UNIVERSIDAD AUTONOMA METROPOLITANA

UNIDAD AZCAPOTZALCO
División de Ciencias Básicas e Ingeniería
Licenciatura en Ingeniería en Computación

ATRIBUCION DE AUTORIAS DE TEXTOS CIENTIFICOS EN ESPAÑOL

"Proyecto Tecnológico"

Justino Ramírez Reyes.
Matrícula: 202305571
ramirezjt@gmail.com.mx

Asesor:
Oscar Herrera Alcántara
Profesor Asociado
Departamento de Sistemas
oha@correo.azc.uam.mx

Co asesor:
José Alejandro Reyes Ortiz
Profesor Titular
Departamento de sistemas
jaro@correo.azc.uam.mx

Trimestre 16 Invierno
Fecha de entrega:
13 de abril de 2016

DECLARATORIA

Yo, Oscar Herrera Alcántara, declaro que aprobé el contenido del presente Reporte de Proyecto de Investigación y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Asesor

Dr. Oscar Herrera Alcántara
Profesor Asociado
oha@correo.azc.uam.mx

Yo, José Alejandro Reyes Ortíz, declaro que aprobé el contenido del presente Reporte de Proyecto de Investigación y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Co-asesor

Dr. José Alejandro Reyes Ortíz
Profesor Titular
jaro@correo.azc.uam.mx

Yo, Justino Ramírez Reyes, doy mi autorización a la Coordinación de Servicios de Información de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, para publicar el presente documento en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Alumno

Justino Ramírez Reyes
Matrícula. 202305571
ramirezjt@gmail.com

Resumen

Uno de los problemas a los que se enfrentan los analistas de noticias, notas periodísticas o notas científicas, es tener que identificar, agrupar u ordenar la información por el autor que las ha escrito. Esto conlleva a invertir un esfuerzo adicional a esta tarea, la cual es tediosa, consume bastante tiempo y representa un costo para ellos.

Este proyecto propone la identificación automática del autor de un texto, ya sea científico, académico o periodístico, utilizando algoritmos de clasificación automática de textos. La idea principal es contar con un conjunto de textos etiquetados para dos o más autores, y el enfoque propuesto es capaz de predecir el autor de un texto no etiquetado, es decir, que no tiene un autor asociado.

Esto puede ser de gran utilidad para la identificación del autor de textos anónimos o simplemente, para agrupar noticias del mismo editor.

Como parte de la evaluación se han probado tres algoritmos, a saber: Bayes, J48 y Tablas de Decisión, de los cuales el mejor resultado se obtuvo con el algoritmo Bayes mostrando un 94% de asignación correcta de autorías a los textos.

CONTENIDO **PAG.**

1. INTRODUCCIÓN

La Atribución de Autoría en Textos Científicos en Español [AACTE] es una actividad que busca identificar el estilo de escritura de autores, a fin de asignar de manera automática textos de autoría desconocida al autor correspondiente. Es decir, pueda reconocer características textuales que al compararlas, permitan seleccionar entre documentos escritos por distintos autores. Los métodos habituales de AACTE extraen marcadores de estilo, que se consideran atributos de los textos y se utiliza para entrenar clasificadores. Estas marcas de estilo, incluyen: frecuencia de caracteres, palabras, frase, n-gramas a nivel carácter, combinaciones de palabras o n-gramas de palabras, por citar algunos.

Es importante señalar que la AACTE no debe ser abordada de forma temática, toda vez que las características textuales más importantes no son de tipo temático, el objetivo es modelar el tipo de escritura de cada autor con el fin de diferenciarlos, incluso en el mismo contexto. Hoy en día, la cantidad de información disponible es gigantesca y gran parte de ella está en texto plano [correos electrónicos, blogs, foros en línea, etc.]. En este sentido, han surgido diversos temas que involucran a la AACTE, por ejemplo: detección de plagio, correo no deseado, informática forense, detección de fraude, autenticidad de documentos.

En el presente proyecto se propone un método para combinar características léxicas y sintácticas empleando una representación novedosa conocida como representación holográfica reducida 'RHR', propuesta por Plate[1] como mecanismo para representar estructuras complejas y jerárquicas que no se limitan al lenguaje, pues este tipo de estructuras se encuentran en otras áreas como el análisis de imágenes. Por otra parte cuando los documentos se representan utilizando la aproximación de bolsa de palabras, sabemos que la dimensión vectorial es igual al tamaño del vocabulario de la colección. Para optimizar el procesamiento existen métodos que buscan reducir dicha dimensión, uno de los más utilizados en recuperación de información es la indexación semántica latente. Sin embargo éste método hace uso de la descomposición en valores singulares que es un proceso computacionalmente costoso. Como alternativa Sahlgren[2] proponen un método conocido como Random Indexing RI [indexación aleatoria]. El Presente proyecto utiliza la técnica de RI para reducir la dimensión vectorial y optimizar el tiempo de

procesamiento. La principal aportación de la presente investigación es la aplicación del método RHR a la actividad de AACTE, donde no ha sido aprovechado, de acuerdo con la información disponible a la fecha.

2. ANTECEDENTES

La gran cantidad de textos en español presente de forma anónima o bajo seudónimo ha provocado, desde hace siglos, infinidad de propuestas de autoría para los mismos.

Sirva como ejemplo lo ocurrido con El Lazarillo de Tormes, obra que, desde que en 1605 José de Sigüenza se la atribuyese a fray Juan de Ortega, ha conocido –con mayor o menor éxito– muy diferentes propuestas de paternidad (bien es verdad que con diferente fortuna y con distinto rigor filológico). Algo parecido ha ocurrido con el Quijote de Avellaneda y, en fin, con muchas otras obras de interés relevante en el canon español.

Las propuestas de autoría, impulsadas por el comprensible deseo del investigador de alzarse con el dudoso honor de llegar a unir su nombre con la resolución de un misterio, han dado aliento a estudios no siempre compensadores de los esfuerzos invertidos por el investigador ni por los que éste, al hacer público su trabajo, demanda de los lectores.

Aunque los estudios de atribución son tan viejos como la literatura misma, sólo desde hace algunas décadas se ha realizado un esfuerzo digno de elogio por incorporar las fórmulas y los procedimientos que, desde la década de los 60, se han ido consolidando en el terreno de la Lingüística forense, sobre todo en el ámbito de la filología anglosajona; procedimientos que, desde luego, han supuesto una apuesta interesante de cara a la construcción de una metodología fiable (por su capacidad de objetivación de los fenómenos observados) y al establecimiento de unos protocolos de actuación en los que se redujese notablemente el espacio concedido a la subjetividad o a la elección arbitraria por parte del analista de los fenómenos considerados para hacer atribución.

