conjunto 2
Ni con todo el maquillaje del mundo arreglaras tu fealdad. Baja de peso porfas🐽 oink	@Luisifhm jajaaja con razón cabrón, puto desquicio desde las 8:30, no mamen. 😭😢😩😩
Me pondría celosa, pero estas bien culera 🤳😂	@FerCastGam te amo mamitaa💕💕
Chamoyadas con mi negro💁👄🤙 orlandoarribeno ❤❤ @ Plaza Caracol, Puerto… https://t.co/DJSai9dHJi	Me tocó aprender a valorar cada segundo contigo y a odiar cada día sin ti. @LuisHuertaPomp
Que gusto me da ver gente "gordita" echándole todas las ganas en el gimnasio. Bien por ellos.	Creo te puede interesar @Densho https://t.co/pVtSDfhKDe

Tabla 5.1 Ejemplos de mensajes de acoso obtenidos por medio del extractor

5.1.2 Etapa de etiquetado manual de los mensajes.

Con los dos conjuntos de datos, se les proporcionó a los jueces en un formato de texto plano para que procedieran a la categorización. Para el primer corpus se les indico que podían clasificar los mensajes en 4 clases inicialmente y para el segundo en solo dos. Además, para el primer *corpus* no se les comentó ninguna regla adicional para llevar a cabo la clasificación, sólo se les solicito que tuvieran en cuenta que aunque en algunos mensajes aparecieran palabras altisonantes pero sin intención de molestar a la otra persona, no se tomara como acoso y que se basaran únicamente en lo que decía el mensaje.

Para el segundo *corpus* se les dieron las siguientes reglas adicionales:

1. Para que sea un mensaje de acoso primeramente se tiene que tener una persona-usuario que está atacando a otra persona-usuario.
   - No es acoso si se ataca a un ente (idea, lugar, cosa, situación). Ejemplos:
     - ✓ He sobrevivido como a 4 #FinDelMundo pura mamada
     - ✓ De algo han servido las putas clases de dimensión...”
     - ✓ No hay cosa que odie mas a que no me contesten rápido.
   - No es acoso si se ataca a sí mismo. Ejemplos:
     - ✓ El pinche osos de mi vida
     - ✓ Esto de comprar tonterías es lo mio! 💪 (@ Office Depot in Puebla, PUE) https://t.co/iqYkYT53xt
     - ✓ Ya me cago la madre traer el cabello recogido todos los días.😭
   - No es acoso aunque haya palabras altisonantes pero que no tienen el objeto de insultar a la persona. Ejemplos:
     - ✓ Siendo bien vergas pero sin ejercer.
     - ✓ Por que puta te dije que ya no😭
     - ✓ Mi vecina ni me habla pero es buena para madrearme 😀
     - ✓ jajajaja te extraño niñita pedorrita💗

     Hay que tener mucho cuidado en el último ejemplo, ya que, aunque hay una palabra altisonante claramente dirigida a alguien, por el resto del mensaje se sabe que no es acoso, es decir se usan los verbos extrañar, el diminutivo niñita y el emoji 💗.

   - No es acoso cuando se trata de menciones demeritorias a otra persona, ya que tienen que ser agresiones directas. Ejemplos:
     - ✓ me dio asco, que pantalones tan mugrosos llevaba
     - ✓ como ves su estúpida idea del no circula
     - ✓ con ese profe no me meto, sus pinches clases están bien culeys

2. Limitarse solo al contenido del mensaje, es decir no hacer uso del sentido común o conocimiento propio extra. Tampoco tomar en cuenta imágenes o archivos adjuntos.

3. Realizar la tarea con calma, no etiquetar todos los mensajes de una sola vez para evitar el fastidio y caer en etiquetar de manera mecánica.

Al ver las evaluaciones de los jueces se notó que había varias discrepancias, por las situaciones ya comentadas acerca de falta de contexto, idiosincrasia del juez, etc. por lo que se decidió ver cuál era el grado de acuerdo de inter-anotación.

Ya con los datos obtenidos de los dos *corpus* anotados se ingresa a la página de Jeroen Geertzen[13] la cual contiene un *software* en línea para calcular la fórmula.

Primero se analizan los resultados del primer *corpus*. Con este conjunto de datos se obtiene el resultado que se puede apreciar en la Figura 5.1, donde: “n+” es igual a la clase donde no hay nada de acoso, “n-“ representa a la clase de acoso leve es decir hay un indicio de acoso o palabras altisonantes pero no es completamente acoso, “s+” representa la clase de acoso directo y “s-“ se asigna a la clase de acoso moderado ya que en algunos casos no es muy claro si en verdad es acoso, como por ejemplo se tiene el caso del auto-acoso. En la Tabla 5.2 se presentan algunos ejemplos de cómo los jueces etiquetaron los mensajes.

Mensaje	Clasificación
BigBrother_PM Que asco es un cerdo, mega naco... fuera de la casa y que lo limpie con la lengua	Acoso directo (s+)
Voy a acabar negra, NEGRA dije con este horrible sol	Acoso moderado (s-)
Su risa es tan asdfghjkl :') pinchemente varonil😈	Acoso leve (n-)
Mi gordo es lo mejor 😍	Nada de acoso (n+)

Tabla 5.2 Ejemplos de clasificación de mensajes

Se puede apreciar que en la segunda fila, puede tratarse de un caso de auto acoso mismo que se explicó que no debía ser anotado como acoso en el *corpus* 2.

[13] https://nlp-ml.io/jg/software/ira/

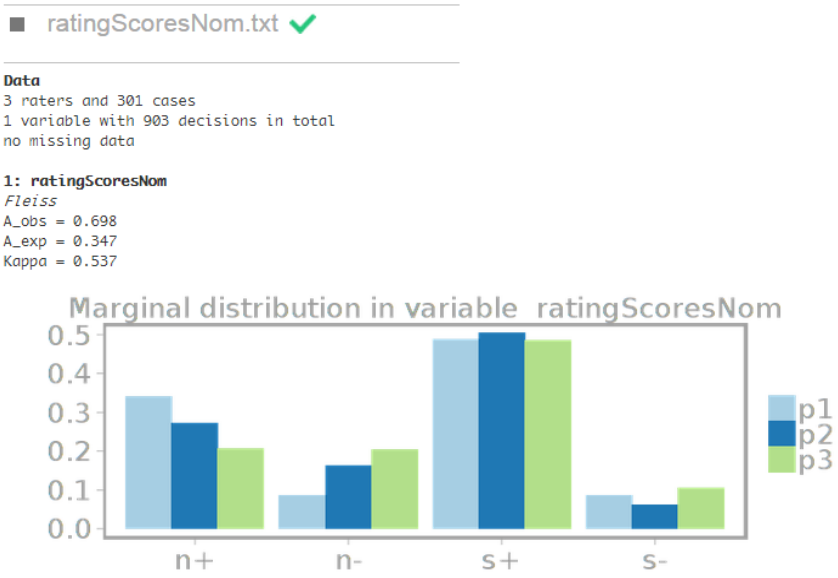

Figura 5.1 Coeficiente de Fleiss Kappa para 3 anotadores

En este primer experimento se tomaron en cuenta los 301 mensajes así como los 3 evaluadores. El valor de Kappa alcanzó un valor de k=0.537, el cual, tomando en cuenta la escala de Landis y Koch correspondería a una concordancia moderada.

Para mejorar el valor del acuerdo se hicieron ciertos cambios, primero se quitaron los registros que son totalmente diferentes en la anotación. Ahora, volviendo a introducir estos datos en la página de Jeroen se obtiene lo siguiente, Figura 5.2:

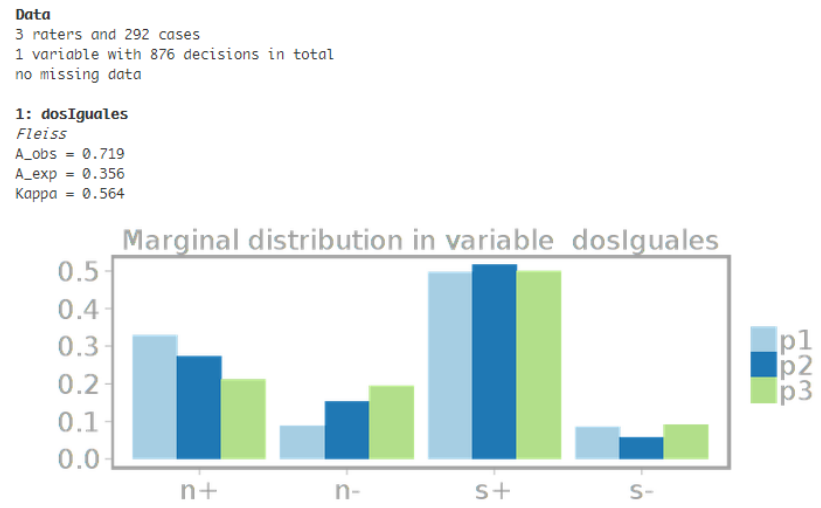

Figura 5.2 Coeficiente Kappa de Fleiss sin registros diferentes

Se puede apreciar que el valor Kappa de Fleiss aumentó ya que ahora es igual a 0.564 aunque de manera poco significativa dado que sigue manteniéndose en el nivel moderado, de hecho sólo se quitaron 9 registros en los que los 3 anotadores estaban totalmente en desacuerdo. En la Tabla 5.3 se muestran algunos de estos casos:

Mensaje	Clasificación
@humano_foxy DDDD: Foxy....parece que no eres TAN tontito.... DDD:	P1: nada acoso, P2: acoso moderado, P3: acoso leve
Voy a acabar negra, NEGRA dije con este horrible sol.	P1: nada de acoso, P2: acoso leve, P3: acoso moderado
jajajaja te extraño niñita pedorrita💓	P1: nada acoso, P2: acoso directo, P3:acoso leve

Tabla 5.3 Ejemplos de mensajes en los cuales los jueces están en total desacuerdo

Ya que aún es bajo el valor *Kappa* de *Fleiss* dada una baja concordancia entre varios anotadores debido a la idiosincrasia de cada uno, se propone probar con distintas combinaciones de jueces. Por ejemplo, se toman combinaciones con sólo dos anotadores esperando tener mejores resultados. Los resultados de tomar al juez uno (p1) y dos (p2) se muestran en la Figura 5.3:

