Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
Minería de Datos
Integrantes:

Leslie Rodrigues	10-10613
Edward Fernández	10-11121
Joel Rivas	11-10866
Georvic Tur	12-11402

Detección de Spam en Twitter

Capítulo I: Introducción

Las redes sociales son cada vez más utilizadas y en los últimos años se han convertido en los sitios con mayor número visitantes. Entre los que resalta encontramos *Twitter*, por ser uno de los sitios con más rápido crecimiento. Sin embargo, su popularidad atrae a usuarios maliciosos que tienden a propagar *spam*, mensajes falsos y/o *malware*.

En este informe, se describen los pasos seguidos para construir y evaluar un modelo basado en los *random forests* con el fin de detectar *spam* y se compara con otros tres modelos. Específicamente, se busca detectar noticias falsas en cuentas de *Twitter* que representen portales de noticias bien conocidos. Esto puede presentar ciertos inconvenientes. En primer lugar, a diferencia de [1], no resulta práctico determinar para cada tuit si contiene una noticia falsa o verdadera. En el trabajo de investigación mencionado, determinar si un tuit es *fake* resulta más sencillo, debido a la especificidad del tema estudiado por el autor. En dicho caso, el conjunto de datos tratado hacía referencia a dos desastres naturales particulares (El huracán Sandy y el tornado Moore). Por esto, determinar si una noticia es falsa implica conocer cierta información sobre el tópico de estudio para verificar su veracidad, por ejemplo, la hora y fecha del evento [1] (p. 6). En segundo lugar, tal como lo señala [3](pp. 68-69), usar los atributos relacionados con el historial de tuits de cada cuenta presenta grandes dificultades, debido a los límites impuestos por la empresa *Twitter* en el número de peticiones, y el poder de cómputo requerido para procesarlos. Esto último es un aspecto fundamental, pues se pretende desarrollar al final de esta investigación, una herramienta como la que propone el autor en la publicación [3]. Otra limitación es lidiar con cuentas de Twitter de conocidos portales de noticias, ya que no son usuarios *spammer* típicos como los vistos en [1], [2] y [3]. Por esta razón, la definición de *spam* ofrecida por dichos autores puede que no sea la más adecuada para esta investigación.

Teniendo en cuenta las limitaciones mencionadas, se pretende evaluar si el común denominador de los modelos presentados por dichos autores, así como sus metodologías, se pueden aplicar a la detección de noticias falsas. De ser este el caso, el modelo que será producto de esta investigación será de gran utilidad para los usuarios de redes sociales que frecuenten esta clase de portales informativos. Entre ellos se puede mencionar a periodistas y políticos que deseen una herramienta que les llame la atención ante un posible caso de noticia falsa, para que luego éstos puedan indagar si efectivamente se trata de *spam*. Finalmente, en esta investigación, a diferencia de las antedichas, se usa análisis de tópicos en conjunto con el análisis estadístico de los tuits. Esto podría contribuir con el refinamiento de los resultados presentados por dichos autores.

Objetivos

Objetivo General

- Detectar los spams de las noticias en CNN usando mensajes de microblogs.

Objetivos Específicos

- Programar y buscar herramientas que contribuyan con el proceso de recolección de tuits y su procesamiento
- Seleccionar los atributos a usar en el modelo de clasificación
- Procesar los tuits obtenidos para generar los vectores de atributos que usará el modelo a entrenar.
- Explorar los datos para confirmar que los atributos seleccionados contribuyen en la clasificación de los tuits
- Seleccionar el algoritmo de clasificación
- Entrenar el modelo de clasificación seleccionado
- Evaluar el modelo
- Describir el programa construido con el modelo

Estado del Arte

Como parte de las investigaciones similares a la presente, se puede mencionar la hecha por [2] y [3]. En el caso [3], el objetivo fue entrenar un modelo de *random forest,* que luego pudiera ser usado en una futura aplicación de fácil integración a un navegador web, con la finalidad de clasificar tuits en *spam* y *no spam*. Los autores de esta investigación eligieron este modelo basándose en los resultados de [2], donde se deseaba detectar tuits spam utilizando Clasificadores tradicionales, obteniendo un 95.7% de precisión.

Adicionalmente, esta herramienta debería clasificar las publicaciones en diversas clases definidas por el autor. A diferencia de [3], en la presente investigación sólo se busca clasificar en *spam* y *no spam*.

Por otra parte, dicho autor [3] define dos metodologías para entrenar el modelo. La primera es de una fase y está destinada a mejorar la clasificación binaria. La segunda es de dos fases y su objetivo es mejorar la clasificación multiclase. Los autores de este informe desean utilizar una clasificación binaria. Por eso, no se necesita usar la metodología de dos fases de [3].

Finalmente, [3] también muestra cómo se puede usar el enfoque de teoría de redes, para hacer un análisis de redes sociales aplicado a la detección de usuarios *spammers*. El autor no encontró buenos resultados con este enfoque y, por lo tanto, no está planteado su uso en esta investigación. Otro factor, por el cual no está planeado su uso, es que las cuentas de usuarios estudiadas pertenecen a cadenas de noticias bien conocidas. Esto se debe a que se conoce cuáles son los usuarios, y sólo se quiere determinar cuáles de sus tuits pueden ser considerados como noticias falsas.

Algoritmos Usados

Fueron usados cuatro algoritmos de clasificación: Naive Bayes, Adaboost, KNN y Random Forest.

Naive Bayes: este es un algoritmo de clasificación cuya característica más notoria es la asunción de que los atributos de los datos estructurados son independientes entre sí. Para elegir la mejor clasificación se basa en el principio de la máxima probabilidad posterior (Marshland, pp. 29-30).

Adaboost: es un meta algoritmo de aprendizaje por comités. Su característica más notoria es la asignación de prioridades a aquellos atributos de los datos estructurados que más contribuyen en aumentar el error (Marshland, p. 273). Esta técnica busca reducir la contribución del sesgo en el error.

KNN: es un algoritmo de agrupamiento que requiere conocer de antemano el número de puntos maś cercanos a considerar. Por esta razón, es importante definir una métrica adecuada. La clase a la cual pertenece cada punto se decide por votación entre sus vecinos más cercanos (Marshland, p. 158).

Random Forest: la principal característica de este método es la reducción de la contribución de la varianza, sin disminuir mucho el sesgo, en el error. Esto se debe al empleo de un método de votación entre variedades de árboles de decisión, cada uno entrenado con una muestra con reemplazo del conjunto de datos original, además de estar sujeto cada uno a la restricción de tener que optimizar sobre subconjuntos distintos del conjunto total de atributos (Marshland, p. 275).

Capítulo II: Metodología

Preprocesamiento de Datos

Se descargaron 3000 tuits de las cuentas de *Twitter*: *CaraotaDigital*, *la_patilla* y *CNNEE*, con *Birdwatcher*, de las cuales se tomó una muestra aleatoria de 946 tuits, para luego ser clasificados manualmente (en *spam* o *no spam*), siguiendo las recomendaciones de [3] (p. 19, sección 2.3) para las distintas clases de tuits y de [1] (p. 6) para las los tuits *fake.* Para clasificarlos se usó una herramienta que se ha programado y que se basó en el trabajo descrito por [3] (p. 30, sección 3.3). De dichos tuits ya clasificados, se usaron 700 como conjunto de entrenamiento, tal como se puede ver en la tabla N° 3. Para el conjunto de validación se apartaron 246 tuits. La distribución de estos últimos se muestra en la tabla N° 4. Para la prueba final, se descargó, en una fecha distinta, 1000 tuits sólo de la cuenta de Twitter *CNNEE* y se tomó una muestra de 265 de ellos para ser clasificados manualmente usando la antedicha herramienta.

Al mismo tiempo, se usó el texto de cada tuit para computar ocho atributos: número de *hashtags*, número de menciones, número de letras mayúsculas, número de caracteres no alfanuméricos, número de direcciones *URL*, longitud del tuit, cantidad de números en el tuit y la clase (si es spam o no, como se indicó arriba) como se muestra en [2]. A diferencia de [3], sólo se están usando los atributos propios de cada tuit y no se están tomando en cuenta los atributos relacionados con las cuentas de los usuarios ni con el historial de tuits. Un ejemplo de esto se puede apreciar en la tablas N° 1 y N° 2.

Por otra parte, además de los atributos indicados por los autores [1], [2] y [3], se usaron tópicos para poder discriminar los tuits. Para poder obtener los tópicos fue necesario partir de los textos de cada uno de éstos. En este sentido, se ha tratado a cada tuit como un documento y se ha tomado cada palabra de dicho documento como una unidad. Antes de calcular los tópicos, fue necesario remover las palabras más comunes que no contribuyen en nada al significado (*stopwords*). Así mismo, se removieron todas las vocales con tildes al transformarlas en su equivalente *ASCII* más cercano. Finalmente, sólo para obtener los tópicos, fueron transformadas todas las letras a minúsculas. Una vez terminado el preprocesamiento para los tópicos, se calculó la matriz de frecuencias de términos y se le aplicó una técnica conocida como factorización no negativa de matrices, o *NMF* por sus siglas en inglés. Este, es un algoritmo determinístico que obtiene el tópico de cada documento (tuit). En este paso se usaron todos los tuits descargados, sin importar si estaban clasificados o no. Los tópicos obtenidos se puede apreciar en la tabla N° 5.

Otros Pasos de Preprocesamiento

A partir del conjunto de datos de entrenamiento, se generaron otros *dataset* utilizando otros filtros de Weka. En primer lugar, se usó *InterquartileRange* para marcar las instancias consideradas *outliers* y valores extremos. Luego se aplicaron los filtros *normalize* y *randomize*.

Con estos, se probaron los algoritmos Naive Bayes, Adaboost, KNN y Random Forest. En el caso de Naive Bayes y Adaboost se aplicó adicionalmente el filtro *discretize*, con diez intervalos. Para comparar resultados con y sin *outliers* y valores extremos, se usó *removeWithValues*.

Además de esto, para comparar resultados anteriores, se generó otro conjunto de datos a partir del *dataset* de entrenamiento sin el atributo de clase *spam*. Este fue llenado, para cada instancia, de manera manual. Por esta razón, también se desea evaluar la influencia de dicha clasificación usando un algoritmo de *clustering*. En este caso, se usó K-Means, con distancia euclídea y dos centros, para clasificar de manera automática el conjunto de datos. Para lograr esto, se usó el filtro *addCluster* de Weka. En este caso, no se aplicaron los antes mencionados.

Atributos de los tuits obtenidos con Birdwatcher

A continuación sólo indicamos los atributos de los objetos *status* obtenidos con Birdwatcher. Dichos atributos son una mezcla de las características ofrecidas por el API de Twitter y aquéllas que la herramienta Birdwatcher guarda en su esquema *statuses*.

> Id,workspace_id,user_id,twitter_id,text,source,retweet,geo,longitude,latitude,place_type,place_name,place_country_code,place_country,favorite_count,retweet_count,sentiment,possibly_sensitive,lang,posted_at,updated_at,created_at

De estos atributos sólo interesan el texto y el identificador para el entrenamiento. Más adelante resultará necesario tomar en cuenta atributos que no dependen estrictamente del texto, sino también de las características de redes como el número de retuits y el número de favoritos.