En esencia, los fundamentos teóricos de los estudios de atribución a los que nos estamos refiriendo resultan bastante claros y de difícil contestación: un autor, al escribir, produce un discurso que indefectiblemente —salvo manipulaciones específicamente concebidas para ocultar ciertas huellas— presenta toda una serie de marcas verbales, patrones de escritura, que lo caracterizan e identifican frente a otros discursos salidos de diferente mano.

El convencimiento de que cada hablante/escritor posee un idiolecto propio y exclusivo —esto es, unos hábitos personales que se escapan a lo consciente—, es lo que ha propiciado que de un tiempo a esta parte se trabaje en el establecimiento de unos algoritmos de atribución fiables que, ante un texto anónimo o sólo respaldado por un seudónimo, permitan proponer una autoría con cierta seguridad.

Por lo general, los recientes estudios de atribución de autoría en el ámbito del habla en español, parten de la convicción anterior y son bastantes ya los trabajos que, en el mejor de los casos, pretenden aplicar alguno de los muchos protocolos metodológicos desarrollados desde la lingüística forense anglosajona para determinar la autoría de un texto anónimo o trasmitido bajo seudónimo. Sin embargo, todavía se observa en ellos ciertos vicios de método que, groso modo, pueden

Antes de poder poner en práctica cualquier técnica de atribución se impone la necesidad de conocer con precisión el grado de seguridad y de fiabilidad que ofrecen los métodos y procedimientos de medición textual con los que abordar en ellos el tema de la autoría. Y, junto a todo lo anterior, debe seguirse reclamando un profundo conocimiento filológico del texto en cuestión, tanto de sus peculiaridades histórico-literarias como de textualidad, tanto en lo que se refiere a su gestación como a su transmisión.

Es de la constatación de la necesidad de contar con métodos fiables de la que arranca nuestro trabajo, que no tiene otra pretensión que la de someter a juicio algoritmos atributivos utilizados hasta la fecha, con el fin de evaluar cuantitativamente su grado de confianza.

3. OBJETIVO

Implementar un método computacional para la asignación de autorías en textos científicos en español basado en el contenido de los documentos

3.1 OBJETIVO PARTICULAR

- Implementar un módulo para extraer la información relevante de las publicaciones científicas a partir del modelo de datos.

- Implementar un módulo para la comparación de la información relevante basada en la sintaxis y la semántica del contenido.

- Implementar un módulo para descubrir autorías y representarlas en un modelo de datos**.**

4. JUSTIFICACION

Hoy en día, la cantidad de información disponible en internet es ilimitada, y gran parte de la misma se encuentra en texto plano [correos electrónicos, blogs, foros en línea, sitios web, etc.]. En este sentido, han surgido diversos tópicos que involucran a la atribución de autoría como herramienta de ayuda a fin de facilitar la detección de plagios, correos electrónicos no deseados, informática forense, detección de fraudes, autenticidad de documentos, entre otros ejemplos.

La AACTE motivo del presente proyecto, sirve como instrumento de apoyo para el reconocimiento de obras concebidas por el autor o el licenciante, asimismo favorece a evitar el plagio, usurpación de autoría, la ilegitimidad y el fraude, fortaleciendo la propiedad del creador y de los derechos que posee en el caso de que sus contenidos sean susceptibles de copiarse, distribuirse, exhibirse y hasta en las facultades que puede tener sobre obras derivadas a partir de las suyas.

Asimismo, la tecnología que nos permite crear, publicar y acceder a la información parece ir continuamente por delante de las leyes que no consiguen estar a la altura de la situación, aunque se han producido algunos avances significativos. La historia de los derechos de autor es una constante adaptación según se van produciendo los vertiginosos cambios comerciales y técnicos. Las Tecnologías de la Información y Comunicación "TICs", crean continuamente nuevos retos y la ley trata de ir respondiéndolos de manera paralela, pero esto no significa que el uso y gestión de la tecnología no esté regulada, o está más allá de la ley. En cualquier caso, todos los aspectos de los derechos de autor son igualmente aplicables a las obras y materiales dispuestos en Internet, que pueden ser de uso privado, shareware, freeware, etc. Algunos tienen licencias de uso donde se declara qué se puede hacer con ese material según lo decidido por el propietario de los derechos. En ningún caso se puede presumir que si una obra está accesible en Internet es que carece de derechos de autor, independientemente de la facilidad con la que podamos acceder a ella, copiarla, modificarla o distribuirla

En este contexto, el presente proyecto contribuye en la regulación de la reproducción, distribución, comunicación y transformación de autorías en textos científicos en español, toda vez que los asociará con su autor original, según sus correspondientes reglas inherentes de implementación.

5. MARCO TEORICO

5.1 Establecimiento de los algoritmos de atribución:

Los algoritmos de atribución en que basaremos nuestro estudio giran en torno contenidos textuales, cuya funcionalidad para la medición textual con intención atributiva cuenta ya con una bibliografía interesante, las marcas de puntuación, las palabras, los grafemas, la frecuencia de *n-gramas* y la riqueza de vocabulario.

***5.1.1 Frecuencia de marcas de puntuación*.-** En el trabajo de Grieve se contemplan cinco mediciones de frecuencia de puntuación: las tres primeras son variantes del *perfil de marca de puntuación simple (simple punctuation mark profile)*, según se calcule la frecuencia relativa de una serie de signos de puntuación en función del número total de caracteres, signos o palabras

que contiene un determinado texto, y las otras dos son combinaciones del perfil de puntuación con los perfiles de grafema (*punctuation and grapheme profile*) y de palabra (*punctuation and word profile*):

a) El *perfil de marca de puntuación simple 1* se calcula dividiendo la frecuencia de una serie de signos de puntuación entre el número total de caracteres contenidos en el texto.

b) El *perfil de marca de puntuación simple 2* se obtiene dividiendo la frecuencia de cada uno de los signos de puntuación seleccionados entre el cómputo total de los que se contabilizan en el texto.

c) El *perfil de marca de puntuación simple 3* se calcula dividiendo la frecuencia de las marcas de puntuación escogidas entre el número total de palabras del texto.

d) El *perfil de puntuación y palabra* contempla y mide la frecuencia del conjunto formado por un signo de puntuación dado y la palabra que lo precede o lo sigue.

e) El *perfil de puntuación y grafema* mide la frecuencia del conjunto formado por un signo de puntuación dado y el grafema que lo precede o lo sigue.