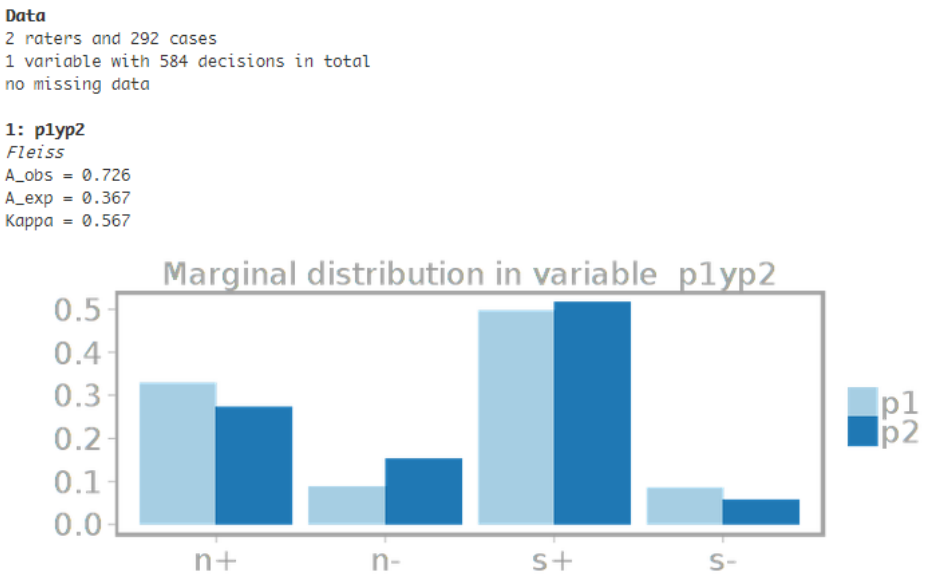

Figura 5.3 Anotadores persona 1 y persona 2

Se puede apreciar que sube un poco el valor a 0.567 pero aún continúa en un nivel moderado. Ahora se hacen las pruebas con los jueces uno y tres (p3), véase Figura 5.4:

Data
2 raters and 292 cases
1 variable with 584 decisions in total
no missing data

1: p1yp3
Fleiss
A_obs = 0.664
A_exp = 0.35
Kappa = 0.484

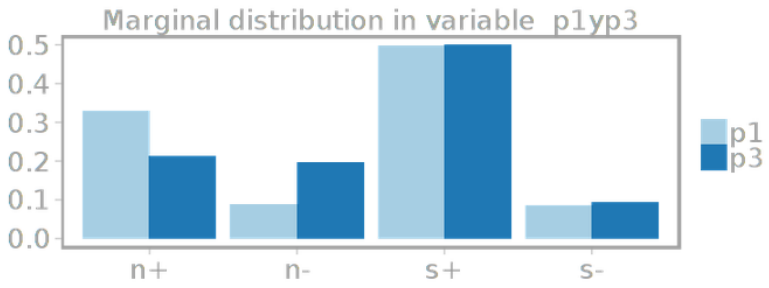

Figura 5.4 Anotadores persona 1 y persona 3

Este es el peor caso (0.484) y da pie a formular la siguiente hipótesis: La discrepancia en las anotaciones tiene que ver con el contexto sociocultural de cada juez; queda para trabajo a futuro validarla. Realizando un análisis somero podría decirse que el juez 1 es más liberal y el juez 3, más conservador. Es por esto que difieren más en la categorización de los mensajes. Finalmente véanse los resultados con el juez dos y tres, en la Figura 5.5.

Data
2 raters and 292 cases
1 variable with 584 decisions in total
no missing data

1: p2yp3
Fleiss
A_obs = 0.767
A_exp = 0.353
Kappa = 0.64

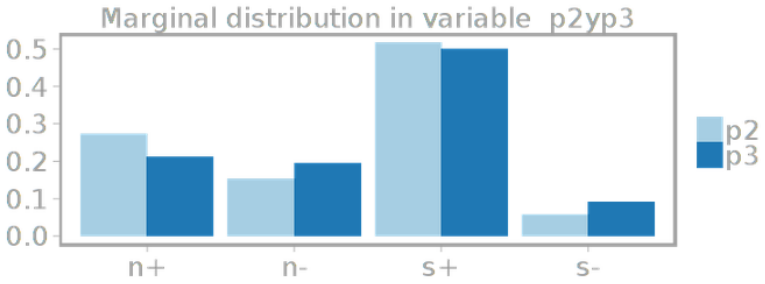

Figura 5.5 Anotadores persona 2 y persona 3

En esta última figura se puede apreciar que los jueces dos y tres tienen un contexto cultural similar y por ende aumenta el valor de Kappa a 0.64 lo cual nos lleva al siguiente nivel de la escala que es: “sustancial”. Debido a que éste es el valor más alto que se ha obtenido se decide utilizar esta combinación como el *corpus* inicial para trabajar.

Ya definido el *corpus* 1 para trabajar, se crearon programas en Python para que cada mensaje se quedara con solo una clase.

Para el segundo *corpus* se procedió de manera igual que en el primero. Inicialmente se quitaron los registros en los que los jueces estaban totalmente en desacuerdo, esto es posible aunque solo sean dos clases debido a que un juez pudo hacer olvidado poner la etiqueta para ese mensaje y por consecuente las otras dos restantes eran distintas.

Al final de este proceso se obtuvo un *corpus* de 542 mensajes. Posteriormente se introdujo las clases de este *corpus* de cada anotador para cada mensaje en la página antes citada para calcular el valor de Kappa de Fleiss, con lo que se obtuvo un valor de .029, que de acuerdo a la escala de Landis y Koch es un valor “leve”. Esto se puede apreciar en la Figura 5.6. Finalmente debido a la poca experticia de los jueces y al poco tiempo que se les dio para etiquetar este *corpus* se optó por mejor solo manejar la clasificación de un juez, debido a que este fue el más calificado, debido a que participo desde un inicio en la creación de este *corpus*.

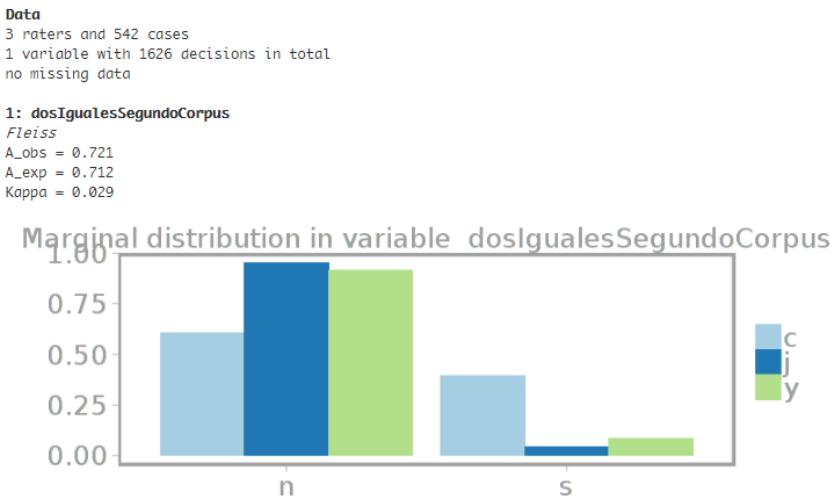

Figura 5.6 Valor Kappa de Fleiss para el segundo corpus.

5.1.3 Etapa de pre procesamiento de los datos y selección de características

Para realizar las tareas siguientes se programaron varios *scripts* en *Python*. Ahora, para que el clasificador pueda procesar los mensajes y llevar a cabo su tarea de aprendizaje a partir del conjunto de datos, es necesario representar esta información en un formato adecuado de acuerdo al *software* o paquetería que se está utilizando. En este caso específico se utilizó la biblioteca *scikit-learn*[14], la cual pide como entrada a los distintos clasificadores que implementa, un conjunto de vectores de entrenamiento, estos vectores deben tener valores numéricos en sus componentes. Al formar los vectores se debe seleccionar qué características son de importancia, es decir cuántos y qué componentes va a tener el vector. Esta es una tarea que aún

[14] http://scikit-learn.org/stable/

se está tratando en la literatura dada su importancia y dificultad. Inicialmente se va a ocupar lo que se conoce como bolsa de palabras que básicamente toma como características de los vectores todas las palabras que se encuentran en los mensajes. Por tanto, como primer paso para formar los vectores se tomaron todos los mensajes y se creó un *lexicón* o diccionario que contiene a todas las palabras que se encontraron en los mensajes sin repetir. Después, para cada mensaje se creó su correspondiente vector de tamaño igual al número de palabras en el diccionario y como valor de cada componente se tomaron en cuenta dos medias: Para la primera se puso un valor binario, esto es, si aparecía la palabra en el mensaje se pone 1, de lo contrario, 0. La otra medida es TF·IDF la cual es una medida clásica que se utiliza en el ámbito de recuperación la información. Básicamente es una medida que pretende reflejar la importancia de una palabra a un documento en una colección.

Ahora ya que se tiene la forma de representar los mensajes en vectores también es de interés procesar algunos caracteres o entidades, ya que dichos entes no aportan mucha información útil al clasificador y por ende es recomendable sustituirlos o eliminarlos, la razón principal es que en la fase de entrenamiento y pruebas del clasificador se tarda más porque hay más componentes (características) en cada vector y además debido a su aparición inconstante, introducen ruido.

Cabe mencionar que hay que hacer una distinción entre los símbolos *emojis* y los emoticonos, los cuales ambos se procesan y se tratan como entidades distintas, la diferencia básica es que un símbolo *emoji* es una imagen y el emoticono es la unión de letras y signos de puntuación, ambos representan un sentimiento o emoción, como por ejemplo alegría, enfado, etc.

Las actividades de limpieza realizadas fueron:

- Eliminar las URL ya que estás referencian a otro contenido extra complementario al mensaje, como por ejemplo imágenes, videos o sitios donde viene más a fondo una noticia.
- Cambiar todas las referencias a usuario(@juanMtz) por USERNAME.
- Eliminar signos de puntuación ya que salvo ciertos casos y en ciertas tareas específicas no aportan información útil al clasificador. Ver lista completa en el apéndice 1.
- Quitar palabras de no contenido (*stop words*) porque al igual no aportan información útil, es evidente que estas palabras son comunes a la gran mayoría de mensajes. Ver lista completa en apéndice 2.
- Procesar los *emojis* de acuerdo a una lista predefinida para evitar problemas con la codificación y además se hace uso de los estándares ya que dicha lista está definida por el consorcio de Unicode. Básicamente el procesamiento que se hace es mapear un símbolo *emoji* a una cadena de caracteres, que es la representación nombrada de ese símbolo. Para esto se hizo uso de la biblioteca *emoji* 0.3.9 programada en el lenguaje *Python*. Ver lista completa en http://www.webpagefx.com/tools/emoji-cheat-sheet/

- Procesar los emoticonos también de acuerdo a una lista definida, esta lista se creó manualmente basándose en varios trabajos e investigaciones. Ver lista completa en apéndice 3.
- Agrupar distintas flexiones de una palabra en su raíz, lo que se conoce como *stemming*. Aquí se utilizó la librería *NLTK* y el paquete *Snowball*.