Atributos de los datos estructurados

A continuación se describe en detalle cada uno de los atributos seleccionados para el entrenamiento.

tweet_id	hashtags	mentions	uppercase	nonalpha	urls	len	numbers	spam

Identificador del tuit	Cantidad de hashtags	Cantidad de menciones	Cantidad de mayúsculas	Cantidad de caracteres no alfanuméricos (toma en cuenta no visibles y espacios)	Cantidad de direcciones URL	Longitud del tuit	Cantidad de números en el tuit (no parte del URL)	Si es spam o no
11123	0	0	10	15	2	132	0	n

Tabla N° 1: ejemplo de vector de atributos

tweet_id	texto
11123	"Muere la venezolana Sofía Ímber, periodista y defensora de los derechos de la mujerhttps://t.co/tuSi0bjq8W https://t.co/MXXwspDdJI"

Tabla N° 2 : texto del tuit del cual se extrajeron los atributos mostrados en la tabla N° 1

Como ya se ha mencionado, hay tres atributos cuya presencia varía entre algunas evaluaciones. Estos son: número de favoritos que ha recibido el tuit, número de retuits y el identificador del tópico al que fue asignado

Estadísticas

Las tablas N° 3 y N° 4 indican que nos enfrentamos a un problema de distribución no balanceada de los datos. Este resulta común, pues se encontró en los tres trabajos estudiados como antecedentes [1], [2] y [3]. De los tres, [3] fue el único que trató de subsanar este inconveniente con ayuda de un clasificador automático de tuits. En este informe también se describirán las evaluaciones con los mismos conjuntos de datos etiquetados manualmente.

spam	No spam	Total
136	564	700

Tabla N° 3: distribución de las clases para el conjunto de entrenamiento

spam	No spam	Total
45	201	246

Tabla N° 4: distribución de las clases para el conjunto de validación

Los tópicos de la tabla N° 5 reflejan la realidad de las fechas cuando fueron generados los tuits. Se puede observar que la mayor parte de los mismos muestra la realidad política de las Américas. El tópico noveno, en cambio, alude a eventos alejados de la realidad latinoamericana, lo cual se refleja en la cantidad de sus tuits. El tercero alude a una noticia muy

puntual que casualmente tuvo mucha popularidad en las redes sociales y que se reportó de diversas formas.

id_topic	wordsbag	Cantidad de tuits	spam	No spam
0	'video chavista periodista sentada dtb'	75	34	41
1	'venezuela vivo cnn soyfdelrincon senal'	115	18	97
2	'trump donald mes opinion mexico'	102	18	84
3	'muerte dolares angeles prostitutas venden'	17	6	11
4	'ecuador vuelta cne lomasvisto resultados'	106	5	101
5	'uu ee indocumentados inmigrantes aissami'	72	6	66
6	'sports illustrated 2017 rubia portada'	28	11	17
7	'fotos lomasvisto anos accesorios orgullo'	108	32	76
8	'muere brutal embarazada companeras recibir'	41	2	39
9	'jong kim nam muerte corea'	36	4	32

Tabla N° 5: tópicos obtenidos con sus respectivas cinco palabras más representativas y cantidad de tuits asignados en el dataset de entrenamiento. (La cantidad de tuits fue obtenida con weka)

Atributo	Promedio	Desviación estándar	Máximo	Mínimo
hashtags	0.323	0.539	3	0
mentions	0.117	0.418	2	0
uppercase	10.323	3.905	32	3
nonalpha	12.846	4.421	27	2
urls	1.44	0.557	2	0

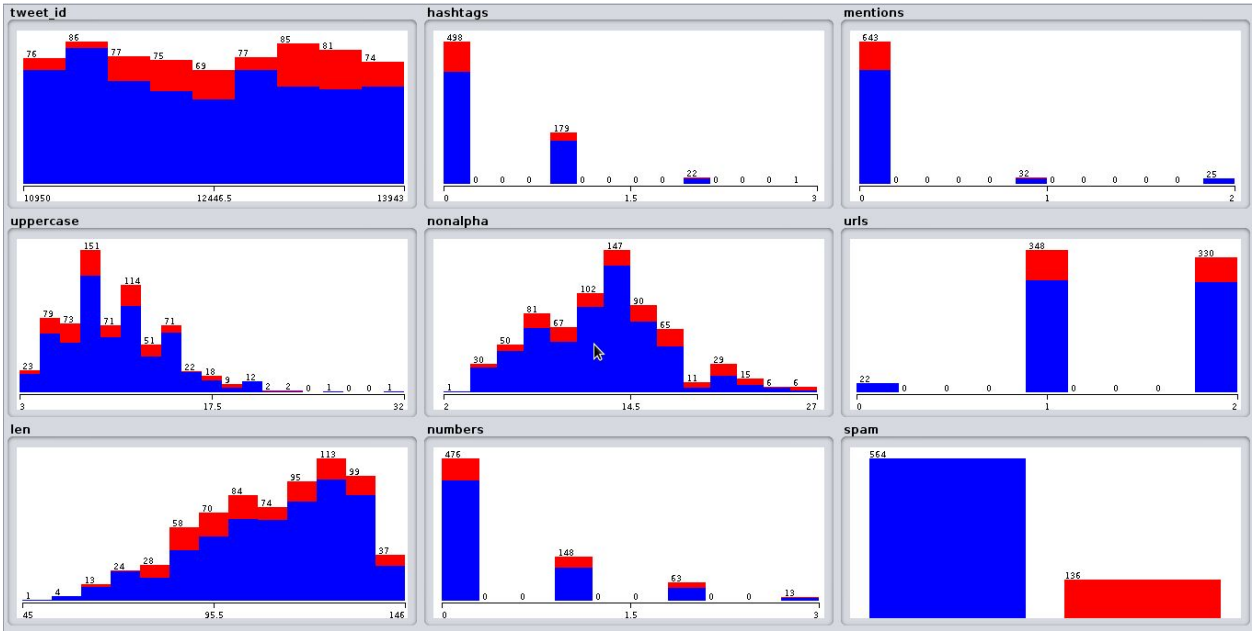
len	111.243	20.17	146	45
numbers	0.447	0.734	3	0

Tabla N° 6: resumen de cada atributo obtenido con Weka para el conjunto de entrenamiento

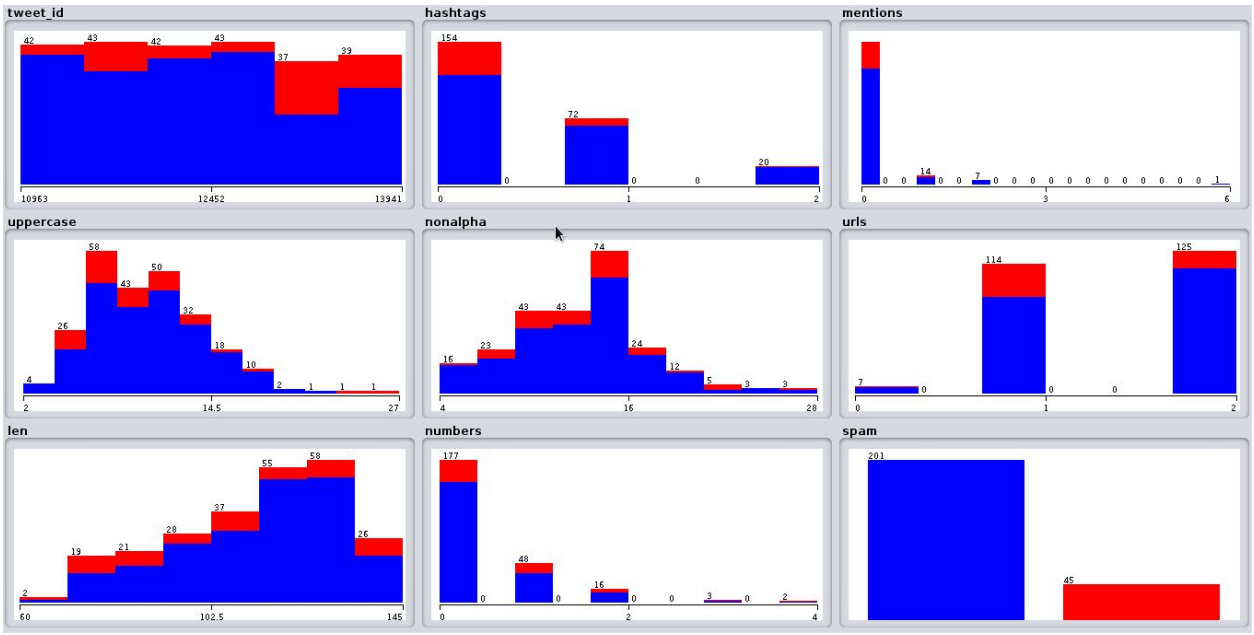
Atributo	Promedio	Desviación estándar	Máximo	Mínimo
hashtags	0.455	0.642	2	0
mentions	0.138	0.547	6	0
uppercase	10.459	3.772	27	2
nonalpha	13.244	4.486	28	4
urls	1.48	0.555	2	0
len	113.008	18.982	145	60
numbers	0.394	0.736	4	0

Tabla N° 7: resumen de cada atributo obtenido con Weka para el conjunto de pruebas