Para nuestro test hemos optado por el *perfil de marca de puntuación simple1* al ser esta medición la que ofrece mayores garantías en la lengua de Shakespeare, pues mientras las variantes 2 y 3 ofrecen un 53 y un 57%, respectivamente, para discriminar entre diez autores, la variante 1 llega al 58%; por otro lado, hemos seleccionado la combinatoria del *perfil de puntuación y palabra* por ser la medición más discriminante de todas las analizadas por Grieve, llegando a alcanzar el 95% a la hora de discernir entre dos autores.

***5.1.2 Frecuencia de palabra*.-** Grieve somete a prueba tres mediciones diferentes de frecuencia de palabra:

a) El *perfil de palabra simple* (*simple word profile*) se define como la frecuencia relativa de una serie de términos de alta frecuencia y se calcula dividiendo la frecuencia de una determinada palabra entre el número total de palabras del texto. El *perfil de palabra en posición sencilla* (*single-position word profile*) mide la frecuencia relativa de una serie de

palabras que aparecen en una posición particular dentro de las oraciones de un texto (por ejemplo, primera palabra, segunda palabra… o última palabra dentro de la oración), y se obtiene de dividir la frecuencia de una palabra en una posición seleccionada por el número de oraciones que contienen dicha posición.c)

b) El *perfil de palabra en multiposición* (*multi-position word profile*) recoge las mediciones de diversas palabras en posición individual múltiple (por ejemplo, las primeras cuatro palabras de una oración).

De las tres mediciones de frecuencia de palabra que describe Grieve en su artículo, únicamente hemos contemplado en nuestro estudio el *perfil de palabra simple* por ser la que en la lengua española ofrece una mayor efectividad.

5.1.3 Frecuencia de grafema.- Grieve prueba cuatro tipos diferentes de frecuencias de grafemas:

a) El *perfil de grafema simple* consiste en la frecuencia relativa del total de grafemas de la lengua a analizar (26 en el caso del alfabeto inglés), que, al igual que en el caso del perfil de palabra simple, se calcula dividiendo la frecuencia de ese grafema en el texto por el número total de grafemas. El *perfil de grafema en posición sencilla* hace referencia a la frecuencia relativa de los grafemas que aparecen en una posición particular dentro de las palabras de un texto (por ejemplo, primer grafema, segundo grafema o último grafema de la palabra) y se calcula dividiendo la frecuencia que tiene cada grafema en esa posición entre el número total de palabras que contienen esa posición.

b) El *perfil de grafemas en multiposición* debe entenderse como el perfil de grafemas en posición individual múltiple (por ejemplo, los tres primeros grafemas de una palabra)El *perfil de grafema en interior de palabra*, que consiste en el porcentaje de palabras de un texto que contiene cada uno de los grafemas de la lengua a estudiar, y se calcula dividiendo el número de palabras en un texto que contienen al menos un caso de ese grafema por el número total de palabras. *Frecuencia de colocación, tanto a nivel de palabra como de grafema* (*N-gramas*).- La medición *n-gramas* recoge la frecuencia

relativa de una determinada cadena de *n* grafemas o de *n* palabras, dividiendo la frecuencia de aparición de esa secuencia concreta de *n-gramas* a analizar entre el número total de secuencias existentes de esos *n-gramas*. Por ejemplo: 2-gramas de *es* = Frecuencia de la secuencia *es* / Total de secuencias de 2 caracteres.

***5.1.4 Riqueza de vocabulario*.-** De los 11 tipos de mediciones que ofrece Grieve, nosotros sólo hemos contemplado la *ratio type/token* y las mediciones ofrecidas por las fórmulas de Honoré y Yule. A pesar de que los algoritmos referentes a la riqueza de vocabulario constituyen algunas de las medidas menos efectivas en el caso del inglés, principalmente cuando hay que discriminar entre 5 o más autores, en nuestro deseo por confrontar y comparar los resultados porcentuales de los algoritmos de atribución en las dos lenguas, consideramos tan interesante ver si se mantienen los altos porcentajes como si lo hacen los bajos.

En definitiva, las variables contempladas, bien por ser las más operativas o las más ineficaces en la lengua española, son las que señalamos a continuación:

▪ *Perfil de marca de puntuación simple 1*.- Hemos determinado, a través del programa de análisis textual *WordSmith Tools*, las frecuencias relativas de cuatro marcas de puntuación —punto (.), coma (,), punto y coma (;) y dos puntos (:)—, dividiendo el número de veces que aparece cada una de ellas entre el número total de caracteres del texto.

▪ *Perfil de puntuación y palabra (puntuación + palabra)*.- Para calcular esta medición, que combina el perfil de puntuación con el de palabra, hemos optado por unificar cualquier signo de puntuación —a pesar de que presuponemos más rentable el análisis distintivo de cada marca— por el hecho de que, en un texto del Siglo de Oro (y en última instancia este trabajo arranca de nuestra preocupación por su literatura) la diferenciación entre los diferentes signos de puntuación muy raramente es responsabilidad del autor.

▪ *Perfil de palabra simple*.- Para computar este algoritmo, se ha contrastado la frecuencia relativa de ocho de las palabras más frecuentes en los textos dubitados (*de, la, que, el, en, y, a, los*) con las frecuencias que en cada uno de los autores tienen esas palabras.

▪ *Perfil de grafema en multiposición*.- Se ha seleccionado el análisis de los 6 grafemas iniciales de palabra y se ha descartado el de los seis últimos porque, aunque para la lengua inglesa resulta una medición altamente operativa, consideramos que la flexión del español podría alterar los resultados. Para este cálculo nos hemos ayudado del programa *KfNgram* (versión 2002-2007) de William H. Fletcher.

▪ *Perfil de n-grams*.- A través de *KfNgram*, hemos calculado la frecuencia relativa de 2 y 3 gramas de palabra y de 2, 3, 4 y 5 gramas de grafema. La frecuencia relativa de 2-gramas de grafema se calcula dividiendo la frecuencia de una determinada secuencia formada por dos grafemas entre el número total de combinaciones de dos grafemas; y así, con el resto de caso.