5.1.4 Etapa de entrenamiento de los clasificadores

En esta etapa se utilizó la biblioteca *scikit-learn* ya que es una simple y eficiente herramienta para minería de datos y análisis de datos, contiene los tres algoritmos de aprendizaje automático que van a utilizarse, *Naïve Bayes*, Máquinas de soporte vectorial y regresión logística. Además, es de uso gratuito y está programado en el lenguaje de programación *Python*.

5.1.5 Etapa de resultados

Esta serie de experimentos está dividida en dos etapas la primera con el primer conjunto de datos 292 y la segunda con el conjunto de 500 mensajes.

5.1.5.1 Análisis con el *corpus* de 292 mensajes

En esta primera etapa de los experimentos se va a trabajar con el *corpus* de 292 mensajes. Se van a hacer experimentos haciendo ciertos trabajos de pre-procesamiento como quitar palabras de no contenido (*stop words*), dejar la raíz de una palabra *stemming*, etc., además por último se aborda una línea especial de análisis la cual consiste en ver que tanto influyen los símbolos *emojis* en la clasificación.

Además, para tener una idea clara de cómo esta balanceado el corpus se incluye en la Figura 5.7, el total de mensajes para cada clase. Se puede apreciar que el *corpus* está balanceado con un 58% de clases positivas y el restante 42% de clases negativas.

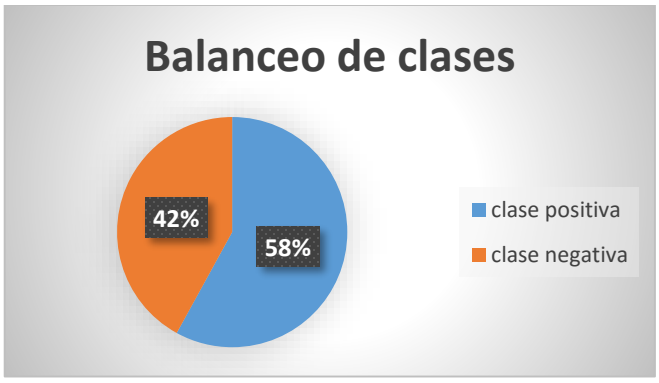

Figura 5.7 Numero de mensajes para cada clase.

Experimento 1. En este experimento se quiere ver cuál es la línea base, es decir, los valores que sin hacer ningún experimento complejo marcan el inicio de partida, para posteriormente al hacer los experimentos más sofisticados aumentar estos valores. Para esto se propone la forma de clasificación más "ingenua" la cual es donde el clasificador siempre predice la clase positiva (mensaje de acoso). En la Tabla 5.4 se muestran los resultados para los tres distintos clasificadores que se utilizan a lo largo de la experimentación.

N-gramas			#mensajes #características	Representación		Métrica			
1	2	3		TF·IDF	Binaria	Exactitud	Precisión	Especificidad	F1
✓	✓	✓	292, 1520	✓	✓	58%	58%	100%	73%

Tabla 5.4 Valores de la línea base.

Resultados: Todas las tablas de resultados tienen el mismo número de columnas; la primera (n-gramas), representa cómo se formaron los vectores con sus componentes (características), es decir, si se tomó usa sola palabra (unigrama), dos palabras (bigrama), etc. La segunda muestra el número de mensajes que se utilizaron, así como el número de características que resultaron para de cada vector. La tercera (representación), columna apunta a el valor que se le dio a cada componente de los vectores. Finalmente, la última columna(métrica) indica las medidas de evaluación explicadas en la sección 4.1.6.1. En este primer experimento es claro ver porque el valor para especificidad es del 100%, como siempre se predice a positivo no hay ningún falso negativo y de acuerdo a la fórmula para especificidad vista en la sección 4.1.6.1 se cubren todos los casos de acoso. De igual manera es fácil ver porque tanto en exactitud como en precisión el valor es 58%, en ambos casos aunque las formulas son diferentes el 58% corresponde con el 58% de las clases positivas del total del *corpus*. Se espera que en los próximos experimentos se superen estos valores al crear modelos más sofisticados.

Experimento 2. Como inicio de los experimentos se desea ver cuál es la calidad de los resultados; también se desea ver qué tanto influye la representación de los mensajes en una representación común (bolsa de palabras), por ende, se procede con una clasificación de *tweets* de manera "básica" es decir se empezará por hacer un pre procesamiento sencillo, ya que sólo se quitaron las URL, también se cambiaron las referencias a usuario (@username) por "username". De igual manera se eliminaron los signos de puntuación. Se hizo uso de un clasificador común en la tarea de análisis de sentimientos (*Naïve Bayes*). Como entradas al clasificador, los vectores de palabras son formados a partir de un diccionario generado a partir del *corpus* creado. Como valor de cada componente de los vectores se utilizaron dos medidas diferentes por separado: en este caso específico la medida de valor binario que indica si la palabra está presente o ausente en el mensaje y la medida TF·IDF.

Además, para estabilizar los valores, ya que de una ejecución del programa a otra varían los resultados, se ejecutó 100 veces el programa y se promediaron los valores obtenidos. También se hizo uso de validación cruzada a 5 pliegues. En la

Tabla 5.5 se presentan los resultados tanto para vectores formados de una sola palabra, como para dos y tres palabras tomadas en conjunto (bigramas y trigramas).

N-gramas			#mensajes	Representación		Métrica			
1	2	3	#características	TF·IDF	Binaria	Exactitud	Precisión	Especificidad	F1
✓			292, 1520		✓	65%	66%	81%	72%
✓				✓		57%	66%	52%	58%
	✓		292, 3391		✓	55%	69%	41%	51%
	✓			✓		49%	72%	21%	32%
		✓	292, 3564		✓	46%	73%	12%	20%
		✓		✓		41%	84%	4%	7%

Tabla 5.5 Resultados con un Clasificador Bayesiano con pre procesamiento clásico.

Resultados: Al observar los resultados y analizando la primera fila se percibe que tomando en cuenta el pre procesamiento clásico y usando un clasificador bayesiano, se obtienen resultados acordes a la literatura, bastando únicamente con tomar unigramas a pesar de que son pocos mensajes aunados al estilo de escritura vago. Además, con esta primera prueba se ha superado la línea base. También es interesante destacar que todas las demás filas de la tabla no importando el modelo de características o la representación de estas los resultados salen peor que en la línea base en las métricas: exactitud especificidad y medida f1, pero siempre se supera a la línea base en precisión.

Experimento 3: Ahora se realiza más pre procesamiento de la información con el fin de ver si es posible mejorar los resultados, en la literatura se recomienda hacer lematización, *stemming*, eliminación de *stop words*, quitar signos de puntuación, etc. Por tanto, para este experimento se eliminaron las palabras de no contenido, esto es palabras que se usan más como conectores y que no aportan mucha información útil para el clasificador, además también se hizo uso del *stemming* para reducir el número de características y de alguna manera agrupar todas las distintas variantes de una misma palabra, a esto se le llama procesamiento robusto. Los resultados se muestran en la Tabla 5.6.

N-gramas			#Mensajes	Representación		Métrica			
1	2	3	#Características	TF·IDF	Binaria	Exactitud	Precisión	Especificidad	F1
✓			292, 1089		✓	69%	70%	80%	75%
✓				✓		61%	69%	60%	64%
	✓		292, 1823		✓	49%	65%	29%	39%
	✓			✓		45%	75%	9%	15%

N-gramas			#Mensajes	Representación		Métrica			
1	2	3	#Características	TF·IDF	Binaria	Exactitud	Precisión	Especificidad	F1
		✓	284, 1587		✓	46%	59%	25%	34%
		✓		✓		42%	83%	4%	7%

Tabla 5.6 Resultados con un Clasificador Bayesiano con pre procesamiento robusto.

Resultados: Se puede apreciar que los resultados comparados con los del experimento 1 son mejores pero sólo cuando se toman unigramas. En la parte de bigramas sale peor y en trigramas los resultados son muy parecidos. Con esto se reafirma que los bigramas consideran el contexto local de cada palabra y por ende, al eliminar las *stop words* se pierde algo (aunque sea poco) de información sintáctica que aportan los n-gramas.

Experimento 4: A modo de comprobación de la hipótesis pasada en la que al hacer la eliminación de *stop words* afectaba a los bigramas, se realizará este experimento pero sólo se realizará *stemming*. Los resultados se muestran en la Tabla 5.7.

N-gramas			#Mensajes	Representación		Métrica			
1	2	3	#Características	TF·IDF	Binaria	Exactitud	Precisión	Especificidad	F1
✓			292, 1089		✓	66%	67%	83%	74%
✓				✓		60%	67%	62%	64%
	✓		292, 1823		✓	54%	68%	42%	51%
	✓			✓		50%	72%	24%	36%
		✓	284, 1587		✓	46%	73%	12%	20%
		✓		✓		41%	79%	4%	8%

Tabla 5.7 Resultados con un Clasificador Bayesiano con pre procesamiento clásico y stemming.

Resultados: Los resultados para los unigramas y trigramas son parecidos, lo que se comprobó fue que salieron mejor los resultados para los bigramas al no hacer eliminación de *stop words*.

Experimento 5: Ya que se han visto los resultados previos tanto al pre procesar de manera básica como de una manera más “robusta”, ahora se desea conocer que tanto las clases son separables, es decir se va a utilizar todo el conjunto de datos para entrenar el clasificador y con este mismo conjunto de datos se va a hacer las pruebas. De entrada se piensa que debe dar un valor muy cercano al cien por ciento, ya que el clasificador va a ver todos los datos y por ende puede clasificarlos correctamente a todos. Los resultados aparecen en la Tabla 5.8.

N-gramas			#Mensajes	Representación		Métrica			
1	2	3	#Características	TF·IDF	Binaria	Exactitud	Precisión	Especificidad	F1
✓			292, 1089		✓	97%	96%	99%	97%
✓				✓		100%	100%	100%	100%
	✓		292, 1823		✓	100%	100%	100%	100%
	✓			✓		100%	100%	100%	100%
		✓	284, 1587		✓	100%	100%	100%	100%
		✓		✓		100%	100%	100%	100%

Tabla 5.8 Resultados de un clasificador bayesiano entrenado con el 100 % de datos y evaluado con el mismo 100%.