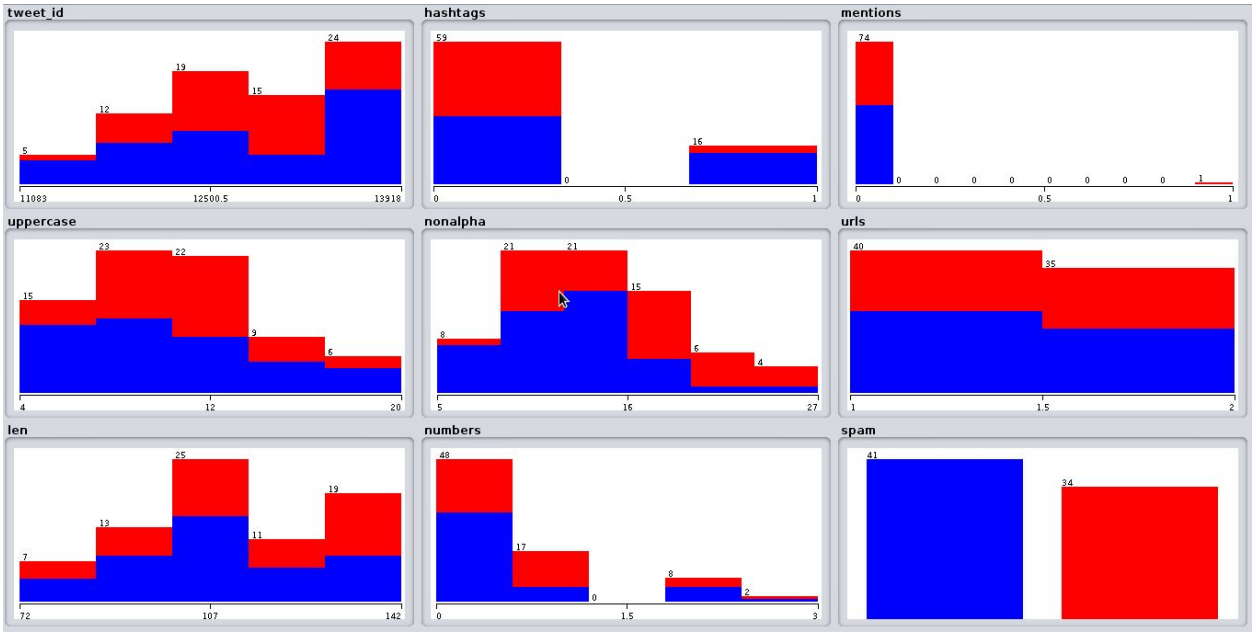
Exploración de Datos



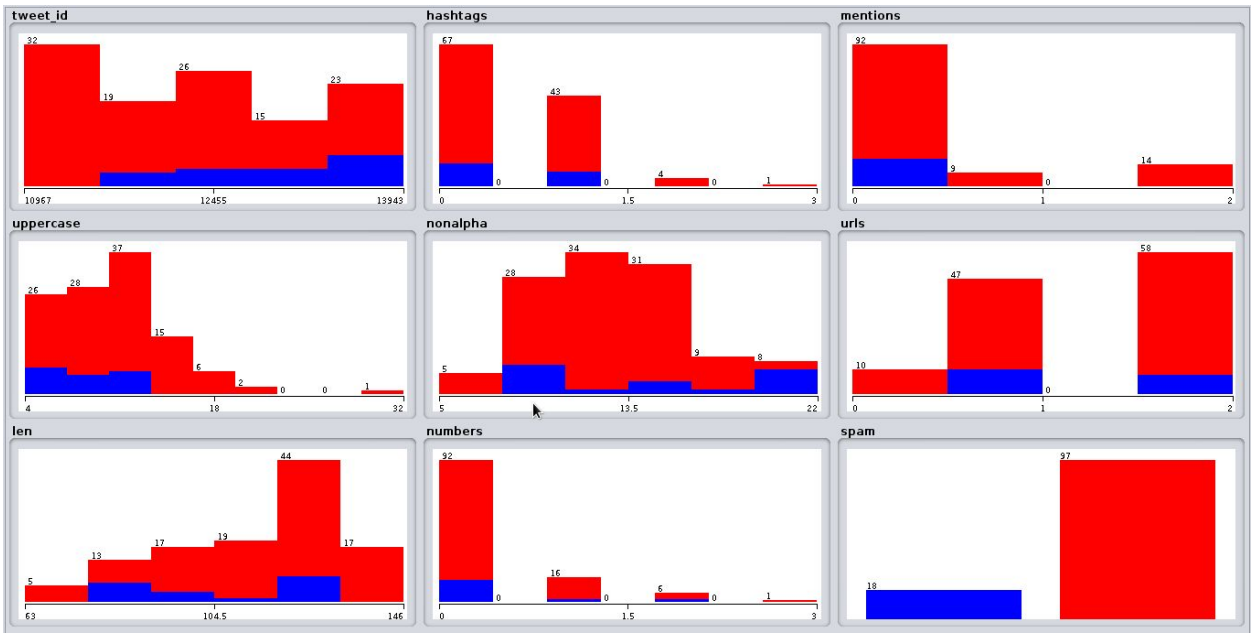
Gráfica N° 1 : histrogramas del conjunto de entrenamiento. Azul: sanos. Rojo: spam



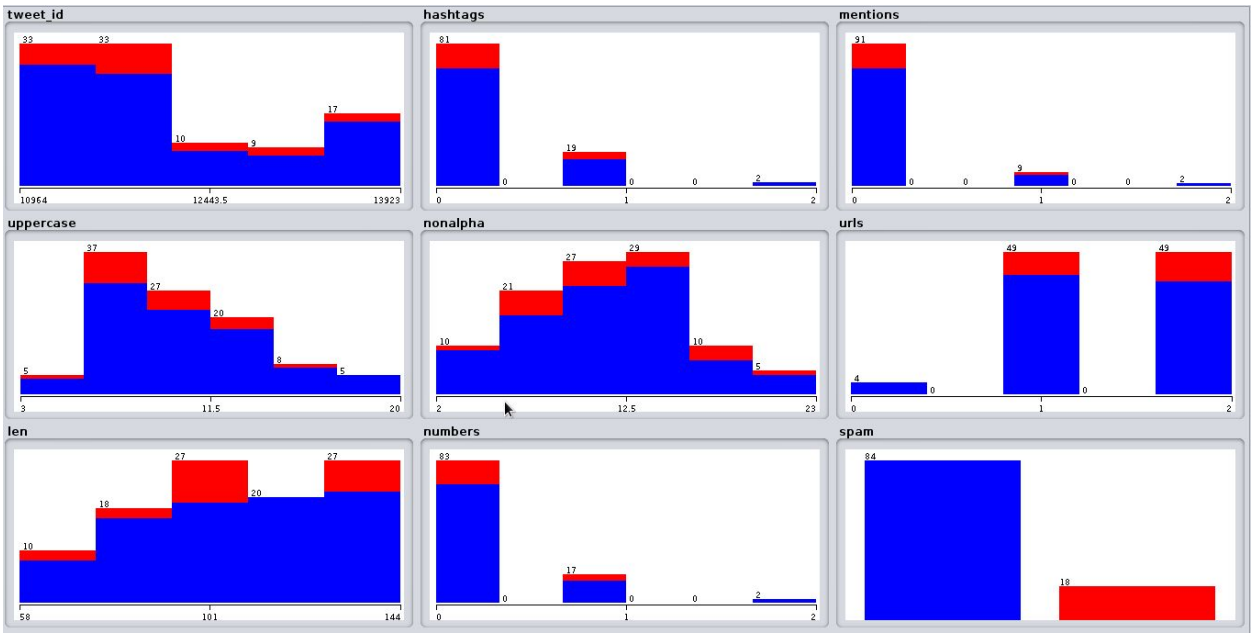
Gráfica N° 2 : histrogramas del conjunto de validación. Azul: sanos. Rojo: spam



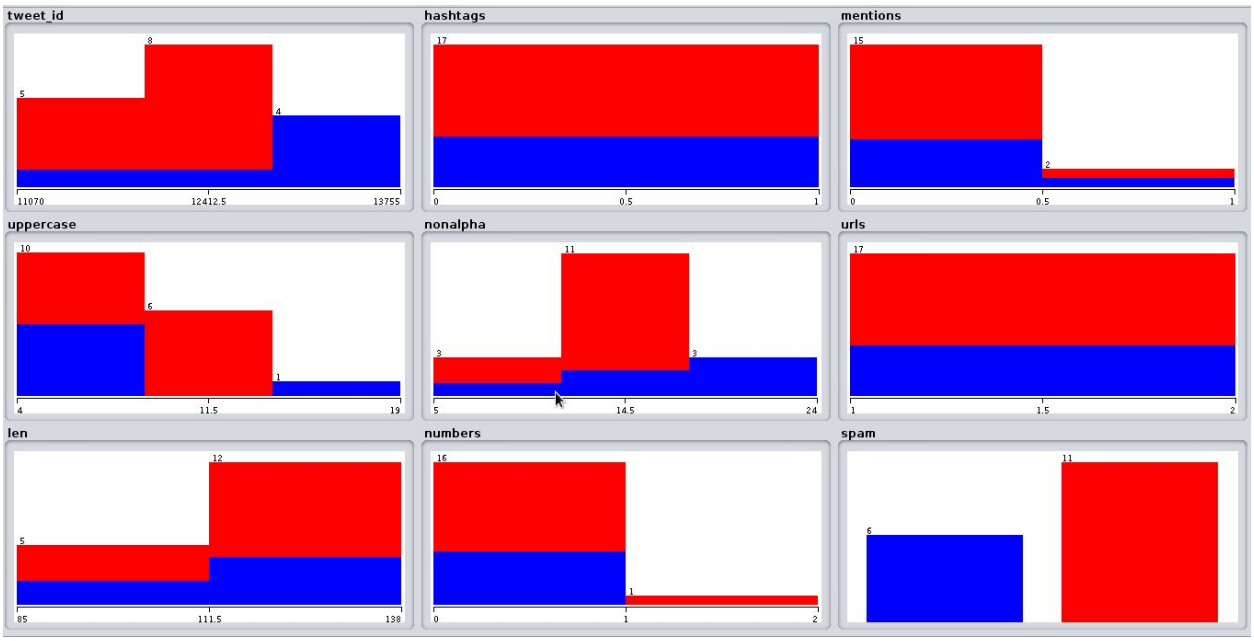
Gráfica N° 3 : histrogramas del conjunto de entrenamiento según el tópico 0. Azul: sanos. Rojo: spam



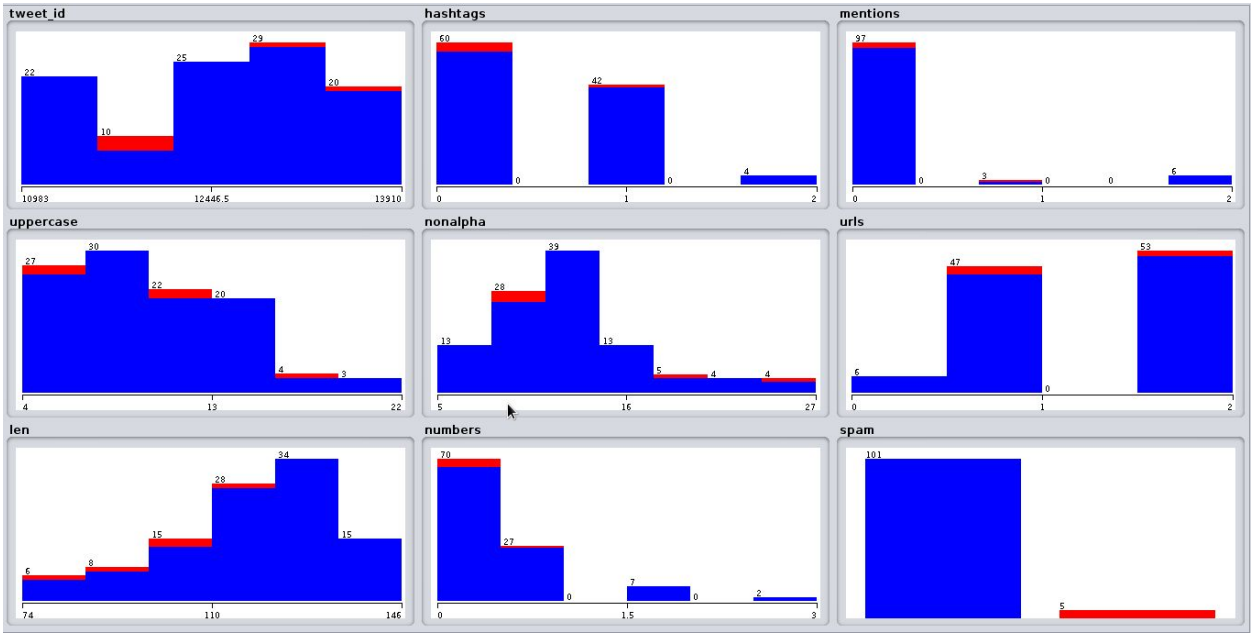
Gráfica N° 4 : histrogramas del conjunto de entrenamiento según el tópico 1. Azul: spam. Rojo: sano