▪ *Ratio type/token*.- La *ratio type/token* se calcula dividiendo el número de *types* (formas o palabras diferentes) entre el número de tokens (palabras totales). Si tenemos en cuenta que la lista de palabras funcionales vacías de significado (*closed set*) es mucho más reducida que la que constituye la lista de palabras significativas de una lengua (*open set*), advertiremos que la *ratio type/token* será desproporcionada en función de la longitud de los textos, pues, a partir de una determinada extensión, las palabras funcionales tenderán a repetirse pero no necesariamente las palabras lexicales. Por ello, para calcular la *ratio type/token*, tanto de los dubitados como de los indubitados, hemos tomado muestras con las primeras 500 palabras de cada texto, pues de otro modo el análisis se vería condicionado a la longitud de los textos.

Hemos establecido 3 niveles de análisis distintos: la *ratio type/token* global, la *ratio type/token* de las palabras lexicales y la *ratio type/token* de las funcionales.

▪ *Riqueza de Honoré* y *Yule.*- A través del programa *Vocalyse Toolkit* (JVocalyse v 2.05), diseñado por David Woolls, hemos podido obtener de manera inmediata los resultados de las dos mediciones de *richness* que hemos cotejado: Honoré y Yule.

6. DESARROLLO DEL PROYECTO

6.1 Procesamiento

En primera instancia se procesaron los documentos del conjunto de entrenamiento y pruebas, eliminando símbolos no alfanuméricos, así como valores numéricos respecto del contenido de los textos de dos autores (A y B), los cuales se pueden ver en la figura 1 y 2. Estos textos fueron dividisos en dos conuntos, uno de entrenamiento y otro para las pruebas. En el apéndice A se muestra el código en Java necesario para la realizacion de la fase de procesamiento.

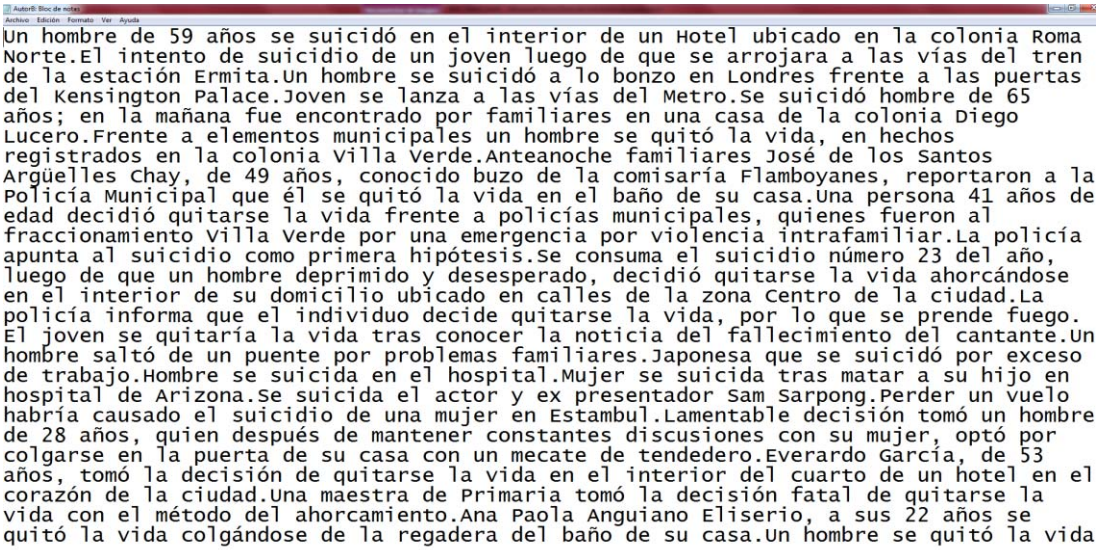
Un hombre de 59 años se suicidó en el interior de un Hotel ubicado en la colonia Roma Norte.El intento de suicidio de un joven luego de que se arrojara a las vías del tren de la estación Ermita.Un hombre se suicidó a lo bonzo en Londres frente a las puertas del Kensington Palace.Joven se lanza a las vías del Metro.Se suicidó hombre de 65 años; en la mañana fue encontrado por familiares en una casa de la colonia Diego Lucero.Frente a elementos municipales un hombre se quitó la vida, en hechos registrados en la colonia Villa Verde.Anteanoche familiares José de los Santos Argüelles Chay, de 49 años, conocido buzo de la comisaría Flamboyanes, reportaron a la Policía Municipal que él se quitó la vida en el baño de su casa.Una persona 41 años de edad decidió quitarse la vida frente a policías municipales, quienes fueron al fraccionamiento Villa Verde por una emergencia por violencia intrafamiliar.La policía apunta al suicidio como primera hipótesis.Se consuma el suicidio número 23 del año, luego de que un hombre deprimido y desesperado, decidió quitarse la vida ahorcándose en el interior de su domicilio ubicado en calles de la zona Centro de la ciudad.La policía informa que el individuo decide quitarse la vida, por lo que se prende fuego. El joven se quitaría la vida tras conocer la noticia del fallecimiento del cantante.Un hombre saltó de un puente por problemas familiares.Japonesa que se suicidó por exceso de trabajo.Hombre se suicida en el hospital.Mujer se suicida tras matar a su hijo en hospital de Arizona.Se suicida el actor y ex presentador Sam Sarpong.Perder un vuelo habría causado el suicidio de una mujer en Estambul.Lamentable decisión tomó un hombre de 28 años, quien después de mantener constantes discusiones con su mujer, optó por colgarse en la puerta de su casa con un mecate de tendedero.Everardo García, de 53 años, tomó la decisión de quitarse la vida en el interior del cuarto de un hotel en el corazón de la ciudad.Una maestra de Primaria tomó la decisión fatal de quitarse la vida con el método del ahorcamiento.Ana Paola Anguiano Eliserio, a sus 22 años se quitó la vida colgándose de la regadera del baño de su casa.Un hombre se quitó la vida

Figura 1. Ejemplo de textos del autor A.