Resultados: Con base en lo que se obtuvo queda claro que sí es posible hacer una separación completa de las clases, y por consecuente queda como tarea encontrar la mejor representación de los mensajes así como encontrar qué características son las idóneas para que el clasificador pueda aprender bien, además de ver cuál es la mejor elección de la cuantificación de los componentes de cada vector.

Experimento 6: Es este experimento se tomó la misma configuración que en el experimento 2 pero ahora se hizo uso de otro clasificador. En específico se usó el clasificador de máquinas de soporte vectorial ya que en la literatura se menciona como una opción conveniente en las tareas de clasificación de documentos. Los resultados se muestran en la Tabla 5.9.

N-gramas			#Mensajes	Representación		Métrica			
1	2	3	#Características	TF·IDF	Binaria	Exactitud	Precisión	Especificidad	F1
✓			292, 1089		✓	66%	68%	76%	71%
✓				✓		60%	65%	79%	71%
	✓		292, 1823		✓	54%	57%	97%	72%
	✓			✓		50%	58%	97%	72%
		✓	284, 1587		✓	46%	57%	99%	72%
		✓		✓		41%	57%	99%	72%

Tabla 5.9 Resultados con un Clasificador SVM con procesamiento robusto.

Resultados: Como se esperaba se observa que los resultados no varían mucho, aunque cabe destacar que en la métrica de especificidad aumentó bastante tanto para bigramas como para trigramas.

Experimento 7: Por último, en este experimento se continúa tomando la misma configuración que el experimento anterior, pero ahora con un clasificador de

regresión logística, se espera que se obtengan resultados muy parecidos, los cuales pueden verse en la Tabla 5.10.

N-gramas			#Mensajes	Representación		Métrica			
1	2	3	#Características	TF·IDF	Binaria	Exactitud	Precisión	Especificidad	F1
✓			292, 1089		✓	67%	68%	82%	74%
✓				✓		65%	66%	83%	73%
	✓		292, 1823		✓	57%	57%	99%	72%
	✓			✓		58%	58%	96%	72%
		✓	284, 1587		✓	57%	57%	99%	73%
		✓		✓		57%	57%	99%	72%

Tabla 5.10 Resultados con un Clasificador de regresión logística con pre procesamiento robusto

Resultados: Al igual se puede ver que los resultados no varían mucho, pero al igual que paso con el clasificador de SVM aquí también aumento la métrica de especificidad.

Experimento 8: Ya que ha visto que la variación no es mucha de acuerdo a si se escoge uno u otro clasificador, en esta serie de tres experimentos siguientes se van a incluir más características con el fin de ver si mejoran los resultados. Para esto, se prueban distintas combinaciones de unigramas, bigramas y trigramas. En este primer experimento se utilizó un clasificador bayesiano. Véase Tabla 5.11.

N-gramas			#Mensajes	Representación		Métrica			
1	2	3	#Características	TF·IDF	Binaria	Exactitud	Precisión	Especificidad	F1
✓	✓	✓	292, 4499		✓	63%	74%	55%	63%
✓	✓	✓		✓		50%	71%	25%	37%
✓	✓		292, 2912		✓	67%	73%	68%	70%
✓	✓			✓		56%	72%	39%	50%
✓		✓	284, 2676		✓	67%	73%	69%	71%
✓		✓		✓		55%	71%	38%	49%
	✓	✓	292, 3410		✓	46%	70%	16%	26%
	✓	✓		✓		43%	82%	5%	10%

Tabla 5.11 Resultados con un clasificador Bayesiano con combinaciones de n-gramas

Resultados: Se puede apreciar que los mejores resultados son las combinaciones de unigramas mas bigramas y unigramas mas trigramas con representación binaria. También ya se puede ir viendo dado los experimentos anteriores que la medida

TF·IDF en este caso específico de modelado de la información (bolsa de palabras) en detección de acoso no ayuda mucho.

Experimento 9: En esta prueba se desea observar si la variación es pequeña teniendo presente la misma configuración, pero ahora con otro clasificador. En este caso específico el clasificador usado es una máquina de soporte vectorial. Se espera que se obtengan resultados parecidos. Véase Tabla 5.12.

N-gramas			#Mensajes	Representación		Métrica			
1	2	3	#Características	TF·IDF	Binaria	Exactitud	Precisión	Especificidad	F1
✓	✓	✓	292, 4499		✓	65%	66%	82%	73%
✓	✓	✓		✓		63%	63%	90%	74%
✓	✓		292, 2912		✓	66%	67%	80%	73%
✓	✓			✓		64%	64%	87%	73%
✓		✓	284, 2676		✓	64%	66%	78%	72%
✓		✓		✓		63%	64%	86%	73%
	✓	✓	292, 3410		✓	57%	58%	98%	73%
	✓	✓		✓		58%	58%	98%	73%

Tabla 5.12 Resultados con un clasificador SVM con combinaciones de n-gramas

Resultados: Se puede comprobar que efectivamente la variación es pequeña.

Experimento 10: Por último, es necesario cerciorarse que de igual manera la variación debe ser pequeña teniendo la presente la misma configuración, pero ahora con un clasificador de regresión logística. Los resultados se muestran en la Tabla 5.13.

N-gramas			#Mensajes	Representación		Métrica			
1	2	3	#Características	TF·IDF	Binaria	Exactitud	Precisión	Especificidad	F1
✓	✓	✓	292, 4499		✓	66%	65%	88%	75%
✓	✓	✓		✓		64%	64%	89%	74%
✓	✓		292, 2912		✓	67%	66%	87%	75%
✓	✓			✓		65%	65%	87%	74%
✓		✓	284, 2676		✓	67%	67%	86%	75%
✓		✓		✓		65%	64%	88%	74%
	✓	✓	292, 3410		✓	57%	58%	99%	73%

N-gramas			#Mensajes	Representación		Métrica			
1	2	3	#Características	TF·IDF	Binaria	Exactitud	Precisión	Especificidad	F1
	✓	✓		✓		58%	58%	98%	73%

Tabla 5.13 Resultados con un clasificador de regresión logística con combinaciones de n-gramas

Resultados: Al igual se puede ver que los resultados no varían mucho, pero al igual que paso con el clasificador de SVM aquí también la métrica de especificidad aumentó.

Experimento 11: Dado estos tres últimos experimentos es posible percibir que el clasificador de regresión logística tuvo los mejores resultados, por ende, se van a analizar con este clasificador qué resultados se obtienen si no se eliminan los *stop words* y solo se hace *stemming*, esto porque al eliminar las palabras auxiliares se puede perder información de utilidad a los bigramas y trigramas. Véase Tabla 5.14.

N-gramas			#Mensajes	Representación		Métrica			
1	2	3	#Características	TF·IDF	Binaria	Exactitud	Precisión	Especificidad	F1
✓	✓	✓	292, 4499		✓	61%	62%	85%	71%
✓	✓	✓		✓		62%	62%	89%	73%
✓	✓		292, 2912		✓	61%	63%	81%	71%
✓	✓			✓		62%	63%	85%	72%
✓		✓	284, 2676		✓	61%	63%	82%	71%
✓		✓		✓		63%	63%	89%	73%
	✓	✓	292, 3410		✓	58%	58%	98%	73%
	✓	✓		✓		57%	59%	87%	70%

Tabla 5.14 Resultados con un clasificador de regresión logística (Solo stemming)

Resultados: Inicialmente se pensó que al incluir los *stop words* para ayudar a los bigramas y trigramas se tendría un mejor desempeño en el clasificador, pero se ha comprobado que no. Ya que los resultados salen peor que cuando se eliminaron.

Experimento 12: Para finalizar esta etapa de experimentos se observa qué pasa si cuando se hacen combinaciones de n-gramas no se quitan ni *stop words* ni se hace *stemming*. El clasificador a utilizar es el mismo de Regresión logística. Véase Tabla 5.15.

N-gramas			#Mensajes	Representación		Métrica			
1	2	3	#Características	TF·IDF	Binaria	Exactitud	Precisión	Especificidad	F1
✓	✓	✓	292, 8475		✓	61%	62%	85%	71%

N-gramas			#Mensajes	Representación		Métrica			
1	2	3	#Características	TF·IDF	Binaria	Exactitud	Precisión	Especificidad	F1
✓	✓	✓		✓		58%	60%	86%	70%
✓	✓		292, 4911		✓	60%	61%	89%	72%
✓	✓			✓		58%	61%	82%	69%
✓		✓	292, 5084		✓	61%	62%	85%	71%
✓		✓		✓		58%	60%	83%	69%
	✓	✓	292, 6955		✓	61%	62%	87%	72%
	✓	✓		✓		58%	58%	98%	73%

Tabla 5.15 Resultados con un clasificador de regresión logística con pre procesamiento clásico.

Resultados: Los valores salen más bajos que en la tabla pasada, por lo que se comprueba que al no incluir las dos características de pre procesamiento (eliminar stop words y hacer stemming) perjudican los resultados.

Dado la vasta cantidad de experimentos se muestra en la Tabla 5.16 un resumen donde se exponen los mejores resultados de cada modelo.

Modelo	Exactitud	F1
Unigrama, Bayes	69%	75%
Unigrama, SVM	66%	71%
Unigrama, RL	67%	74%
Unigrama + bigrama, Bayes	67%	70%
Unigrama + bigrama, SVM	66%	73%
Unigrama + bigrama o Unigrama + Trigrama, RL	67%	75%

Tabla 5.16 Resumen de resultados

Resultados: Se puede apreciar que el mejor modelo es el de unigramas además de que el mejor clasificador es el de *Naïve Bayes*.

Línea de *Emojis* y emoticonos

A partir de este tipo de experimentos ejecutados en el mismo *corpus* ahora se va a seguir una línea especial de análisis partiendo de solo tomar los mensajes que

contengan *emojis* o emoticonos, esto porque se quiere analizar si estos símbolos pueden ayudar al clasificador a tener un mejor desempeño.

Experimento 13: Al igual que en los otros experimentos se representó los mensajes en un modelo de espacio vectorial también como valor de las características se hizo uso tanto de medida binaria como TF·IDF. También se tomó validación cruzada de 5 pliegues y se ejecutaron 100 iteraciones. Dado que en el experimento dos se observaron los resultados tomando en cuenta tanto las palabras junto con los *emojis* y emoticonos, ahora solo se tomará en cuenta las puras palabras. Los resultados se muestran en la Tabla 5.17, considerando los siguientes n-gramas y usando el clasificador de Regresión logística.