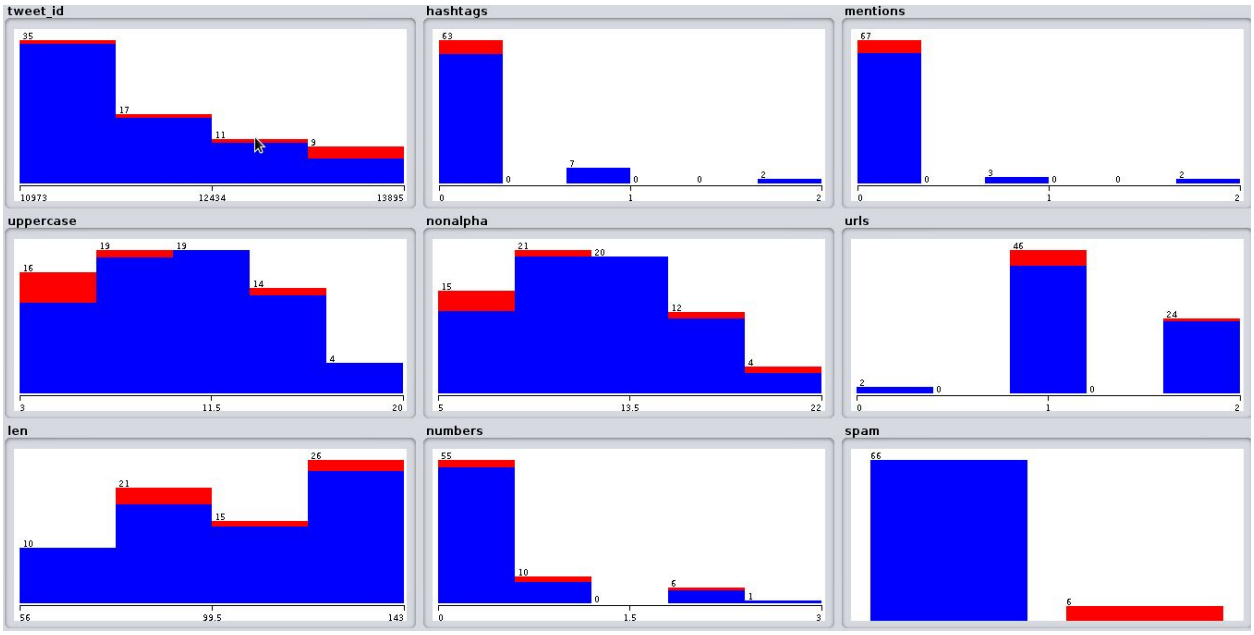
Gráfica N° 5 : histrogramas del conjunto de entrenamiento según el tópico 2. Azul: sanos. Rojo: spam



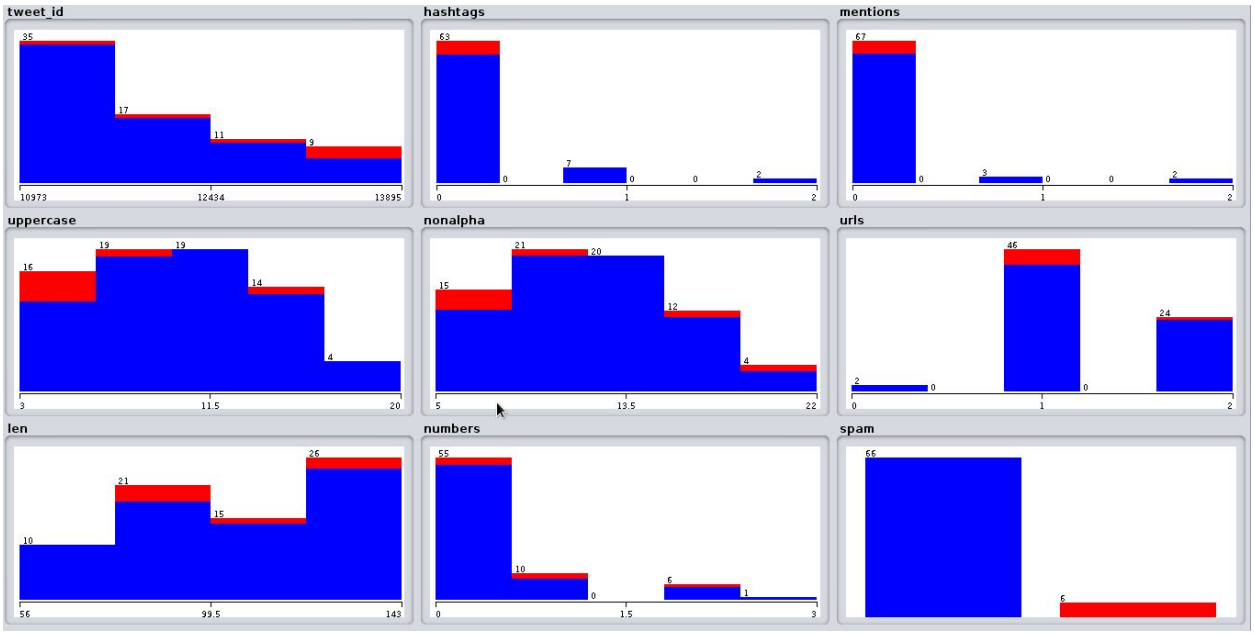
Gráfica N° 6 : histrogramas del conjunto de entrenamiento según el tópico 3. Azul: spam. Rojo: sanos



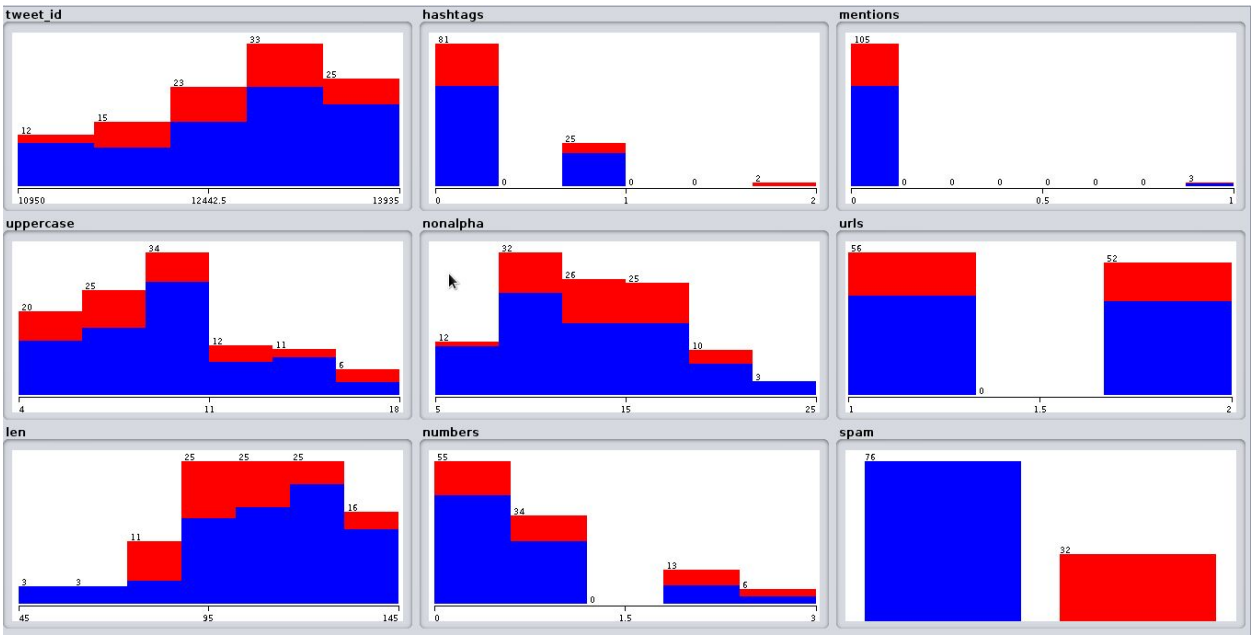
Gráfica N° 7 : histrogramas del conjunto de entrenamiento según el tópico 4. Azul: sanos. Rojo: spam



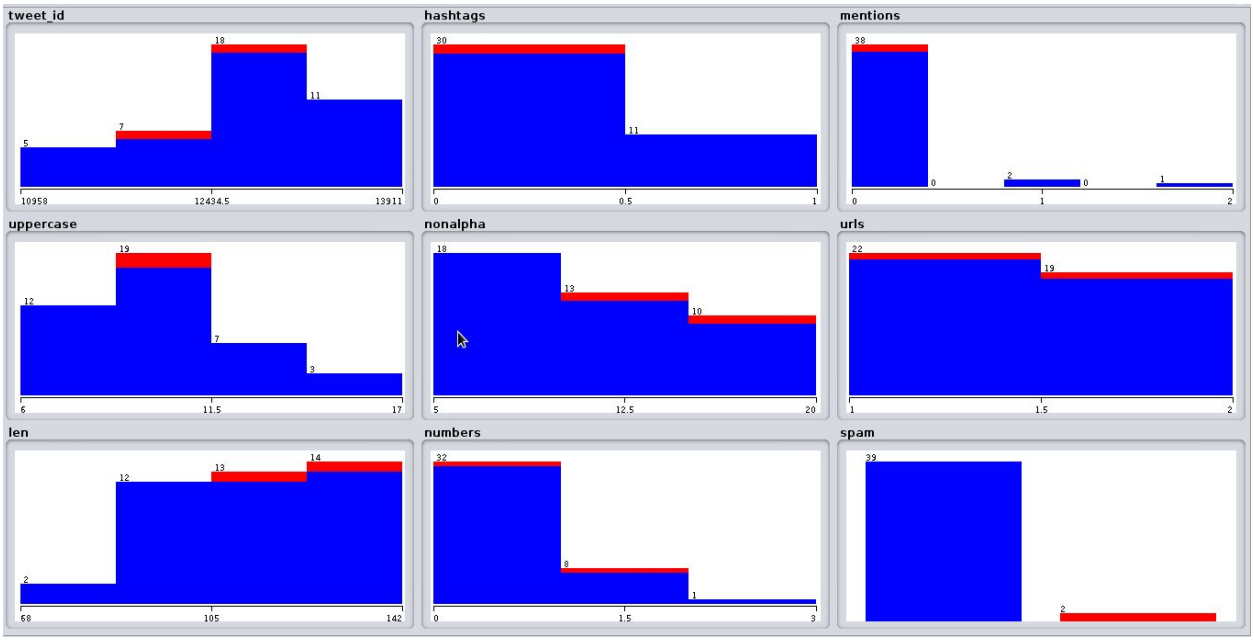
Gráfica N° 8 : histrogramas del conjunto de entrenamiento según el tópico 5. Azul: sanos. Rojo: spam