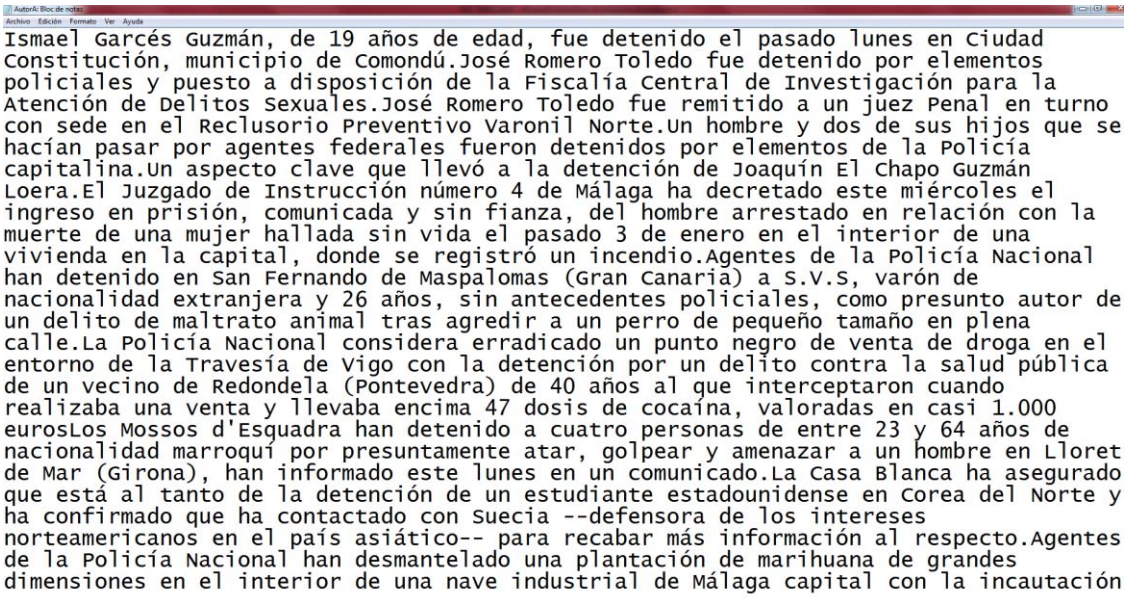
Ismael Garcés Guzmán, de 19 años de edad, fue detenido el pasado lunes en Ciudad Constitución, municipio de Comondú.José Romero Toledo fue detenido por elementos policiales y puesto a disposición de la Fiscalía Central de Investigación para la Atención de Delitos Sexuales.José Romero Toledo fue remitido a un juez Penal en turno con sede en el Reclusorio Preventivo Varonil Norte.Un hombre y dos de sus hijos que se hacían pasar por agentes federales fueron detenidos por elementos de la Policía capitalina.Un aspecto clave que llevó a la detención de Joaquín El Chapo Guzmán Loera.El Juzgado de Instrucción número 4 de Málaga ha decretado este miércoles el ingreso en prisión, comunicada y sin fianza, del hombre arrestado en relación con la muerte de una mujer hallada sin vida el pasado 3 de enero en el interior de una vivienda en la capital, donde se registró un incendio.Agentes de la Policía Nacional han detenido en San Fernando de Maspalomas (Gran Canaria) a S.V.S, varón de nacionalidad extranjera y 26 años, sin antecedentes policiales, como presunto autor de un delito de maltrato animal tras agredir a un perro de pequeño tamaño en plena calle.La Policía Nacional considera erradicado un punto negro de venta de droga en el entorno de la Travesía de Vigo con la detención por un delito contra la salud pública de un vecino de Redondela (Pontevedra) de 40 años al que interceptaron cuando realizaba una venta y llevaba encima 47 dosis de cocaína, valoradas en casi 1.000 eurosLos Mossos d'Esquadra han detenido a cuatro personas de entre 23 y 64 años de nacionalidad marroquí por presuntamente atar, golpear y amenazar a un hombre en Lloret de Mar (Girona), han informado este lunes en un comunicado.La Casa Blanca ha asegurado que está al tanto de la detención de un estudiante estadounidense en Corea del Norte y ha confirmado que ha contactado con Suecia --defensora de los intereses norteamericanos en el país asiático-- para recabar más información al respecto.Agentes de la Policía Nacional han desmantelado una plantación de marihuana de grandes dimensiones en el interior de una nave industrial de Málaga capital con la incautación

Figura 2. Ejemplo de textos del autor B.

6.2 Etiquetado de Información

Una vez que se procesaron los documentos, se procedió a etiquetar cada uno de los documento del conjunto de entrenamiento Ya identificados todos los documentos, se determinaron las etiquetas sintácticas únicas contenidas en los mismo, un ejemplo del etiquetado de textos se muestra en la Figura 3. En el apéndice B, se presenta el código en Java para realizar el etiquetado de textos. Para esta tare se ha utilizado la API de TreeTagger para Java.

Figura 3. Ejemplo de etiquetado de textos.

6.3 Extracción de Información

Este módulo es el encargado de extraer las carcaterísticas de los textos. Esta tarea consiste en extraer el volcabulario de los textos y obtener su pesado. El módulo de extracción recibe de entrada las palabras procesadas de los textos obtenidos del módulo anterior.

Se utilizará indexación aleatoria para reducir el espacio vectorial, representando todo el vocabulario como vectores de ceros y unos , como se muestra en los vectores 1 y 2 de la figura 4.

Las etiquetas sintácticas se representarán como RHR y se asociarán mediante a convolución circular a los vectores generados.

	capacitacion	moviles	juicio	nayarit	ramon	emma	delegacion	brooklyn	……	CLASE
1	0	0	0	0	0	0	0	0	……	Autor A
2	0	0	0	0	0	0	0	0	……	Autor A
3	0	0	0	0	0	0	0	0	…….	Autor B

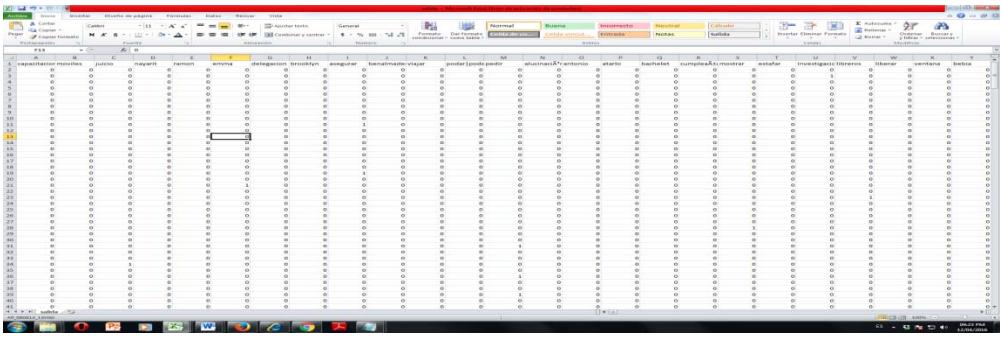

Figura 4. Ejemplo de la salida de textos representados como características.