N-gramas			#Mensajes	Representación		Métrica			
1	2	3	#Características	TF·IDF	Binaria	Exactitud	Precisión	Especificidad	F1
✓			292, 1039		✓	67%	68%	82%	74%
✓				✓		66%	67%	83%	74%
	✓		292, 1714		✓	57%	58%	99%	73%
	✓			✓		57%	58%	97%	72%
		✓	277, 1467		✓	58%	58%	100%	73%
		✓		✓		58%	58%	100%	73%

Tabla 5.17 Resultados con un clasificador de regresión logística tomando solo palabras

Resultados: Dado los resultados obtenidos es posible decir, que si influye el hecho de tener símbolos de *emojis* y emoticonos en un mensaje, ya que en esta vez salieron mejor los resultados con solo procesar el texto.

Experimento 14: Dado que en el experimento pasado salieron mejor los resultados al no incluir los *emojis* y *emoticones*, queda la interrogante si al incluir estos símbolos afecta el desempeño del clasificador, porque también cabe mencionar que se pudo haber dado el caso que en dos mensajes distintos con clases distintas se usara el mismo símbolo, en dado caso el clasificador no sabría cómo clasificar un mensaje que incluya ese mismo símbolo en ambos mensajes como característica. En el siguiente experimento lo que se hizo fue solo utilizar los mensajes que contengan al menos un símbolo *emoji* o emoticono y ver qué resultados entrega. Ver Tabla 5.18.

N-gramas			#Mensajes	Representación		Métrica			
1	2	3	#Características	TF·IDF	Binaria	Exactitud	Precisión	Especificidad	F1
✓			80,49		✓	55%	58%	42%	46%
✓				✓		55%	56%	46%	48%
	✓		35, 37		✓	64%	84%	40%	49%

N-gramas			#Mensajes	Representación		Métrica			
1	2	3	#Características	TF·IDF	Binaria	Exactitud	Precisión	Especificidad	F1
	✓			✓		60%	81%	39%	47%
		✓	13, 15		✓	60%	73%	87%	73%
		✓		✓		62%	76%	85%	74%

Tabla 5.18 Resultados con un clasificador de regresión logística tomando solo símbolos.

Resultados: Estos resultados indican que si es factible hacer uso de los *emojis* y emoticonos para tenerlos presentes como características de los vectores de entrenamiento de un clasificador. La justificación a esto es que usando únicamente símbolos se tiene un valor más alto que la línea base.

Experimento 15: En el siguiente experimento vamos a ver la línea tope que se puede alcanzar si solo se usan los símbolos *emojis*. Por ende se va a entrenar un clasificador de regresión logística con todos los mensajes y con estos mismos se va a probar. Véase Tabla 5.19.

N-gramas			#Mensajes	Representación		Métrica			
1	2	3	#Características	TF·IDF	Binaria	Exactitud	Precisión	Especificidad	F1
✓			80,49		✓	81%	84%	73%	78%
✓				✓		81%	87%	71%	78%
	✓		35, 37		✓	91%	100%	80%	88%
	✓			✓		91%	87%	93%	90%
		✓	13, 15		✓	92%	87%	100%	93%
		✓		✓		92%	87%	100%	93%

Tabla 5.19 Resultados de un clasificador de regresión logística entrenado con el 100% de datos.

Resultados: Se puede apreciar que la línea tope esta alrededor del 90% usando bigramas y como valor de las características TF·IDF.

Experimento 16: Para finalizar esta etapa de experimentos con símbolos *emojis* y emoticonos vamos a ver qué pasa cuando hacemos combinaciones de n-gramas. El clasificador a utilizar es el de Regresión logística. Véase Tabla 5.20.

N-gramas			#Mensajes	Representación		Métrica			
1	2	3	#Características	TF·IDF	Binaria	Exactitud	Precisión	Especificidad	F1
✓	✓		80, 101		✓	55%	59%	44%	46%
✓	✓			✓		54%	56%	48%	48%

N-gramas			#Mensajes	Representación		Métrica			
1	2	3	#Características	TF·IDF	Binaria	Exactitud	Precisión	Especificidad	F1
✓	✓		80, 86		✓	55%	58%	43%	46%
✓	✓			✓		55%	56%	48%	48%
✓		✓	80, 624		✓	56%	59%	42%	46%
✓		✓		✓		57%	59%	50%	51%
	✓	✓	35, 52	✓		62%	85%	38%	48%
	✓	✓		✓		64%	84%	40%	50%

Tabla 5.20 Resultados de un clasificador de regresión logística con combinaciones de n-gramas.

Resultados: Los mejores valores se generan cuando el modelo es de trigramas, esto refuerza la idea de que entre más símbolos *emojis* juntos más intensifica la polaridad del mensaje.

5.1.5.2 Análisis de casos

En este apartado se analizó la combinación de clasificadores para ver si aumentan los valores de las métricas, dada la Tabla 5.21 se definieron las siguientes reglas:

- Se va a analizar cada clasificador por separado y como se ayuda por medio de los otros dos.
- Es decir cada columna representa a un clasificador, las palomas indican que se hizo una clasificación correcta y los taches indican que no se hizo la clasificación correcta.
- La última columna indica que si algún clasificador contiene la clasificación correcta se podría tomar como acierto la fila al formar la combinación (aunque cabe aclarar que esto es trabajo a futuro ya que no sabemos de antemano cuál de los tres acertó).

Naïve Bayes	SVM	RL	1 de 3
✓	X	X	✓
X	✓	X	✓
X	✓	X	✓
X	X	✓	✓
X	X	X	X
✓	✓	✓	✓

✓	✓	X	✓

Tabla 5.21 Análisis de casos

Como parte de este análisis se obtuvieron los siguientes resultados mostrados en la Tabla 5.22 (la paloma más resaltada indica que ese fue el clasificador primario). Además de manera general se obtuvieron los siguientes valores:

- ✓ Ningún clasificador acertó: 20%
- ✓ Uno de los tres acertó: 79%
- ✓ Todos acertaron: 55%

N-gramas			Representación		Métrica				
1	2	3	TF·IDF	Binaria	Bayesiano	SVM	LR	Especificidad	F1
✓				✓	✓			69%	75%
✓				✓	✓	✓	✓	68%	75%
✓				✓		✓		66%	71%
✓				✓	✓	✓	✓	68%	75%
✓				✓			✓	67%	74%
✓				✓	✓	✓	✓	68%	75%

Tabla 5.22 Combinación de resultados

5.1.5.3 Análisis con el *corpus* de 500 mensajes

En esta segunda etapa de los experimentos se trabajó con el segundo *corpus* recolectado de 556 mensajes. Cabe recordar que este corpus se creó a partir de la taxonomía de acoso propuesta y por ende solo se enfoca en un tipo de acoso específico: acoso directo, por lo que si hay otro tipo de acoso en los mensajes de dicho corpus se va a manejar como no acoso, además en este corpus se encuentra balanceado con el 40% de clases positivas y el restante negativas, de la misma manera que en la fase pasada se va a empezar por hacer un pre-procesamiento básico y posteriormente se van a realizar más tareas de limpieza. También se consideró la línea especial de análisis de *emojis* y emoticonos.

Experimento 17: En este primer experimento solo se procesó de una manera básica los mensajes, ya que solo se eliminó las URL, además se cambió las referencias a usuario(@username) por “username” y se eliminaron los signos de puntuación, El primer clasificador a utilizar como lo hicimos en la fase pasada va a ser un clasificador bayesiano, así mismo las entradas al clasificador son vectores de palabras formados a partir del diccionario generado del segundo *corpus* de esta

investigación. Como valor de cada componente de los vectores se van a utilizar también las dos medidas pasadas, en este caso específico tanto la medida de valor binario que indica si la palabra está presente o ausente en el mensaje y la medida TF·IDF.

Igualmente para estabilizar los valores se ejecutaron 100 veces el programa y se promediaron los valores obtenidos, también se va a hacer uso de validación cruzada a 5 pliegues. A continuación se presentan los resultados tanto para vectores formados de una sola palabra, como de dos y tres palabras tomadas en conjunto (bigramas y trigramas). Véase Tabla 5.23.

N-gramas			#Mensajes	Representación		Métrica			
1	2	3	#Características	TF·IDF	Binaria	Exactitud	Precisión	Especificidad	F1
✓			556, 2173		✓	78%	74%	70%	71%
✓				✓		76%	75%	63%	68%
	✓		556, 5298		✓	72%	80%	43%	56%
	✓			✓		73%	71%	56%	62%
		✓	554, 5768		✓	64%	75%	17%	27%
		✓		✓		63%	68%	26%	35%

Tabla 5.23 Resultados con un Clasificador Bayesiano con pre procesamiento clásico.

Resultados: Al observar los resultados vemos que si aumentaron los valores de todas las métricas con respecto al *corpus* pasado.

Experimento 18: Ahora vamos a hacer más pre-procesamiento de la información con el fin de mejorar los resultados, se eliminaron las palabras de no contenido esto es palabras que se usan más como conectores y que no aportan información útil para el clasificador, además también se hizo uso del *stemming* para reducir el número de características y de alguna manera agrupar todas las distintas variantes de una misma palabra. Véase Tabla 5.24.

N-gramas			#Mensajes	Representación		Métrica			
1	2	3	#Características	TF·IDF	Binaria	Exactitud	Precisión	Especificidad	F1
✓			556, 1607		✓	74%	69%	67%	68%
✓				✓		69%	62%	63%	62%
	✓		556, 3142		✓	69%	80%	30%	44%
	✓			✓		71%	74%	43%	54%
		✓	541, 2882		✓	60%	60%	6 %	10%
		✓		✓		61%	71%	6%	11%

Tabla 5.24 Resultados con un Clasificador Bayesiano con pre procesamiento robusto.

Resultados: Pese a todos los pronósticos los resultados salieron peor al hacer más pre procesamiento.

Experimento 19: Dado los resultados pasados ahora se va a hacer *stemming* y ver qué resultados se obtienen. Véase Tabla 5.25.