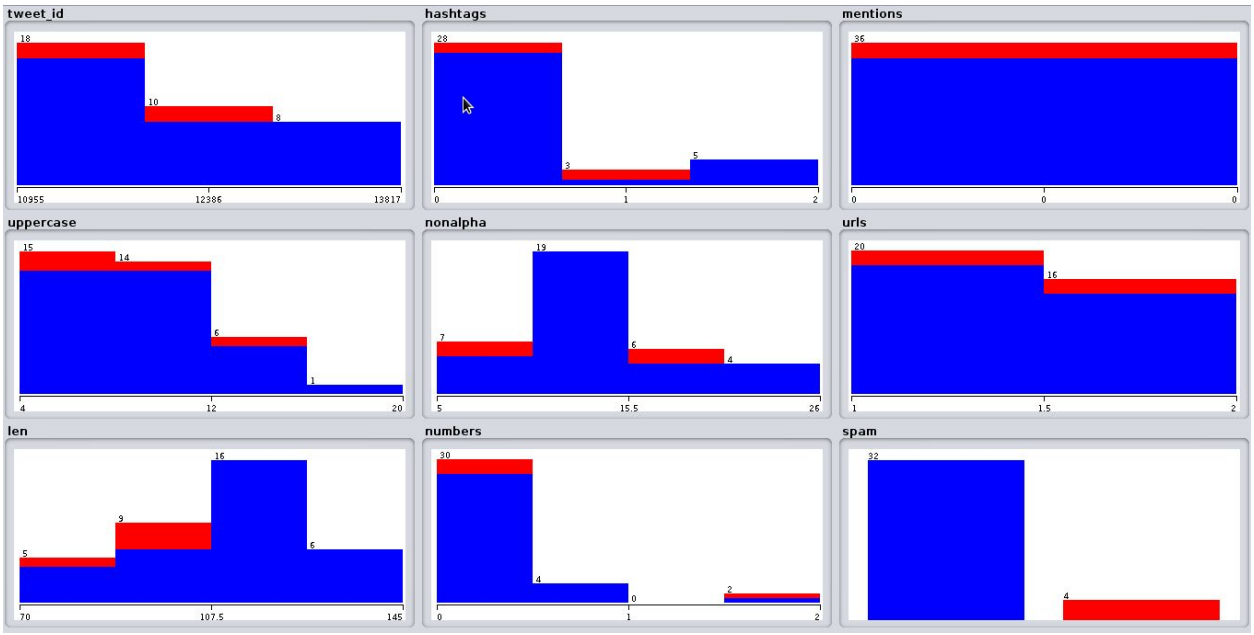
Gráfica N° 9: histrogramas del conjunto de entrenamiento según el tópico 6. Azul: sanos. Rojo: spam



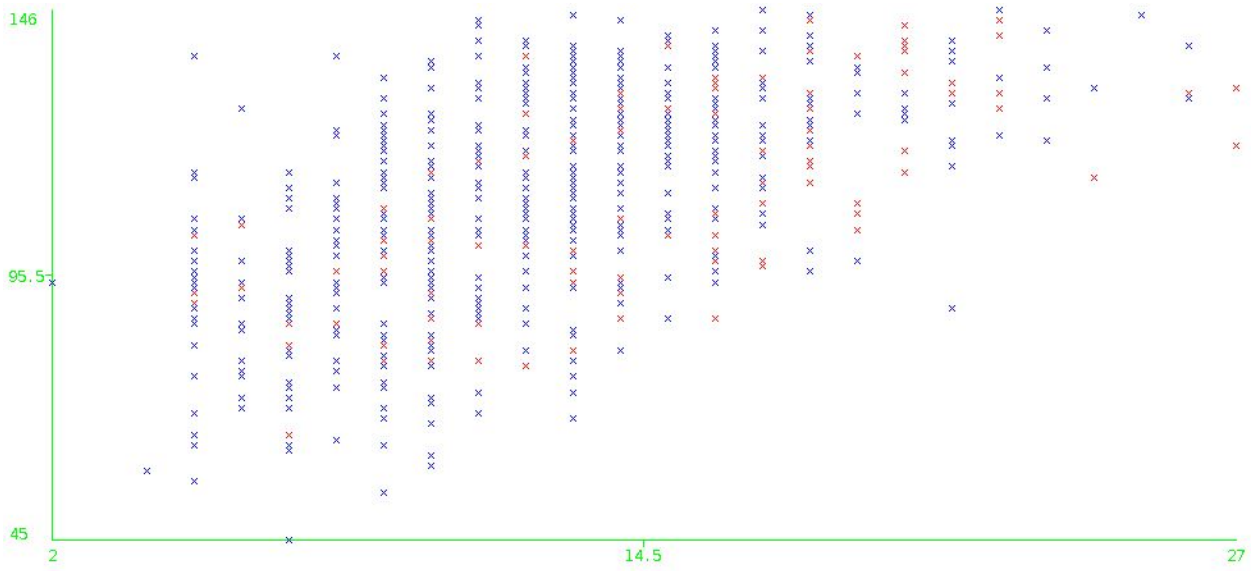
Gráfica N° 10: histrogramas del conjunto de entrenamiento según el tópico 7. Azul: sanos. Rojo: spam



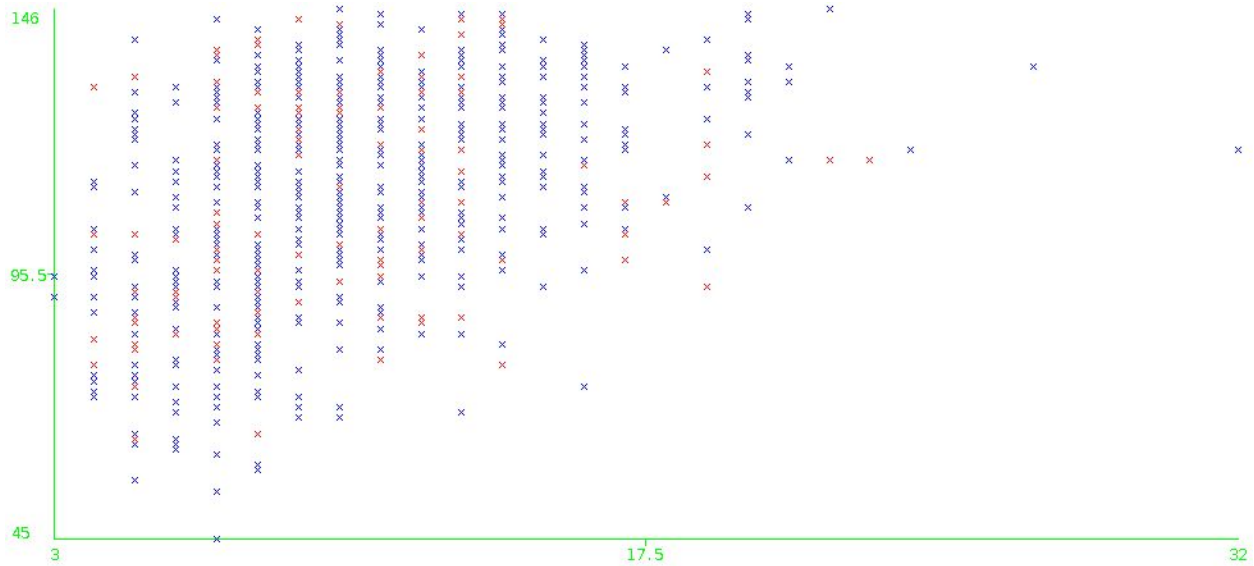
Gráfica N° 11 : histrogramas del conjunto de entrenamiento según el tópico 8. Azul: sanos. Rojo: spam



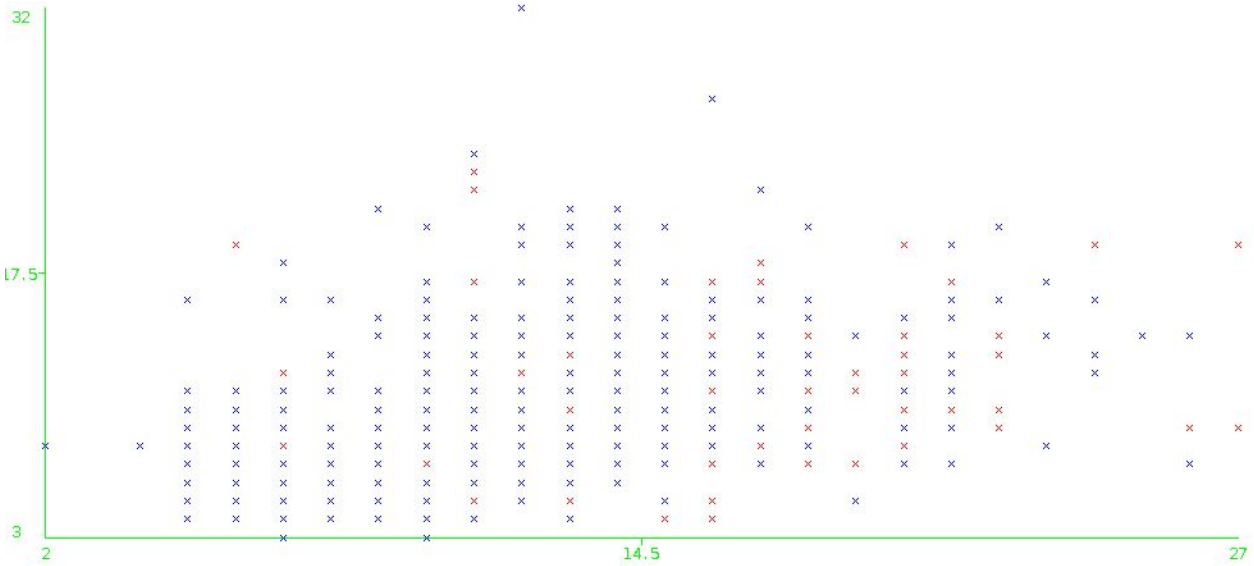
Gráfica N° 12 : histrogramas del conjunto de entrenamiento según el tópico 9. Azul: sanos. Rojo: spam



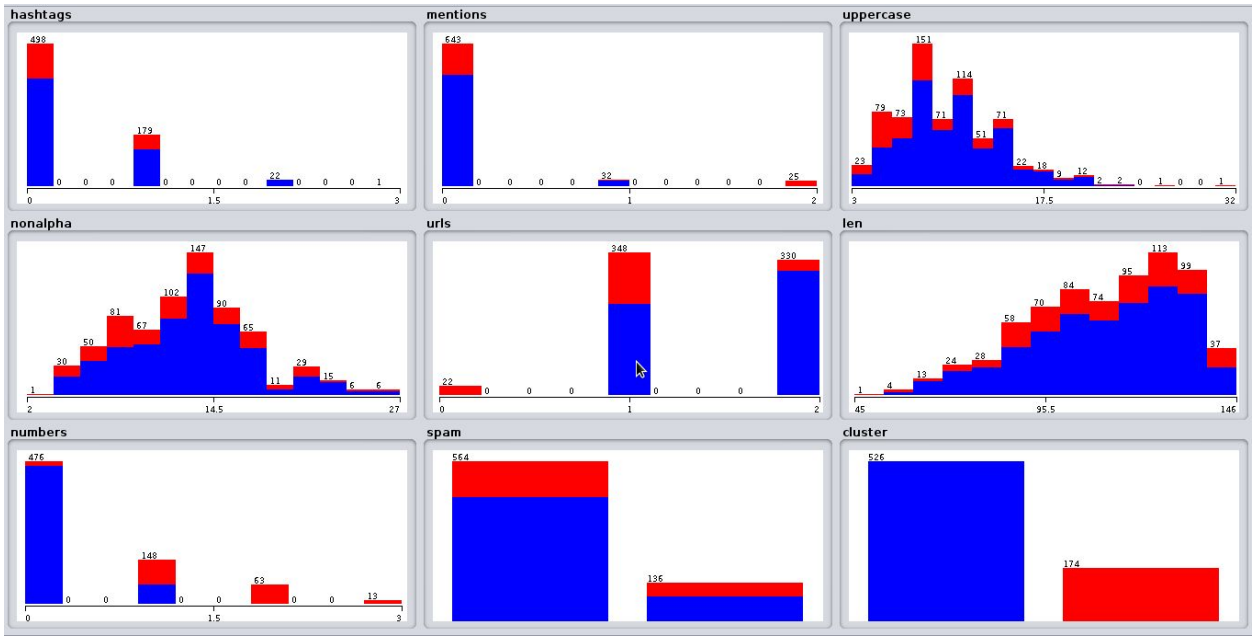
Gráfica N° 13 : scatterplot del conjunto de entrenamiento según los atributos nonalpha y len. Azul: sanos. Rojo: spam