6.4 Asignación de Autorías

Una vez obtenidos los vectores tanto para el conjunto de entrenamiento como los de prueba, se colocó toda la información en el modelo de datos. Utilizando la aplicación de escritorio WEKA, se asigna autoría, que para el caso concreto denominamos autor A y Autor B. Se han utilizado un conjunto de 373 (ver Tabla) textos de dos autores, los cuales serán utilizados para la etapa de entranamiento y pruebas.

Tabla 1. Total de textos

Texto A	205
Texto B	168
Total	373

En la Figura 5 se muestra un ejemplo de la herramienta WEKA que fue utilizada para la asignacion de autorías a los textos. En ella se probaron los algoritmos de Bayes, J48 y Tablas de Decisión.

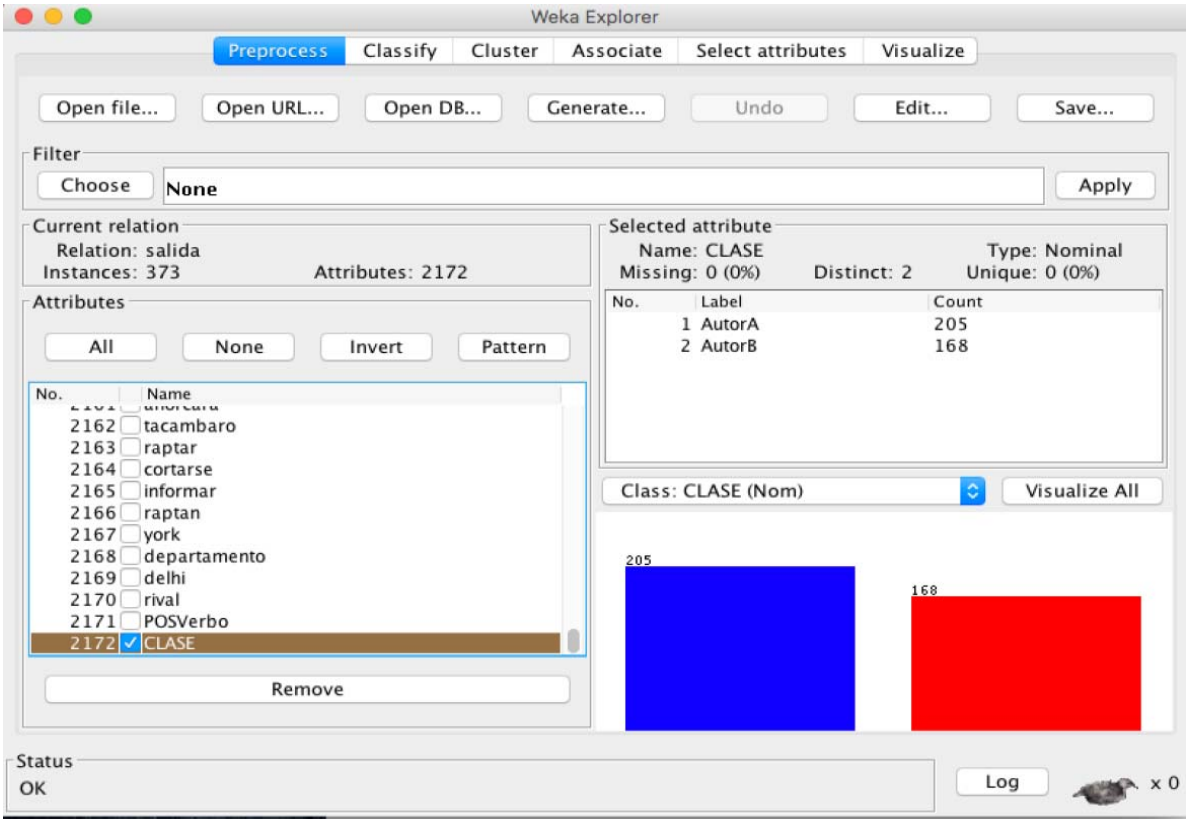

Figura 5. Ejemplo de pantalla de herramienta WEKA

7. RESULTADOS

Se realizaron pruebas para dos inputs Textos A y Textos B, con resultados de 205 y 168 respectivamente. La figura 6 muestra los resultados del algoritmo Bayes, el cual muestra una precisión de 95 %.

Bayes.txt

```
Time taken to build model: 0.07 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         358               95.9786 %
Incorrectly Classified Instances        15                4.0214 %
Kappa statistic                          0.9186
Mean absolute error                      0.0477
Root mean squared error                  0.1867
Relative absolute error                  9.6249 %
Root relative squared error             37.5313 %
Total Number of Instances               373

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.971     0.054      0.957     0.971     0.964      0.979    AutorA
                 0.946     0.029      0.964     0.946     0.955      0.979    AutorB
Weighted Avg.    0.96      0.043      0.96      0.96      0.96       0.979

=== Confusion Matrix ===

   a   b   <-- classified as
 199   6 |   a = AutorA
   9 159 |   b = AutorB
```

Figura 6. Resultados de Bayes

La figura 7 muestra los resultados del algoritmo J48, el cual muestra una precisión de 92 %.

J48.txt

```
Time taken to build model: 0.29 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         346               92.7614 %
Incorrectly Classified Instances        27                7.2386 %
Kappa statistic                          0.8531
Mean absolute error                      0.1041
Root mean squared error                  0.2552
Relative absolute error                 21.0304 %
Root relative squared error             51.2974 %
Total Number of Instances               373

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.956     0.107      0.916     0.956     0.936      0.945    AutorA
                 0.893     0.044      0.943     0.893     0.917      0.945    AutorB
Weighted Avg.    0.928     0.079      0.928     0.928     0.927      0.945

=== Confusion Matrix ===

   a   b   <-- classified as
 196   9 |   a = AutorA
  18 150 |   b = AutorB
```

Figura 7. Resultados de J48

La figura 8 muestra los resultados del algoritmo Tabla de Decisión, el cual muestra una precisión de 89 %.

DecisionTable.txt

```
Feature set: 36,149,1241,1245,1367,1471,1557,1636,1758,1779,1844,2172

Time taken to build model: 5.35 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         335               89.8123 %
Incorrectly Classified Instances        38               10.1877 %
Kappa statistic                          0.7906
Mean absolute error                      0.1445
Root mean squared error                  0.2678
Relative absolute error                 29.1769 %
Root relative squared error             53.8281 %
Total Number of Instances              373

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.985     0.208      0.852     0.985     0.914      0.94     AutorA
                 0.792     0.015      0.978     0.792     0.875      0.94     AutorB
Weighted Avg.    0.898     0.121      0.909     0.898     0.896      0.94

=== Confusion Matrix ===

   a   b   <-- classified as
 202   3 |   a = AutorA
  35 133 |   b = AutorB
```

Figura 8. Resultados de Tabla de Decisión

En tabla 2 se muestran un resumen de los resultados mostrados con lo tres algoritmos.