N-gramas			#Mensajes	Representación		Métrica			
1	2	3	#Características	TF·IDF	Binaria	Exactitud	Precisión	Especificidad	F1
✓			556, 1729		✓	71%	70%	70%	71%
✓				✓		69%	64%	66%	69%
	✓		554, 5107		✓	73%	45%	55%	73%
	✓			✓		66%	58%	61%	66%
		✓	554, 5740		✓	72%	18%	28%	72%
		✓		✓		64%	27%	37%	64%

Tabla 5.25 Resultados con un Clasificador Bayesiano

Resultados: Al parecer si beneficia hacer *stemming* ya que los resultados tienen un menor valor.

Experimento 20: También se quiere comprobar que resultados se obtienen si solo se eliminan las *stop words*. Los resultados aparecen en la Tabla 5.26.

N-gramas			#Mensajes	Representación		Métrica			
1	2	3	#Características	TF·IDF	Binaria	Exactitud	Precisión	Especificidad	F1
✓			556, 1990		✓	76%	75%	64%	68%
✓				✓		74%	70%	64%	66%
	✓		556, 3184		✓	68%	84%	25%	39%
	✓			✓		70%	78%	38%	50%
		✓	541, 2886		✓	60%	54%	4%	8%
		✓		✓		60%	67%	4%	8%

Tabla 5.26 Resultados con un Clasificador Bayesiano con eliminación de stop words

Resultados: Dados los resultados presentes y comparados con los pasados se puede apreciar que estos son los peores resultados cuando se hace más pre-procesamiento.

Experimento 21: Por último, vamos a analizar las combinaciones de n-gramas para ver su comportamiento. Véase, Tabla 5.27.

N-gramas			#Mensajes	Representación		Métrica			
1	2	3	#Características	TF·IDF	Binaria	Exactitud	Precisión	Especificidad	F1
✓	✓	✓	556, 13239		✓	78%	74%	68%	71%
✓	✓	✓		✓		76%	80%	54%	64%
✓	✓		556, 7471		✓	79%	76%	70%	72%
✓	✓			✓		77%	79%	58%	67%
✓		✓	556, 7941		✓	77%	73%	68%	70%
✓		✓		✓		75%	79%	54%	64%
	✓	✓	556, 11066	✓		71%	80%	39%	52%
	✓	✓		✓		72%	70%	55%	61%

Tabla 5.27 Resultados con un Clasificador Bayesiano con combinaciones de n-gramas.

Resultados: En estos últimos resultados se resalta el hecho de que tomar combinaciones de unigramas más bigramas aporta mejores resultados.

Línea de *Emojis* + emoticonos

Experimento 22: En este experimento se va a analizar solo puro texto. Los resultados se muestran en la Tabla 5.28.

N-gramas			#Mensajes	Representación		Métrica			
1	2	3	#Características	TF·IDF	Binaria	Exactitud	Precisión	Especificidad	F1
✓			556, 2036		✓	77%	74%	69%	71%
✓				✓		77%	76%	63%	68%
	✓		554, 4893		✓	73%	81%	45%	57%
	✓			✓		73%	71%	59%	64%
		✓	549, 5285		✓	65%	89%	14%	24%
		✓		✓		63%	71%	28%	34%

Tabla 5.28 Resultados con un Clasificador Bayesiano con solo texto.

Resultados: En estos resultados se resalta el hecho de que tomar combinaciones aporta mejores resultados.

Experimento 23: En este experimento se va a analizar únicamente con *emojis*. Véase Tabla 5.29.

N-gramas			#Mensajes	Representación		Métrica			
1	2	3	#Características	TF·IDF	Binaria	Exactitud	Precisión	Especificidad	F1
✓			247, 123		✓	57%	42%	23%	27%
✓				✓		53%	37%	32%	33%
	✓		133, 156		✓	68%	74%	14%	21%
	✓			✓		65%	63%	19%	27%
		✓	79, 105		✓	57%	28%	38%	31%
		✓		✓		60%	53%	31%	32%

Tabla 5.29 Resultados con un Clasificador Bayesiano con solo emojis

Resultados: Nuevamente los mejores valores se generan cuando se aplica un modelo con más elementos como lo son los trigramas.

5.2 Resumen del capítulo

En este capítulo se vio la implementación de las distintas etapas del sistema revisando a detalle los distintos algoritmos de aprendizaje automático, incluyendo las paqueterías disponibles y su configuración, así como el manejo de los distintos parámetros. También se revisó la mejor representación de los datos dentro del campo de los n-gramas, asi como del análisis de signos diferentes al alfabeto para una idónea clasificación de los mensajes. Dadas estas implementaciones se obtuvieron bastantes resultados dentro de los cuales se esbozaron unas ligeras conclusiones que se profundizan en el próximo capítulo.

Capítulo 6

Conclusiones y trabajos futuros

En este capítulo se presentan las conclusiones que hemos obtenido al término de esta tesis. Primero, se mencionan las limitaciones que se presentaron durante el desarrollo de este trabajo. Luego se abordan los logros alcanzados, con base en los resultados obtenidos al final del mismo. Después se comentan las aportaciones que consideramos que este trabajo presenta, desde varios enfoques. Finalmente, se proponen varias actividades como trabajo a futuro para mejorar el aquí presentado.

6.1 Conclusiones

Cuando se realiza pre procesamiento clásico y usando un clasificador bayesiano, se obtienen resultados acordes a la literatura, bastando únicamente con tomar unigramas a pesar de que son pocos mensajes aunados al estilo de escritura vago, además, se ha superado la línea base.

Inicialmente se pensó que al incluir los *stop words* para ayudar a los bigramas y trigramas se tendría un mejor desempeño en el clasificador, pero se ha comprobado que no. Ya que los resultados salen peor que cuando se eliminan.

Basados en la experimentación se comprobó que al no incluir las dos características de pre procesamiento (eliminar stop words y hacer stemming) los resultados salen con un menor valor.

Dados los distintos modelos de representación de las palabras se pudo apreciar que el mejor es el de unigramas además que el mejor clasificador es el de *Naïve Bayes*.

Cuando se analizaron los símbolos *emojis* y emoticonos y con base en los resultados es posible decir, que si influye el hecho de tener estos símbolos en un mensaje, ya que al incluirlos o no mejoraran o empeoraran los resultados. Por tanto dichos resultados nos indican que si es preferible hacer uso de los *emojis* y emoticonos y agregarlos como características de los vectores de entrenamiento de un clasificador. La justificación a esto es que usando únicamente símbolos se tiene un valor más alto que la línea base. También se puede apreciar que la línea tope para estos símbolos esta alrededor del 90% usando bigramas y como valor de las características TF·IDF. Finalmente en este análisis particular de *emojis* y emoticonos se tiene que los mejores valores se generan cuando el modelo es de trigramas, esto refuerza la idea de que entre más símbolos *emojis* juntos más intensifica la polaridad del mensaje.

Para el segundo corpus se pudo apreciar que sigue una misma tendencia con respecto del primer corpus en cuanto a mejora de los resultados al hacer uso de

lematización, *stemming* y aplicando los mismos modelos, así como el mismo clasificador.

Finalmente, cuando se analizó la combinación de clasificadores, se obtuvo que el clasificador bayesiano presentó un mejor comportamiento cuando se utilizó de manera individual sin combinarlo con los otros dos clasificadores utilizados. Además, la combinación de clasificadores ayuda cuando se elige un método de ponderación sobre el mejor clasificador.

6.2 Limitaciones

Los mensajes capturados mediante el extractor corresponden a un lenguaje común y coloquial, por lo que estos mensajes contienen bastantes faltas de ortografía, términos acuñados al vuelo y gramática errónea. Esto hace que sea difícil modelar el lenguaje y tener características comunes en distintos mensajes que sirvan de ayuda al clasificador.

También se tuvieron corpus anotados pequeños comparados con otros dados en competiciones y esto influyó en el aprendizaje de los clasificadores.

En otros mensajes hizo falta más contexto debido a que el extractor podría devolver un mensaje de un conjunto que pertenecía a una plática y por ende esta falta de información complicaba más el hecho de poder clasificar un mensaje.

6.3 Logros alcanzados

Al final del desarrollo de esta tesis se pudo ver que es posible detectar acoso automáticamente con pocos mensajes y con técnicas no tan robustas.

A pesar de algunas limitaciones con base en los resultados obtenidos consideramos que se cumplió con los objetivos de estudiar distintas técnicas de procesamiento del lenguaje natural, así como aplicar técnicas de aprendizaje automático para cumplir la tarea de detectar acoso en los mensajes de la red social Twitter.

6.4 Aportaciones

A continuación se enlistan las contribuciones generadas a partir del trabajo realizado:

- Se definió una metodología para la detección de acoso en mensajes de texto utilizando una combinación de diferentes técnicas de aprendizaje automático.
- Se definieron reglas para etiquetar un tipo específico de acoso.
- Se crearon dos *corpus* etiquetados para detectar acoso. El primero considera de manera general un mensaje de acoso. El segundo corpus fue creado más cuidadosamente, consta de 556 mensajes, solo se cuenta con dos clases (acoso y no acoso) y la detección de acoso se hace de manera más específica en base a la taxonomía de acoso definida.

- Se definió un método efectivo para clasificar mensajes (69% exactitud).
- Se implementó un sistema de detección de acoso que demuestra la validez de la metodología propuesta.

6.5 Trabajos futuros

Dadas las limitaciones vistas en la sección 6.1 se pueden generar varios trabajos cubriendo estas necesidades como, por ejemplo:

- Incrementar el tamaño de los *corpus*.
- Usar no solo las palabras del mensaje sino también hacer uso de información contextual inherente al mensaje, como lo son las imágenes, otros mensajes relacionados, entre otras.
- Analizar y procesar los mensajes que presentan una mala redacción así como aquellos escritos en una jerga local.
- Analizar y procesar la ambigüedad existe en los mensajes, así como detectar los sarcasmos y las burlas que pueden estar presentes en los mismos.
- Analizar los mensajes utilizando otras características lingüísticas.
- Utilizar técnicas de aprendizaje profundo.
- Utilizar técnicas de representación de datos como Doc2Vec.
- Obtener el grado de acuerdo para el segundo *corpus* así como hacer el análisis de casos para las combinaciones de clasificadores y del uso de los n-gramas sintácticos.

Apéndice 1. Signos de puntuación no utilizados en los mensajes.

Lista de signos de puntuación que fueron removidos de los mensajes de *Twitter*.

Signos			
.	,	;	¿
?	“	!	¡
-	:	/	=
(	)	‘	<
>	[	]	|
@	\	+	{
}	*	&	$
%	#	^	_
º			

Apéndice 2. Palabras de no contenido (*stop words*).

Lista de palabras de no contenido en el idioma español usadas en este trabajo.