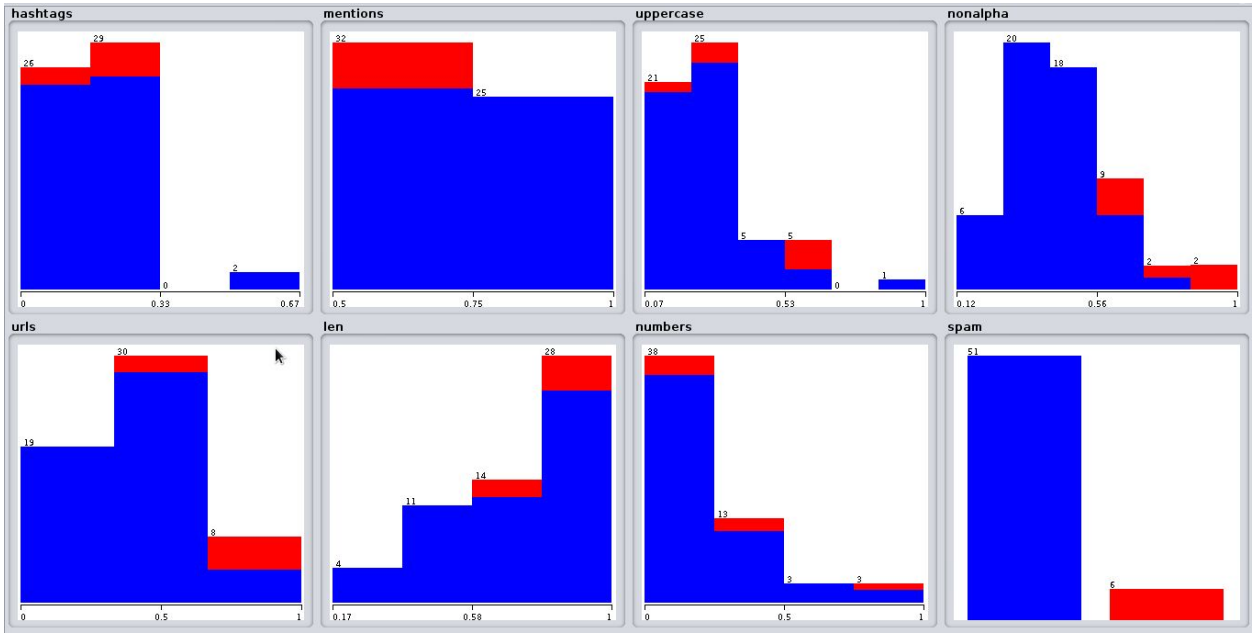
Gráfica N° 14 : scatterplot del conjunto de entrenamiento según los atributos uppercase y len. Azul: sanos. Rojo: spam



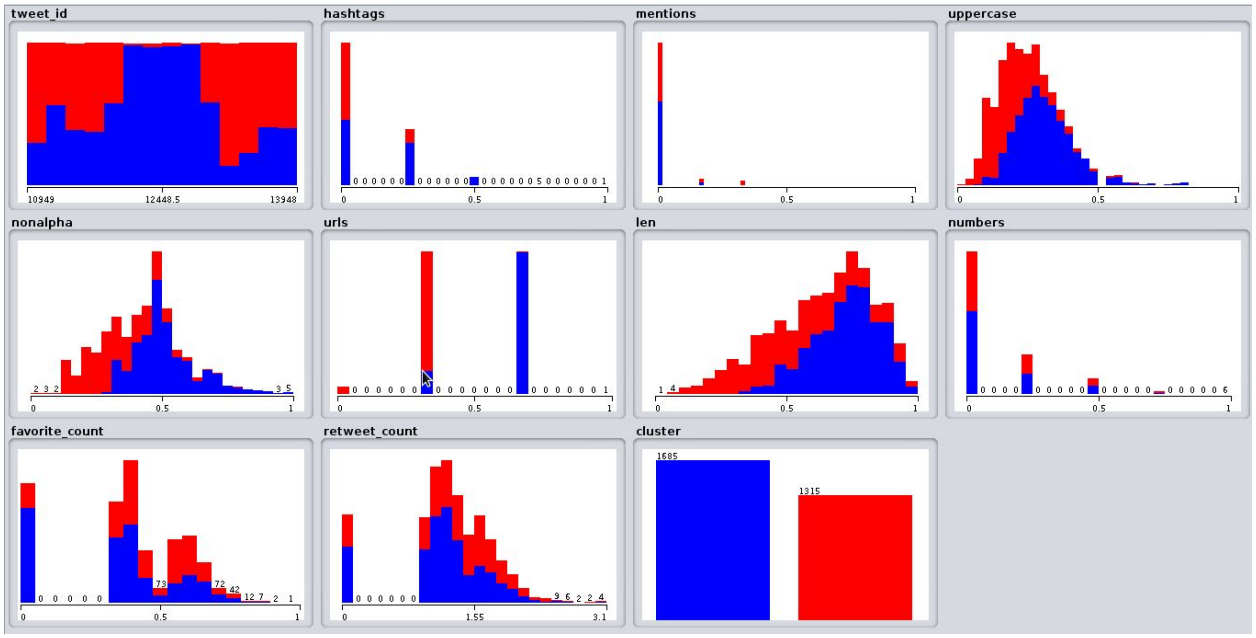
Gráfica N° 15 : scatterplot del conjunto de entrenamiento según los atributos nonalpha y uppercase. Azul: sanos. Rojo: spam



Gráfica N° 16 : histograma para el conjunto de entrenamiento con clase asignada automáticamente con K-Means. Azul: sanos. Rojo: spam



Gráfica N° 17 : histograma para los valores extremos del conjunto de entrenamiento. Azul: sanos. Rojo: spam. Sólo había un *outlier*.



Gráfica N° 18: histogramas para el conjunto original de 3000 tuits etiquetados automáticamente por Weka

Entrenamiento

Se normalizó el conjunto de entrenamiento clasificado manualmente y el clasificado de manera automática. Luego se aplicaron los cuatros algoritmos mencionados.

Fue necesario incluir un nuevo conjunto de datos con dos atributos adicionales: la cantidad de favoritos y la cantidad de retuits. También se incluyó al final como atributo el identificador del tópico que le fue asignado a cada tuit.

Evaluación

A continuación, se muestran los resultados para dos conjuntos de datos: el conjunto de entrenamiento clasificado manualmente, siguiendo los criterios de [3], y el conjunto de datos clasificado automáticamente, con el filtro *addCluster* de Weka. En ellos se observa que los criterios seleccionados por este autor no son útiles para la detección de *spam* en tweets de cuentas de noticias de Twitter. Esto se hace evidente por las métricas de *Recall* y *Precision*, las cuales son sensibles a la clase positiva (*spam*). Por lo tanto, el *acuracy* que muestran los algoritmos para el conjunto de datos clasificado manualmente responde principalmente a su capacidad de detectar tuits sanos (no *spam*). Por esta razón, la clasificación dada automáticamente resulta mucho mejor.

Adicionalmente, como se vio en la gráfica N° 16, la clase asignada automáticamente no corresponde exactamente con la clasificación manual.

De las tablas mostradas a continuación, todas las métricas fueron obtenidas a través de Weka, salvo por el *Negative Predictive Value* y el *Specificity*. Estas últimas fueron calculadas a partir de los datos obtenidos.

Con atributo de clase *spam* colocado manualmente

Estos son los resultados obtenidos con *cross-validation* usando el conjunto de entrenamiento clasificado de manera manual. Como se puede ver, los resultados para la clase positiva (*spam*) no son aceptables.

Algoritmo	Accuracy	Recall	Specificity	Precision	NPV	AUCROC
Naive Bayes	79.8571 %	0,977	0,059	0,811	0,380	0,687
AdaBoost	80.2857 %	0,995	0,001	0,806	0,250	0,630
KNN	81.7143 %	0,950	0,265	0,843	0,563	0,667
RandomForest	80.2857 %	0,931	0,272	0,841	0,487	0,689

Tabla N° 8: Para la clase sana (Manual) con el conjunto de la Tabla N° 3 y cross-validation

Algoritmo	Accuracy	Recall	Specificity	Precision	NPV	AUCROC
Naive Bayes	79.8571 %	0,059	0,977	0,381	0,811	0,687
AdaBoost	80.2857 %	0,007	0,995	0,250	0,806	0,630
KNN	81.7143 %	0,265	0,950	0,563	0,843	0,667
RandomForest	80.2857 %	0,272	0,931	0,487	0,841	0,689

Tabla N° 9: Para la clase spam (Manual) con el conjunto de la Tabla N° 3 y cross-validation

	spam (clasificado)	no spam (clasificado)
spam (real)	8	128
no spam (real)	13	551

Tabla N° 10: Matriz de confusión del Naive Bayes (Manual) con el conjunto de la Tabla N° 3

	spam (clasificado)	no spam (clasificado)
spam (real)	1	135
no spam (real)	3	561

Tabla N° 11: Matriz de confusión del AdaBoost (Manual) con el conjunto de la Tabla N° 3

	spam (clasificado)	no spam (clasificado)
spam (real)	36	100
no spam (real)	28	536

Tabla N° 12: Matriz de confusión del KNN (Manual) con el conjunto de la Tabla N° 3

	spam (clasificado)	no spam (clasificado)
spam (real)	37	99
no spam (real)	39	525

Tabla N° 13: Matriz de confusión del Random Forest (Manual) con el conjunto de la Tabla N° 3

Con atributo de clase *cluster* colocado automáticamente con *addCluster*

En este caso se usó el mismo conjunto de entrenamiento de la sección anterior. Sin embargo, en lugar de usar la clasificación manual como etiqueta, se usó la asignación en dos posibles *clusters* dados por el filtro *addCluster*. En este caso, los resultados son mucho mejores que los anteriores con *cross-validation*.