Tabla 2. Resumen de los resultados

Algoritmo	Porcentaje correcto de asignación de autorías
Bayes	95.9
J48	92.7
Tabla de Decisión	89.8

8. CONCLUSIONES

Nuestro estudio permite establecer un *ranking* de los algoritmos de atribución más eficaces para el idioma del español, al tiempo que permite poner en tela de juicio algunos de los que se han puesto en práctica hasta el momento, pues su bajo grado de discriminación niega su utilidad, al menos en las condiciones en las que nuestro análisis se ha realizado, y plantea la necesidad de buscar una medición más fiable para calcular la riqueza de vocabulario (que, posiblemente, esté relacionada con la distinción entre las palabras de contenido y las de función).

Sin embargo, a pesar de que esta investigación permite establecer una serie de algoritmos fiables para los estudios de atribución de autoría de textos del español actual, encontramos problemas complejos a la hora de poder adaptarlos al caso de textos anónimos auriseculares, y ello es así por dos razones, una relacionada con la fase de creación y la otra con la fase de edición y de las obras.

Por estas dos razones, una conclusión final se impone: aunque, sin duda, hay que seguir investigando (sobre todo para superar los dos problemas arriba comentados), el camino correcto pasa por el establecimiento de unos sistemas de atribución basados en la estilometría y en la cuantificación objetivadora de los fenómenos suceptibles de ser medidos. Queda por delante un camino sembrado de retos, pero también de promesas. Las vías de análisis que se abren y que nosotros aquí hemos querido evaluar, atentan contra el —misterio y el prestigio— que siempre rodean al anonimato, contribuirán sin duda a devolver el foco de atención a los aspectos puramente filológicos de la obra.

Se concluye que el algoritmo más plausible fue el de bayes toda vez que arrojo los mejores resultados, mostrando un 95.9 % de atribución de autorías.

9. REFERENCIAS BIBLIOGRAFICAS

1. Tony Plate. Holographic Reduced Representations: Convolution Algebra for Compositional Distributed Representations. In John Mylopoulos and Ray Reiter, editors, Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI), pp30-35, Morgan Kaufmann, San Mateo, CA, 1991, 6 pages.

2. Magnus Sahlgren. An Introduction to Random Indexing. In Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering, TKE (2005).

3. S.M. Ugalde Chávez, "*Sistema de recuperación de información semántico",* proyecto terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, México, 2012.

4. R.I. Morán Torres, "*Sistema de detección de plagio en archivos de texto",* proyecto terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, México, 2013.

5. J.L. Ugalde Anaya, *"Sistema clasificador de documentos de proyectos terminales usando el concepto de memoria asociativa"*, proyecto terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, México, 2011.

6. J.V. Carrera Trejo, "*Similitud semántica para la identificación de inconsistencias en datos geoespaciales conceptualizados"*, tesis de maestría, Centro de Investigación en Computación, Instituto Politécnico Nacional, México, 2010.

7. B.G. Castro Rolón, G. Sierra Martínez, J.M. Torres Moreno y I. da CunhaFanego,*"El discurso y la semántica como recursos para la detección de similitud textual"*, 2011.

8. Etiquetador Stanford. http://nlp.stanford.edu/software/pos-tagger-faq.shtml (2014)

10 APENDICES DE CODIGOS

10.1 APENDICE A

```
Archivo archivoTextos = new Archivo();
      String rutaTextos;

      rutaTextos = "Textos/AutorA.txt";
      archivoTextos.obtenerTexto(rutaTextos,"AutorA");


      rutaTextos = "Textos/AutorB.txt";
      archivoTextos.obtenerTexto(rutaTextos,"AutorB");

      vocaA.addAll(archivoTextos.getVocabulario());
      eventos=archivoTextos.getEventos();


public void obtenerTexto(String ruta, String clase) throws ParserConfigurationException,
SAXException, IOException, TreeTaggerException{

      BufferedReader br = new BufferedReader(new FileReader(ruta));

     /*   if (strTipoCod==null){
            strTipoCod="ISO-8859-1";
          }
         reader =new BufferedReader(new InputStreamReader(new
FileInputStream(strRutaArchivoP),strTipoCod));
         */

      String linea;
      while ((linea = br.readLine()) != null) {
         tags= new ArrayList<String>();
         TreeTagger tree = new TreeTagger();

          String texto=preprocesarTexto(linea);
         // Extraer lema general
          ArrayList<String> palabras=extraerLema(tree.Etiquetar(texto.split(" ")));

         //ArrayList<String> palabras=extraerLemaPorCategoria(tree.Etiquetar(texto.split(" ")),"V");
// Verbos
        // ArrayList<String> palabras=extraerLemaPorCategoria(tree.Etiquetar(texto.split(" ")),"N");
// Preposiciones
```

```
        // Gramas = 1 queda:
        vocabulario.addAll(palabras);
        eventos.add(new Evento(palabras,clase,tags));


       }
     br.close();
  }


private String preprocesarTexto(String texto) {
        String cadena="";
        texto=texto.replaceAll("\n", " ");
        texto=texto.replaceAll("á", "a");
        texto=texto.replaceAll("é", "e");
        texto=texto.replaceAll("í", "i");
        texto=texto.replaceAll("ó", "o");
        texto=texto.replaceAll("ú", "u");
        texto=texto.replaceAll("[,.]", " ");
        texto=texto.replaceAll("[\\'\\(\\)\\d]", "");
        texto=texto.replaceAll("  ", " ");
        texto=texto.replaceAll("\"", "");
        texto=texto.replaceAll(":", "");
        texto=texto.replaceAll("--", "");
        cadena=texto;
        cadena=cadena.toLowerCase();
        return cadena;
  }
```