Palabras			
a	acá	ahí	ajena
ajenas	ajeno	ajenos	al
algo	algún	alguna	algunas
alguno	algunos	allá	alli
allí	ambos	ampleamos	ante
antes	aquel	aquella	aquellas
aquello	aquellos	aqui	aquí
arriba	asi	atras	aun
aunque	bajo	bastante	bien
cabe	cada	casi	cierta
ciertas	cierto	ciertos	como
cómo	con	conmigo	conseguimos
conseguir	consigo	consigue	consiguen
consigues	contigo	contra	cual
cuales	cualquier	cualquiera	cualquieras
cuan	cuán	cuando	cuanta
cuánta	cuantas	cuántas	cuanto
cuánto	cuantos	cuántos	de
dejar	del	demás	demas
demasiada	demasiadas	demasiado	demasiados
dentro	desde	donde	dos
el	él	ella	ellas
ello	ellos	empleais	emplean
emplear	empleas	empleo	en

encima	entonces	entre	era
eramos	eran	eras	eres
es	esa	esas	ese
eso	esos	esta	estaba
estado	estais	estamos	estan
estar	estas	este	esto
estos	estoy	etc	fin
fue	fueron	fui	fuimos
gueno	ha	hace	haceis
hacemos	hacen	hacer	haces
hacia	hago	hasta	incluso
intenta	intentais	intentamos	intentan
intentar	intentas	intento	ir
jamás	junto	juntos	la
largo	las	Lo	los
mas	más	me	menos
mi	mía	mia	mias
mientras	mio	mío	mios
mis	misma	mismas	mismo
mismos	modo	mucha	muchas
muchísima	muchísimas	muchísimo	muchísimos
mucho	muchos	muy	nada
ni	ningun	ninguna	ningunas
ninguno	ningunos	no	nos
nosotras	nosotros	nuestra	nuestras
nuestro	nuestros	nunca	os
otra	otras	otro	otros
para	parecer	pero	poca

pocas	poco	pocos	podeis
podemos	poder	podria	podriais
podriamos	podrian	podrias	por
por qué	porque	primero	primero desde
puede	pueden	puedo	pues
que	qué	querer	quien
quién	quienes	quienesquiera	quienquiera
quiza	quizas	sabe	sabeis
sabemos	saben	saber	sabes
se	segun	ser	si
sí	siempre	siendo	sin
sín	sino	so	sobre
sois	solamente	solo	somos
soy	sr	sra	sres
sta	su	sus	suya
suyas	suyo	suyos	tal
tales	también	tambien	tampoco
tan	tanta	tantas	tanto
tantos	te	teneis	tenemos
tener	tengo	Ti	tiempo
tiene	tienen	toda	todas
todo	todos	tomar	trabaja
trabajais	trabajamos	trabajan	trabajar
trabajas	trabajo	tras	tú
tu	tus	tuya	tuyo
tuyos	ultimo	un	una
unas	uno	unos	usa
usais	usamos	usan	usar

usas	uso	usted	ustedes
va	vais	valor	vamos
van	varias	varios	vaya
verdad	verdadera	vosotras	vosotros
voy	vuestra	vuestras	vuestro
vuestros	y	ya	yo

Apéndice 3. Lista de emoticonos

Lista de emoticonos usada en este trabajo

Tipo	Emoticono
Felicidad	:) :D ;) :] =) =D :-D :-) :) ;-) -_- xD XD =] 8) :o) :3 :> :} =3 8D :'-) :') (y) ^_^ ;D xd :L ;P 8-| B-| B) 8=D
Enojo - tristeza	:(:'(>:(:[=(:/ :\ :-(: (>:-(:-/ :-\ >(:-c :c :-< :< :-[:[:{ :-|| :@ >:[:S >:\ >:/ :'-(
Otros	:P :p =P :-p :-P O:) 3:) :O :* 8-) B-) B| 3:-) O:-) :-* >:O ^_^ <3 8| :|] o.O O.o :v <:-| |;-) :b xp XP

Referencias

OCDE (2013). *Resultados de TALIS 2013: Estudio internacional sobre la enseñanza y el aprendizaje*. Recuperado el 14 de diciembre de 2016, de http://www.oecd.org/edu/school/talis-2013-results.htm

Committee for children. (2016). *Bullying prevention programs. A nonprofit working globally to prevent bullying*. Recuperado el 14 de diciembre de 2016, de http://www.cfchildren.org/programs/str/overview/

Valadez, B. (2014). *México es el primer lugar de bullying a escala internacional*. México. Recuperado el 14 de diciembre de 2016, de http://www.milenio.com/politica/Mexico-primer-bullying-escala-internacional_0_304169593.html

Castro, P., & Váldez, L. (2012). *MISAAC: Instant messaging tool for Ciberbullying Detection*. In WorldComp'12 – The 2011 World Congress in Computer Science, Computer Engineering, an Applied Computing.

Nalini, K., & Sheela, L. (2015). *Classification of Tweets Using Text Classifier to Detect Cyber Bullying*. Emerging ICT for Bridging the Future - Proceedings of the 49th Annual Convention of the Computer Society of India CSI. Volumen (2), pp 637-645.

Nahar, V., Unankard, S., Li, X., y Pang, C. (2012) *Sentiment Analysis for Effective Detection of Cyber Bullying.* Asia Pacific Web Conference. Volumen (7235), pp.767-774.

Sanchez, H. & Kumar. (2012). *Twitter bullying detection*, ser. NSDI'12. Berkeley, CA, USA: USENIX Association, pp. 15–22.

Galán-García, P., Gaviria, J., Laorden, C., Santos, I., y García, P. (2013). *Supervised Machine Learning for the Detection of Troll Profiles in Twitter Social Network: Application to a Real Case of Cyberbullying*. International Joint Conference SOCO'13-CISIS'13-ICEUTE'13, pp. 419 - 428.

Porter, M. (1980). *An algorithm for suffix stripping.* Program: electronic library and information systems. Volumen (14), pp. 130 - 137.

Manning, C. D., Raghavan, P., y Schütze, H. (2009). *Introduction to Information Retreival*. Cambridge, England.

Sidorov, G., Velasquez, F., Stamatatos, E., Gelbukh, A., y Chanona-Hernández, L. (2012). *Syntactic Dependency-Based N-grams as Classification Features*. En Lecture Notes in Artificial Intelligence. Volumen 7630, pp. 1–11. Springer Berlin Heidelberg

Sidorov, G., Velasquez, F., Stamatatos, E., Gelbukh, A., y Chanona-Hernández, L. (2013). *Syntactic Dependency-Based N-grams: More Evidence of Usefulness in Classification*. En Proceedings of the International Conference on Intelligent Text Processing and Computational Linguistics (CICLing 2013). Volumen (7816) de Lecture Notes in Computer Science, pp. 13–24. Springer Berlin Heidelberg.

Jurafsky, D. y Martin, J. H. (2009). *Speech and Language Processing: An Introduction to Natural Language Processing*, Computational Linguistics and Speech Recognition. Upper Saddle River, New Jersey, 2nd edición.

Mitchell, T. (1997) *Machine Learning*, McGraw Hill. 414 pages. ISBN 0070428077.

Lantz, B. (2015). *Machine learning with R.* Packt Publishing Ltd Second edition. 429 pages. ISBN 978-1-78439-390-8.

Sidorov, G. (2013). N-gramas sintácticos y su uso en la lingüística computacional. Vectores de investigación. Volumen (6), pp 13 -27.

Padró, L., Collado, M., Reese, S., Lloberes, M., & Castellón, I. (2010). *FreeLing 2.1 Five Years of Open-Source Language Processing Tools*. Proceedings of 7th Language Resources and Evaluation Conference (LREC 2010), ELRA La Valletta, Malta.

Fleiss, J. (1981). *Statistical methods for rates and proportions*. 2nd ed. New York: John Wiley, pp. 38–46.

Cortes, C., & Vapnik, V. (1995). *Support-Vector Networks*. Machine Learning. Volumen (20), pp. 273-297.

Landis J.R., & Koch G.G. (1977). *The measurement of observer agreement for categorical data*. Biometrics. Volumen (33), pp.159-174.

Glosario

Acoso. Toda forma de abuso físico, verbal o psicológico que ocurre entre estudiantes repetidamente a lo largo del tiempo.

Análisis morfológico. Es la rama de la lingüística que estudia la estructura interna de las palabras tales como sufijos, prefijos, raíces y flexiones, también estudia el sistema de categorías gramaticales de los idiomas, por ejemplo el género, número, etc. El problema principal de la morfología es que la implementación de sistemas de análisis y síntesis morfológica automática necesita del desarrollo de grandes diccionarios de raíces, lo cual es bastante difícil de obtener.

API. La interfaz de programación de aplicaciones, abreviada como API (del inglés: Application Programming Interface), es el conjunto de subrutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Son usadas generalmente en las bibliotecas de programación.

Aprendizaje automático. En ciencias de la computación el aprendizaje automático o aprendizaje de máquinas es una rama de la inteligencia artificial cuyo objetivo es desarrollar técnicas que permitan a las computadoras aprender. De forma más concreta, se trata de crear programas capaces de generalizar comportamientos a partir de una información no estructurada suministrada en forma de ejemplos. Es, por lo tanto, un proceso de inducción del conocimiento, es decir, un método que permite obtener por generalización un enunciado general a partir de enunciados que describen casos particulares.

Aprendizaje supervisado. Se genera una función que establece una correspondencia entre las entradas y las salidas deseadas del sistema, donde la base de conocimientos del sistema está formada por ejemplos etiquetados a priori (es decir, ejemplos de los que sabemos su clasificación correcta). Un ejemplo de este tipo de algoritmo es el problema de clasificación al que hemos hecho mención anteriormente.

Aprendizaje no supervisado. Donde el proceso de modelado se lleva a cabo sobre un conjunto de ejemplos formados únicamente por entradas al sistema, sin conocer su clasificación correcta. Por lo que se busca que el sistema sea capaz de reconocer patrones para poder etiquetar las nuevas entradas.

Bullies. Usuarios en las redes sociales que se dedican a acosar o intimidar a otros usuarios mediante mensajes ofensivos, revelando sus datos, suplantando su identidad, etc.

Bullying. Es un anglicismo que no forma parte del diccionario de la Real Academia Española (RAE), pero cuya utilización es cada vez más habitual en nuestro idioma. El concepto refiere al acoso escolar y a toda forma de maltrato físico, verbal o psicológico que se produce entre escolares, de forma reiterada y a lo largo del tiempo.