Algoritmo	Accuracy	Recall	Specificity	Precision	NPV	AUCROC
Naive Bayes	93.5714 %	0,970	0,833	0,946	0,901	0,990
AdaBoost	98.4286 %	1,000	0,937	0,980	1,000	1,000
KNN	99.4286 %	1,000	0,977	0,992	1,000	0,989
RandomForest	100%	1,0	1,000	1,0	1,000	1,0

Tabla N° 14: Para la clase sana (automático) con el conjunto de la Tabla N° 3 y cross-validation

Algoritmo	Accuracy	Recall	Specificity	Precision	NPV	AUCROC
Naive Bayes	93.5714 %	0,833	0,970	0,901	0,946	0,990
AdaBoost	98.4286 %	0,937	1,000	1,000	0,980	1,000
KNN	99.4286 %	0,977	1,000	1,000	0,992	0,983
RandomForest	100%	1,0	1,000	1,000	1,000	1,0

Tabla N° 15: Para la clase spam (automático) con el conjunto de la Tabla N° 3 y cross-validation

	spam (clasificado)	no spam (clasificado)
spam (real)	145	29
no spam (real)	16	510

Tabla N° 16: Matriz de confusión del Naive Bayes (automático) con el conjunto de la Tabla N° 3

	spam (clasificado)	no spam (clasificado)
spam (real)	163	11
no spam (real)	0	526

Tabla N° 17: Matriz de confusión del AdaBoost (automático) con el conjunto de la Tabla N° 3

	spam (clasificado)	no spam (clasificado)
spam (real)	170	4
no spam (real)	0	526

Tabla N° 18: Matriz de confusión del KNN (automático) con el conjunto de la Tabla N° 3

	spam (clasificado)	no spam (clasificado)
spam (real)	174	0
no spam (real)	0	526

Tabla N° 19: Matriz de confusión del Random Forest (Automático) con el conjunto de la Tabla N° 3

Pruebas de Validación con el conjunto preparado con *addCluster*

En este caso, se repite la evaluación anterior con el conjunto de entrenamiento cuyas clases fueron etiquetadas con el filtro *addCluster* de Weka. Sin embargo, en lugar de usar *cross-validation*, se usó el conjunto de validación que ya se tenía para hacer una prueba independiente por cada algoritmo. Este conjunto de validación fue procesado de manera independiente y luego se usó en él *addCluster* para obtener de manera automática un etiquetamiento. Resulta evidente que los resultados no son aceptables, aunque resultan mejores que los obtenidos con la clasificación manual.

Es importante destacar el orden de los pasos seguidos en esta sección con el fin de compararla con la siguiente. De los conjuntos de datos clasificados manualmente --el conjunto de entrenamiento y el conjunto de validación--, se desechó su etiquetamiento y procesaron luego de manera independiente.

Algoritmo	Accuracy	Recall	Specificity	Precision	NPV	AUCROC
Naive Bayes	55.2846 %	0,764	0,123	0,640	0,204	0,310
AdaBoost	53.252 %	0,745	0,099	0,628	0,160	0,361
KNN	52.439 %	0,721	0,123	0,626	0,179	0,426
RandomForest	52.8455 %	0,624	0,333	0,656	0,303	0,644

Tabla N° 20: Para la clase sana (Automático) usando el conjunto de la tabla N° 3 como entrenamiento y el conjunto de la tabla N°4 como validación

Algoritmo	Accuracy	Recall	Specificity	Precision	NPV	AUCROC
Naive Bayes	55.2846 %	0,123	0,764	0,204	0,640	0,310
AdaBoost	53.252 %	0,099	0,745	0,160	0,628	0,361
KNN	52.439 %	0,123	0,721	0,179	0,626	0,426

RandomForest	52.8455 %	0,333	0,624	0,303	0,656	0,644

Tabla N° 21: Para la clase spam (Automático) usando el conjunto de la tabla N° 3 como entrenamiento y el conjunto de la tabla N°4 como validación

	spam (clasificado)	no spam (clasificado)
spam (real)	10	71
no spam (real)	39	126

Tabla N° 22: Matriz de confusión del Naive Bayes (automático) usando el conjunto de la tabla N° 3 como entrenamiento y el conjunto de la tabla N°4 como validación

	spam (clasificado)	no spam (clasificado)
spam (real)	8	73
no spam (real)	42	123

Tabla N° 23: Matriz de confusión del AdaBoost (automático) usando el conjunto de la tabla N° 3 como entrenamiento y el conjunto de la tabla N°4 como validación

	spam (clasificado)	no spam (clasificado)
spam (real)	10	71
no spam (real)	46	119

Tabla N° 24: Matriz de confusión del KNN (automático) usando el conjunto de la tabla N° 3 como entrenamiento y el conjunto de la tabla N°4 como validación

	spam (clasificado)	no spam (clasificado)

spam (real)	27	54
no spam (real)	62	103

Tabla N° 25: Matriz de confusión del Random Forest (Automático) usando el conjunto de la tabla N° 3 como entrenamiento y el conjunto de la tabla N°4 como validación

Pruebas de Validación usando el conjunto de datos Original preparado con *addCluster*

A diferencia de la sección anterior, aquí se tomaron los 3000 vectores de atributos originales sin clasificar manualmente. Anteriormente se había seleccionado una muestra de casi 1000 de estos que habían sido clasificados manualmente, y luego fueron divididos en conjuntos de entrenamiento y de validación. Sin embargo, en este caso no se tomó ninguna muestra aleatoria, debido a que se ha usado la totalidad de los tuits originales.

Posteriormente, a este conjunto completo de datos estructurados se le aplicaron filtros de Weka como *normalize*, *numericToNominal, Randomize* y *addCluster*. Luego de esto, se entrenó cada algoritmo seleccionado con una muestra de 70% de los datos, dejándose el restante 30% para validar.

Es importante resaltar que a diferencia de la sección anterior, aquí primero se aplica *addCluster* y luego se divide el conjunto de datos.

Algoritmo	Accuracy	Recall	Specificity	Precision	NPV	AUCROC
Naive Bayes	96.6667 %	0,985	0,941	0,959	0,978	0,994
AdaBoost	96.5556 %	0,966	0,965	0,975	0,952	0,994
KNN	98.3333 %	0,983	0,984	0,989	0,976	0,983
RandomForest	98.2222 %	0,987	0,976	0,983	0,981	0,999

Tabla N° 26: Para la clase sana (Automático) con el conjunto de la gráfica N° 18 usando 70% para entrenamiento y 30% para validación, sin los atributos: retuits y favoritos

Algoritmo	Accuracy	Recall	Specificity	Precision	NPV	AUCROC
Naive	96.6667 %	0,941	0,985	0,978	0,959	0,994

Bayes						
AdaBoost	96.5556 %	0,965	0,966	0,952	0,975	0,994
KNN	98.3333 %	0,984	0,983	0,976	0,989	0,983
RandomForest	98.2222 %	0,976	0,987	0,981	0,983	0,999

Tabla N° 27: Para la clase spam (Automático) con el conjunto de la gráfica N° 18 usando 70% para entrenamiento y 30% para validación, sin los atributos: retuits y favoritos

	spam (clasificado)	no spam (clasificado)
spam (real)	351	22
no spam (real)	8	519

Tabla N° 28: Matriz de confusión del Naive Bayes (automático) con el conjunto de la gráfica N° 18 usando 70% para entrenamiento y 30% para validación, sin los atributos: retuits y favoritos

	spam (clasificado)	no spam (clasificado)
spam (real)	360	13
no spam (real)	18	509

Tabla N° 29: Matriz de confusión del AdaBoost (automático) con el conjunto de la gráfica N° 18 usando 70% para entrenamiento y 30% para validación, sin los atributos: retuits y favoritos

	spam (clasificado)	no spam (clasificado)
spam (real)	367	6
no spam (real)	9	518

Tabla N° 30 Matriz de confusión del KNN (automático) con el conjunto de la gráfica N° 18 usando 70% para entrenamiento y 30% para validación, sin los atributos: retuits y favoritos

	spam (clasificado)	no spam (clasificado)
spam (real)	364	9

no spam (real)	7	520

Tabla N° 31: Matriz de confusión del Random Forest (Automático) con el conjunto de la gráfica N° 18 usando 70% para entrenamiento y 30% para validación, sin los atributos: retuits y favoritos

Pruebas de Validación usando el conjunto de datos original preparado con *addCluster* con dos atributos adicionales

A pesar de los buenos resultados que se obtuvieron cuando se usó la asignación generada por el agrupamiento, resulta poco evidente que los *clusters* generados sean útiles para la detección de *Spam*. Por esta razón, en esta sección se repite el trabajo de la anterior. No obstante, aquí se usa un conjunto de datos estructurados con dos atributos adicionales: el número de retuits y el número de favoritos.

Algoritmo	Accuracy	Recall	Specificity	Precision	NPV	AUCROC
Naive Bayes	99%	1,0	0,979	0,982	1,000	1,000
AdaBoost	95.4444 %	0,950	0,956	0,964	0,944	0,990
KNN	98%	0,985	0,974	0,977	0,983	0,978
RandomForest	93.8889 %	0,996	0,874	0,900	0,995	0,994

Tabla N° 32: Para la clase sana (Automático) con el conjunto de la gráfica N° 18 usando 70% para entrenamiento y 30% para validación

Algoritmo	Accuracy	Recall	Specificity	Precision	NPV	AUCROC
Naive Bayes	99%	0,979	1,000	1,000	0,982	1,000
AdaBoost	95.4444 %	0,960	0,950	0,944	0,964	0,990
KNN	98%	0,974	0,985	0,983	0,977	0,978
RandomForest	93.8889 %	0,874	0,996	0,995	0,900	0,994

Tabla N° 33: Para la clase spam (Automático) con el conjunto de la gráfica N° 18 usando 70% para entrenamiento y 30% para validación

	spam (clasificado)	no spam (clasificado)
spam (real)	411	9
no spam (real)	0	480

Tabla N° 34: Matriz de confusión del Naive Bayes (automático) con el conjunto de la gráfica N° 18 usando 70% para entrenamiento y 30% para validación

	spam (clasificado)	no spam (clasificado)
spam (real)	403	17
no spam (real)	24	456

Tabla N° 35: Matriz de confusión del AdaBoost (automático) con el conjunto de la gráfica N° 18 usando 70% para entrenamiento y 30% para validación

	spam (clasificado)	no spam (clasificado)
spam (real)	409	11
no spam (real)	7	473

Tabla N° 36: Matriz de confusión del KNN (automático) con el conjunto de la gráfica N° 18 usando 70% para entrenamiento y 30% para validación

	spam (clasificado)	no spam (clasificado)
spam (real)	367	53
no spam (real)	2	478

Tabla N° 37: Matriz de confusión del Random Forest (Automático) con el conjunto de la gráfica N° 18 usando 70% para entrenamiento y 30% para validación

Evaluación con el conjunto etiquetado manualmente con el atributo adicional del tópico

En este caso, se usó el conjunto de datos etiquetado manualmente y se evaluó con *cross-validation*. Sin embargo, se usó como atributo adicional el identificador del tópico al que pertenece cada tuit.