10.2 APENDICE B

```
private ArrayList<String> extraerLema(ArrayList<Token> tokens) throws IOException,
TreeTaggerException {
     ArrayList<String> res= new ArrayList<>();

    //Elimina palabras vacias
    for(Token t:tokens){
      /*if(t.getCategoria().startsWith("N"))
```

```
        if(t.getLema().startsWith("deten") || t.getLema().startsWith("arest") ||
t.getLema().startsWith("deten")|| t.getLema().startsWith("asalt")||
t.getLema().startsWith("patad")|| t.getLema().startsWith("puñetaz")||
t.getLema().startsWith("agres")
            || t.getLema().startsWith("asesin")|| t.getLema().startsWith("fallec")||
t.getLema().startsWith("sucidi")|| t.getLema().startsWith("homicidi")||
t.getLema().startsWith("encuentr")|| t.getLema().startsWith("hallazg")
            || t.getLema().startsWith("muert") || t.getLema().startsWith("ahorcam"))
          res.add(t.getLema());*/
       if(t.getCategoria().startsWith("V") || t.getCategoria().startsWith("N") ){
           res.add(t.getLema());
           tags.add(t.getCategoria());

       }
    }

     return res;
  }


/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

package mx.uam.azc.ws.posTag;

import mx.uam.azc.ws.modelo.*;
import java.io.IOException;
import java.util.ArrayList;
import static java.util.Arrays.asList;
```

```
import javax.xml.parsers.ParserConfigurationException;
import org.annolab.tt4j.TokenHandler;
import org.annolab.tt4j.TreeTaggerException;
import org.annolab.tt4j.TreeTaggerWrapper;
import org.xml.sax.SAXException;

ArrayList<Token> ts = Etiquetar(cadena);
    System.out.println(ts.toString());
  }

  public  static ArrayList<Token> Etiquetar(String [] cadenaEntrada) throws IOException, TreeTaggerException{
    final ArrayList<Token> tokens = new ArrayList<Token>();
       // Point TT4J to the TreeTagger installation directory. The executable is expected
    // in the "bin" subdirectory - in this example at "/opt/treetagger/bin/tree-tagger"
    System.setProperty("treetagger.home", "recursos/TreeTagger");

    TreeTaggerWrapper<String> tt = new TreeTaggerWrapper<String>();
    try {
        tt.setModel("recursos/TreeTagger/modelos/sp.par:iso8859-1");
        tt.setHandler(new TokenHandler<String>() {
            public void token(String token, String pos, String lemma) {
                tokens.add(new Token(token,pos,lemma));
                //System.out.println(token + "\t" + pos + "\t" + lemma);
            }
        });

tt.process(asList(cadenaEntrada));
    }
    finally {
      tt.destroy();
    }
```

```
    return tokens;
  }
}
```

10.3 APENDICE C

```
package mx.uam.azc.ws.modelo;

import java.util.ArrayList;

public class VectorCar {
  private ArrayList<Integer> cars;
  private String clase;

   public VectorCar(ArrayList<Integer> cars, String clase) {
      this.cars = cars;
      this.clase = clase;
   }

   public VectorCar() {
   }

   public ArrayList<Integer> getPesos() {
      return cars;
   }

   public void setPeso(ArrayList<Integer> cars) {
      this.cars = cars;
   }

   public String getClase() {
```

```
        return clase;
    }

    public void setClase(String clase) {
        this.clase = clase;
    }

    @Override
    public String toString() {
        return "VectorARFF{" + "pesos=" + cars + ", clase=" + clase + '}';
    }


}


package mx.uam.azc.ws.modelo;

public class Token {
   private String  palabra;
    private String categoria;
    private String lema;


    public Token(String palabra, String categoria, String lema) {
        this.palabra = palabra;
        this.categoria = categoria;
        this.lema = lema;
    }

    public Token() {
    }
```

```
public String getPalabra() {
   return palabra;
}

public void setPalabra(String palabra) {
   this.palabra = palabra;
}

public String getCategoria() {
   return categoria;
}

public void setCategoria(String categoria) {
   this.categoria = categoria;
}

public String getLema() {
   return lema;
}

public void setLema(String lema) {
   this.lema = lema;
}

@Override
public String toString() {
   return "Token{" + "palabra=" + palabra + ", categoria=" + categoria + ", lema=" + lema + '}';
}
```

```
}

public static void main(String[] args) throws ParserConfigurationException, SAXException, IOException, TreeTaggerException {
    Archivo archivoTextos = new Archivo();
    String rutaTextos;

    rutaTextos = "Textos/AutorA.txt";
    archivoTextos.obtenerTexto(rutaTextos,"AutorA");


    rutaTextos = "Textos/AutorB.txt";
    archivoTextos.obtenerTexto(rutaTextos,"AutorB");

    vocaA.addAll(archivoTextos.getVocabulario());
    eventos=archivoTextos.getEventos();


    File archivo = new File("salida.csv");

    FileWriter escribirArchivo = new FileWriter(archivo, true);
    BufferedWriter buffer = new BufferedWriter(escribirArchivo);
    for(String pal:vocaA){
      buffer.write(pal+",");
    }
    buffer.write("POSVerbo"+",");
    buffer.write("CLASE");
    buffer.newLine();

    System.out.println(" Vocabulario obtenido...: "+vocaA.size());
```

```
    ArrayList<VectorCar> vectorCar=obtenerCaracteristicasBin();

    System.out.println(" Caracteristicas extraidas...");

    for(VectorCar ind:vectorCar){
      for(Integer peso:ind.getPesos()){
        buffer.write(peso+",");
      }
      buffer.write(ind.getClase());
      buffer.newLine();
    }
    buffer.close();
  }



    private  static ArrayList<VectorCar> obtenerCaracteristicasBin() throws IOException,
TreeTaggerException{
     ArrayList<VectorCar> vector = new ArrayList<VectorCar>();
    for(Evento ev: eventos){
      ArrayList<Integer> pesos= calcularBin(ev.getPalabras());
       int tagVerbo=0;
       for(String  tag:ev.getPosTags()){
         if(tag.endsWith("adj")) tagVerbo=1;
         if(tag.endsWith("fin")) tagVerbo=2;
         if(tag.endsWith("ger")) tagVerbo=3;
         if(tag.endsWith("inf")) tagVerbo=4;
       }
       pesos.add(tagVerbo);
      vector.add(new VectorCar(pesos,ev.getClase()));
```

```
    }
     return vector;
   }

    private static ArrayList<Integer> calcularBin(ArrayList<String> palabras) throws
IOException, TreeTaggerException{
     ArrayList<Integer> pesos = new ArrayList<>();

     for(String vocaVerb:vocaA){
       int c=0;
       for(String p:palabras){
          if(p.equalsIgnoreCase(vocaVerb))
          {
             c++;
          }
       }
       pesos.add(c);
       /*
       if(palabras.contains(vocaVerb))
        pesos.add(1);
       else
        pesos.add(0);
       */
     }
     return pesos;
    }

}
```

_____