Chat. El chat (término proveniente del inglés que en español equivale a charla), también conocido como cibercharla, designa una comunicación escrita realizada de manera instantánea mediante el uso de un software y a través de Internet entre dos, tres o más personas ya sea de manera pública a través de los llamados chats públicos (mediante los cuales cualquier usuario puede tener acceso a la conversación) o privada, en los que se comunican dos o más personas.

Ciberacoso. El ciberacoso (derivado del término en inglés cyberbullying) también denominado acoso virtual o acoso cibernético, es el uso de medios de comunicación digitales para acosar a una persona o grupo de personas, mediante ataques personales, divulgación de información confidencial o falsa entre otros medios. Puede constituir un delito penal. El ciberacoso implica un daño recurrente y repetitivo infligido a través de los medios electrónicos. Según R. B. Standler,1 2 el acoso pretende causar angustia emocional, preocupación, y no tiene propósito legítimo para la elección de comunicaciones.

Emoji. Suya pronunciación más próxima a la fonética del español es emoyi, es un término japonés para los ideogramas o caracteres usados en mensajes electrónicos y sitios web. El término es una palabra compuesta que significa lo siguiente: imagen + letra Los emojis son utilizados como los emoticonos principalmente en conversaciones de texto a través de teléfonos inteligentes. Algunos de estos caracteres son muy específicos de la cultura japonesa, como imágenes de geishas, templos Dojo y grupos de comida como sushi y onigiri.

Emoticono. (Del acrónimo inglés emoticono) es una secuencia de caracteres ASCII que, en un principio, representaba una cara humana y expresaba una emoción. Posteriormente, fueron creándose otros emoticonos con significados muy diversos. Los emoticonos que expresan alegría u otras emociones positivas se clasifican normalmente como smileys (de smile, «sonrisa» en inglés). En el sistema operativo Windows se pueden obtener smileys pulsando la tecla Alt + 1: ☺X (sonriente blanco) y Alt + 2: ☻ (sonriente negro). Los emoticonos se emplean frecuentemente en mensajes de correo electrónico, en foros, SMS y en los chats mediante servicios de mensajería instantánea.

IDF. El término IDF proviene del inglés Inverse Document Frequency y significa Frecuencia Inversa de Documento. La frecuencia inversa de documento es una medida de si el término es común o no, en la colección de documentos. Se obtiene dividiendo el número total de documentos por el número de documentos que contienen el término, y se toma el logaritmo de ese cociente.

$$IDF(t, D) = \log(\frac{|D|}{|\{d \in D : t \in d\}|})$$

Donde:

|D|: cardinalidad de D, o número de documentos en la colección.

$|\{d \in D : t \in d\}|$: número de documentos donde aparece el término t. Si el término no está en la colección se producirá una división-por-cero. Por lo tanto, es común ajustar esta fórmula a $1 + |\{d \in D : t \in d\}|$

En resumen: el IDF determina la relevancia de un texto con respecto a una palabra clave específica.

LDA. En estadística, Latent Dirichlet Allocation (LDA) es un modelo generativo que permite que conjuntos de observaciones puedan ser explicados por grupos no observados que explican por qué algunas partes de los datos son similares. Por ejemplo, si las observaciones son palabras en documentos, presupone que cada documento es una mezcla de un pequeño número de categorías y la aparición de cada palabra en un documento se debe a una de las categorías a las que el documento pertenece. LDA es un ejemplo de modelo de categorías y fue presentado como un modelo en grafo para descubrir categorías por David Blei, Andrew Ng y Michael Jordan en 2002.

En LDA, cada documento puede verse como una mezcla de varias categorías. Esto es similar a probabilistic latent semantic analysis (PLSA), excepto que en LDA se asume que la distribución de categorías tiene una distribución a priori de Dirichlet. En la práctica, esto resulta en mezclas de categorías en un documento más razonables. Se ha observado, sin embargo, que el modelo PLSA es equivalente al modelo LDA bajo una distribución de Dirichlet a priori uniforme.2

La clave en LDA es que las palabras siguen una hipótesis de bolsa de palabras o, más bien que el orden no importa, que el uso de una palabra es ser parte de un tema y que comunica la misma información sin importar dónde se encuentra en el documento. Esta hipótesis dice que Juan contrató a Pedro es lo mismo que Pedro contrató a Juan. En ambos casos, el conjunto de palabras es la misma junto con la frecuencia de cada palabra. Este supuesto es necesario para que las probabilidades sean intercambiables y que permitan una mayor aplicación de métodos matemáticos. A pesar de que de vez en cuando trata frases semánticamente diferentes como la misma cosa, funciona bien en un documento general.

Máquinas de soporte vectorial. Las máquinas de soporte vectorial o máquinas de vectores de soporte (Support Vector Machines, SVMs) son un conjunto de algoritmos de aprendizaje supervisado desarrollados por Vladimir Vapnik y su equipo en los laboratorios AT&T.

Estos métodos están propiamente relacionados con problemas de clasificación y regresión. Dado un conjunto de ejemplos de entrenamiento (de muestras) podemos etiquetar las clases y entrenar una SVM para construir un modelo que prediga la clase de una nueva muestra. Intuitivamente, una SVM es un modelo que representa a los puntos de muestra en el espacio, separando las clases por un espacio lo más amplio posible. Cuando las nuevas muestras se ponen en correspondencia con dicho modelo, en función de su proximidad pueden ser clasificadas a una u otra clase.

Más formalmente, una SVM construye un hiperplano o conjunto de hiperplanos en un espacio de dimensionalidad muy alta (o incluso infinita) que puede ser utilizado en problemas de clasificación o regresión. Una buena separación entre las clases permitirá una clasificación correcta.

Microblogging. También conocido como nanoblogging, es un servicio que permite a sus usuarios enviar y publicar mensajes breves, generalmente solo de texto. Las

opciones para el envío de los mensajes varían desde sitios web, a través de SMS, mensajería instantánea o aplicaciones ad hoc.

Modelo de bolsa de palabras. El modelo "bolsa de palabras" (del inglés, Bag of Words) es un método que se utiliza en el procesado del lenguaje para representar documentos ignorando el orden de las palabras. En este modelo, cada documento parece una bolsa que contiene algunas palabras. Por lo tanto, este método permite un modelado de las palabras basado en diccionarios, donde cada bolsa contiene unas cuantas palabras del diccionario. En el campo de reconocimiento de objetos, se utiliza una idea similar para las representaciones de imágenes, es decir, una imagen puede ser tratada como un documento y las características extraídas de ciertos puntos de la imagen son consideradas palabras visuales. Las principales ventajas de utilizar este modelo es su facilidad de uso y su eficiencia computacional.

En este modelo el procesamiento de los documentos consta de las siguientes etapas:

1. Preprocesado de los documentos: consiste fundamentalmente en preparar los documentos para su parametrización, eliminando aquellos elementos que se consideran superfluos.

2. Parametrización: es una etapa de complejidad mínima una vez se han identificado los términos relevantes. Consiste en realizar una cuantificación de las características (es decir, de los términos) de los documentos.

Parser. Es un programa que construye los árboles sintácticos. Usualmente se basan en algunas gramáticas formales de varios tipos.

PLSA. El análisis probabilístico de semántica latente (PLSA), también conocido como indexación semántica latente probabilística (PLSI, especialmente en los círculos de recuperaciones de información) es una técnica estadística para el análisis de los modos y los datos de co-ocurrencia. PLSA ha evolucionado desde el análisis semántico latente (LSA), la adición de un modelo probabilístico lo hizo más sólido.

PLSA tiene aplicaciones en la recuperación de la información y filtrado, procesamiento del lenguaje natural, aprendizaje automático a partir del texto, y áreas relacionadas. Fue introducido en 1999 por Juan Puzicha y Thomas Hofmann, y se relaciona con la factorización de matrices no negativas.

Sobreajuste. En aprendizaje automático, el sobreajuste (también es frecuente emplear el término en inglés overfitting) es el efecto de sobreentrenar un algoritmo de aprendizaje con unos ciertos datos para los que se conoce el resultado deseado. El algoritmo de aprendizaje debe alcanzar un estado en el que será capaz de predecir el resultado en otros casos a partir de lo aprendido con los datos de entrenamiento, generalizando para poder resolver situaciones distintas a las acaecidas durante el entrenamiento. Sin embargo, cuando un sistema se entrena demasiado (se sobreentrena) o se entrena con datos extraños, el algoritmo de aprendizaje puede quedar ajustado a unas características muy específicas de los datos de entrenamiento que no tienen relación causal con la función objetivo. Durante la fase de sobreajuste el éxito al responder las muestras de entrenamiento

sigue incrementándose mientras que su actuación con muestras nuevas va empeorando.

Stop words. Palabras de no contenido, son aquellas palabras que en tareas de procesamiento del lenguaje natural no contiene información útil ya que son muy comunes, es decir aparecen en la mayoría de las oraciones como lo son los conectores, auxiliares, preposiciones, etc.

Supervisión lejana. Técnica empleada para clasificar la polaridad de mensajes de acuerdo a si estos contienen emoticones. Por ejemplo, :) en un mensaje indica que el mensaje contiene sentimiento positivo y :(indica que le mensaje contiene sentimiento negativo.

TF·IDF. Tf·idf (del inglés Term frequency – Inverse document frequency), frecuencia de término – frecuencia inversa de documento (o sea, la frecuencia de ocurrencia del término en la colección de documentos), es una medida numérica que expresa cuán relevante es una palabra para un documento en una colección. Esta medida se utiliza a menudo como un factor de ponderación en la recuperación de información y la minería de texto. El valor tf·idf aumenta proporcionalmente al número de veces que una palabra aparece en el documento, pero es compensada por la frecuencia de la palabra en la colección de documentos, lo que permite manejar el hecho de que algunas palabras son generalmente más comunes que otras.

Twitter. Es un término inglés que puede traducirse como “gorjear” o “trinar”, es el nombre de una red de microblogging que permite escribir y leer mensajes en Internet que no superen los 140 caracteres. Estas entradas son conocidas como tweets**.**

Validación cruzada. La validación cruzada o cross-validation es una técnica utilizada para evaluar los resultados de un análisis estadístico y garantizar que son independientes de la partición entre datos de entrenamiento y prueba. Consiste en repetir y calcular la media aritmética obtenida de las medidas de evaluación sobre diferentes particiones. Se utiliza en entornos donde el objetivo principal es la predicción y se quiere estimar cuán preciso es un modelo que se llevará a cabo a la práctica. Es una técnica muy utilizada en proyectos de inteligencia artificial para validar modelos generados.