Algoritmo	Accuracy	Recall	Specificity	Precision	NPV	AUCROC
Naive Bayes	79.1429 %	0,917	0,272	0,839	0,440	0,732
AdaBoost	81.2857 %	0,991	0,074	0,816	0,667	0,731
KNN	79.7143 %	0,871	0,493	0,877	0,479	0,692
RandomForest	84.1429 %	0,934	0,456	0,877	0,626	0,786

Tabla N° 38: Para la clase sana (Manual) usando el conjunto de la tabla N° 3 con el identificador de tópicos como atributo adicional y cross-validation

Algoritmo	Accuracy	Recall	Specificity	Precision	NPV	AUCROC
Naive Bayes	79.1429 %	0,272	0,917	0,440	0,839	0,732
AdaBoost	81.2857 %	0,074	0,991	0,667	0,816	0,731
KNN	79.7143 %	0,493	0,871	0,479	0,877	0,692
RandomForest	84.1429 %	0,456	0,934	0,626	0,877	0,786

Tabla N° 39: Para la clase spam (Manual) usando el conjunto de la tabla N° 3 con el identificador de tópicos como atributo adicional y cross-validation

	spam (clasificado)	no spam (clasificado)
spam (real)	37	99

no spam (real)	47	517

Tabla N° 40: Matriz de confusión del Naive Bayes (Manual) usando el conjunto de la tabla N° 3 con el identificador de tópicos como atributo adicional y cross-validation

	spam (clasificado)	no spam (clasificado)
spam (real)	10	126
no spam (real)	5	559

Tabla N° 41: Matriz de confusión del AdaBoost (Manual) usando el conjunto de la tabla N° 3 con el identificador de tópicos como atributo adicional y cross-validation

	spam (clasificado)	no spam (clasificado)
spam (real)	67	69
no spam (real)	73	491

Tabla N° 42: Matriz de confusión del KNN (Manual) usando el conjunto de la tabla N° 3 con el identificador de tópicos como atributo adicional y cross-validation

	spam (clasificado)	no spam (clasificado)
spam (real)	62	74
no spam (real)	37	527

Tabla N° 43: Matriz de confusión del Random Forest (Manual) usando el conjunto de la tabla N° 3 con el identificador de tópicos como atributo adicional y cross-validation

Evaluación con el conjunto etiquetado manualmente con atributos adicionales sobre tópico, número de retuits y número de favoritos

En este caso, se usó el conjunto de datos etiquetado manualmente y se evaluó con *cross-validation*. Sin embargo, se usaron como atributos adicionales el identificador del tópico, el número de retuits y el número de veces que cada tuit fue marcado como favorito.

Algoritmo	Accuracy	Recall	Specificity	Precision	NPV	AUCROC
Naive Bayes	51.2857 %	0,431	0,853	0,924	0,265	0,726
AdaBoost	82%	0,963	0,228	0,838	0,596	0,760
KNN	85.8571 %	0,952	0,478	0,882	0,468	0,796
RandomForest	79.2857 %	0,869	0,471	0,873	0,703	0,673

Tabla N° 44: Para la clase sana (Manual) usando el conjunto de la tabla N° 3 con el identificador de tópicos, retuits y favoritos como atributos adicionales y cross-validation

Algoritmo	Accuracy	Recall	Specificity	Precision	NPV	AUCROC
Naive Bayes	51.2857 %	0,853	0,431	0,265	0,924	0,726
AdaBoost	82%	0,228	0,963	0,596	0,838	0,760
KNN	85.8571 %	0,471	0,869	0,703	0,873	0,796
RandomForest	79.2857 %	0,478	0,952	0,468	0,882	0,673

Tabla N° 45: Para la clase spam (Manual)usando el conjunto de la tabla N° 3 con el identificador de tópicos, retuits y favoritos como atributos adicionales y cross-validation

	spam (clasificado)	no spam (clasificado)
spam (real)	116	20
no spam (real)	321	243

Tabla N° 46: Matriz de confusión del Naive Bayes (Manual)usando el conjunto de la tabla N° 3 con el identificador de tópicos, retuits y favoritos como atributos adicionales y cross-validation

	spam (clasificado)	no spam (clasificado)
spam (real)	31	105
no spam (real)	21	543

Tabla N° 47: Matriz de confusión del AdaBoost (Manual)usando el conjunto de la tabla N° 3 con el identificador de tópicos, retuits y favoritos como atributos adicionales y cross-validation

	spam (clasificado)	no spam (clasificado)
spam (real)	65	71
no spam (real)	74	490

Tabla N° 48: Matriz de confusión del KNN (Manual)usando el conjunto de la tabla N° 3 con el identificador de tópicos, retuits y favoritos como atributos adicionales y cross-validation

	spam (clasificado)	no spam (clasificado)
spam (real)	64	72
no spam (real)	27	537

Tabla N° 49: Matriz de confusión del Random Forest (Manual)usando el conjunto de la tabla N° 3 con el identificador de tópicos, retuits y favoritos como atributos adicionales y cross-validation

Desarrollo de una Aplicación Web

A manera de demostración, se desarrolló una sencilla aplicación web que le permite al usuario introducir una cuenta de Twitter. De ella se solicitan los diez últimos tuits para su clasificación en *spam* o *sano*. Si el tuit aparece de color rojo, entonces es *spam*. De lo contrario, es *sano*. Adicionalmente, al lado de cada tuit aparece la probabilidad con la que fue asignada su clasificación.

Esta aplicación usa en el lado del servidor el modelo obtenido con el algoritmo Naive Bayes aplicado al conjunto de entrenamiento con todos los atributos, incluidos el tópico, número de retuits y número de favoritos. Dicho algoritmo obtuvo el mejor *recall* de los algoritmos aplicados al conjunto etiquetado manualmente, tal como se puede ver en la tabla N° 45. Sin embargo, también se pueden usar fácilmente los modelos de todos los algoritmos entrenados con el conjunto de entrenamiento original etiquetado automáticamente y con todos los atributos. Ambos tipos de modelos arrojan resultados muy distintos.

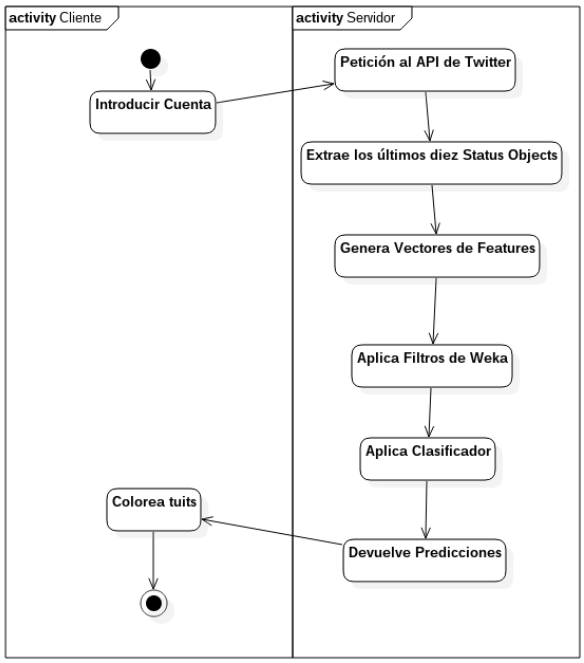


Diagrama de Actividades de la Aplicación Web

Conclusiones

El proyecto desarrollado para esta investigación buscó elaborar un sistema capaz de detectar *Spam* o noticias falsas en cuentas de Twitter de cadenas de noticias. Sin embargo, los resultados vistos con la clasificación manual (Tabla N° 9) demuestran que los criterios usados para esta investigación, tomados de los autores [1], [2] y [3], no son fácilmente aplicables en este ámbito tan específico. De igual manera, estos resultados podrían ser el reflejo del error humano al llevar a cabo la clasificación manual.

Para evitar este problema, se probaron con varias configuraciones de los modelos con el fin de encontrar los mejores candidatos. Una de ellas fue dirigida a evitar los inconvenientes del etiquetamiento manual. Por esta razón, se usó el filtro *addCluster* de Weka para generar un etiquetamiento automático. Tal como se puede apreciar en las tablas 15, 21, 27 y 33, esto, en general, arroja mejores resultados que los modelos que son productos del etiquetamiento manual, sin importar el régimen de pruebas. Sin embargo, el problema de este enfoque radica en que los algoritmos de agrupamiento usados para generar el etiquetamiento detecten clases útiles desde el punto de vista de este proyecto. En este sentido, la gráfica N° 16 demuestra cómo las nociones de *Spam* usadas en este proyecto difieren del etiquetamiento automático. En esta gráfica, sólo basta comparar a la clase *cluster* con la clase *Spam* para notar esto.

Debido a los inconvenientes anteriores, fue menester incluir más atributos en el conjunto de datos etiquetado manualmente. Entre los atributos adicionales se encuentran el número de retuits, el número de favoritos que ha recibido el tuit y su tópico. En el caso de la tabla N° 39, se pueden ver los resultados al agregar el atributo de tópico. En la tabla N° 45, se pueden ver los resultados al agregar los tres atributos. Si bien en el caso de la N° 39 hay una mejoría importante si se le compara con la tabla N° 9, estos resultados siguen siendo inaceptables. En el caso de la tabla N° 45, se obtuvo un resultado sorpresivo con el Naive Bayes. Si bien su *accuracy* es bastante bajo, su *recall* es 0,853. Lo que significa que dicho modelo podría ser usado para la detección de *Spam,* bajo una política agresiva y dispuesta a aceptar los falsos positivos.

Finalmente, en la aplicación desarrollada, se usa el antedicho modelo del Naive Bayes con *recall* de 0.853, así como los modelos de etiquetamiento automático que toman en cuenta todos los atributos. Como era de esperarse, los resultados de las pruebas de dicha aplicación (que no se detallan en el presente informe) son reveladores de una gran disparidad. Para trabajos futuros, se podría implementar un sistema que le permita al usuario de dicha aplicación corregir sus predicciones y usar este mecanismo para mejorar el aprendizaje de sus modelos. Con respecto al proyecto, se podría usar un conjunto de datos más variado, pues sólo se usaron los tuits de tres cuentas de Twitter de cadenas de noticias. Además se podrían considerar más atributos para los datos estructurados.

Referencias

[1] Meet Rajdev (2015). Fake and Spam Messages: Detecting Misinformation During Natural Disasters on Social Media. All Graduate Theses and Dissertations. Paper 4462.

[2] M. McCord, M.Chuah (2011). Spam Detection on Twitter Using Traditional Classifiers, Proceedings of the 8th international conference on Autonomic and trusted computing, 175-186.

[3] Gordon Edwards (2015). Spam Detection on Twitter, Proceedings of the 4th Year Project Report from the School of Informatics of the University of Edinburgh.

[4] CaraotaDigital (@CaraotaDigital) | Twitter. Twittercom. 2017. Available at: https://twitter.com/CaraotaDigital. Accessed March 13, 2017.

[5] Twitter developer documentation | Twitter. Twittercom..2017. Available at: https://dev.twitter.com/overview/api/tweets. Accessed March 13, 2017.

[6] CNN en Español (@CNNEE) | Twitter. Twittercom. 2017. Available at: https://twitter.com/CNNEE. Accessed March 13, 2017.

[7] La Patilla (@la_patilla) on Twitter. Twittercom. 2017. Available at: https://twitter.com/la_patilla. Accessed March 13, 2017.

[8] michenriksen/birdwatcher. GitHub. 2017. Available at: https://github.com/michenriksen/birdwatcher. Accessed March 13, 2017.

[9] michenriksen/birdwatcher. GitHub. 2017. Available at: https://github.com/michenriksen/birdwatcher#getting-schema-information. Accessed March 13, 2017.

[10] Topic modeling in Python — Text Analysis with Topic Models for the Humanities and Social Sciences. Dedariaheu. 2017. Available at: https://de.dariah.eu/tatom/topic_model_python.html. Accessed March 13, 2017.

[11] Marshland, S. (2015). Machine Learning (2nd ed.). New York: Boca Raton.