UNIVERSIDAD DEL BIO-BIO

Facultad de Ciencias Empresariales
Departamento Sistema de Informaciones

MEMORIA PARA OPTAR A TÍTULO DE INGENIERO DE EJECUCIÓN EN COMPUTACIÓN E INFORMÁTICA

Detección de seriedad en las opiniones de usuarios en español

Alumnos: Nelson C. Retamal Silva.

Profesor Guía: Dr. Christian L. Vidal Castro.

CONCEPCÍON, 2019

AGRADECIMIENTOS.

Sin lugar a duda este proyecto de título va dedicado a cada persona que me apoyo en la dura tarea como estudiante de ingeniería, especialmente a mi familia, pareja y círculo cercano que fueron un pilar fundamental en esta carrera universitaria. Y en especial, muchas gracias a mi profesor guía Dr. Christian Vidal por mantener siempre el máximo de voluntad y disposición para ayudar y desarrollar este proyecto, al igual que a todas las personas que de alguna u otra manera, aportaron en que este proyecto de título se haya vuelto realidad.

RESUMEN

En la actualidad las aplicaciones de aprendizaje automático han dado solución a diferentes problemáticas y ha sido un tema abierto de trabajo y de nuevas investigaciones. Uno de los grandes desafíos que aún no da resultados concretos son las investigaciones desarrolladas en torno a la detección de ironía y/o sarcasmo, dado que a menudo las sociedad se expresa de esta manera ya sea en redes sociales o en la vida cotidiana. Un problema aún mayor es la detección de seriedad, la cual abarca el tema de la detección de ironía y/o sarcasmo. Resulta de suma relevancia poder detectar seriedad en comentarios, publicaciones, foros de investigación, etc. ya que la poca seriedad se puede decir que no inspira confianza y que genera escasa credibilidad.

Por lo cual en el siguiente trabajo se propone un modelo de detección de seriedad para comentarios en español usando técnicas de aprendizaje automático, la cual detectaría comentarios con ironía y/o sarcasmo, bromas y exceso de garabatos.

El proyecto se basa en primer lugar en la recolección de datos desde twitter de alguna temática de índole serio, la cual es la base para la construcción de un corpus de comentarios para la creación del modelo, el siguiente paso consta en una limpieza y etiquetado de los comentarios para identificar los serios y no serios, seguidamente se construye el corpus de comentarios agregando o eliminando ciertas características que mejoren la creación del modelo, finalmente se realizan pruebas utilizando los diferentes corpus de comentarios y utilizando diferentes algoritmos de aprendizaje automático para encontrar los mejores rendimientos de predicción junto con un software como prototipo para la predicción de comentarios ingresados por teclado.

Palabras claves: Detección de seriedad para comentarios en español, Aprendizaje automático, Seriedad, Ironía y/o Sarcasmo.

ABSTRACT

Nowadays, machine learning applications have given solutions to different problems and it has been an open topic of work and new research. One of the great challenges that still does not give concrete results are the investigations developed around the detection of irony and / or sarcasm, since society often expresses itself in this way either in social networks or in daily life. An even greater problem is the detection of seriousness, which covers the subject of the detection of irony and / or sarcasm. It is extremely important to be able to detect seriousness in comments, publications, research forums, etc., since the lack of seriousness can be said that it does not inspire confidence and that it generates scant credibility.

Therefore, in the following work we propose a seriousness detection model for comments in Spanish using automatic learning techniques, which would detect comments with irony and / or sarcasm, jokes and excess of scribbles.

The project is based in the first place on the data collection from twitter of some serious topic, which will serve for the construction of a corpus of comments for the creation of the model, then in a cleaning and labeling of comments to identify comments serious and not serious, then the corpus of comments is built adding or eliminating certain characteristics that improve the creation of the model, then tests are made using the different corpus of comments and different learning algorithms to find the best prediction yields.

Keywords: Seriousness detection for comments in Spanish, Machine Learning, Seriousness, Irony and / or Sarcasm.

Índice general

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

Actualmente internet viene siendo unos de los medios más utilizados en todo el mundo, donde aproximadamente un 40% de la población mundial tiene acceso a internet hoy en día [1], en cual los usuarios pueden acceder a diversos contenidos, así también, generar contenido propio.

Blogs, microblogs y redes sociales en general se han ido transformando hoy en día en un punto de encuentro, donde las personas pueden expresan opiniones y/o comentarios. Dentro de este contenido generado por los propios usuarios de internet se obtiene una fuente valiosa de información.

La magnitud de datos relevantes generada por los usuarios viene siendo una fuente valiosa para diferentes empresas y/o consumidores, ya que pueden convertir estos datos y generar información relevante para futuros estudios, tales como analizar comportamientos y pensamientos de la población en algún tema en específico o estudios de mercado. El poder saber lo que la gente opina respecto a un determinado producto, servicio o alguna situación en particular, puede ser fundamental y relevante.

El problema es analizar completamente cada una de estas opiniones y comprender a qué se refieren dentro de un dominio en particular, ya que podemos encontrar miles o incluso millones de comentarios haciendo casi imposible que sean analizados por humanos, por lo cual nace la necesidad de automatizar estos procesos desarrollando sistemas avanzados que permitan extraer datos, analizarlos y entregar resultados.

Existen diferentes métodos para realizar estos reconocimientos, clasificación masiva y análisis de un gran número de comentarios, como es el caso de Análisis de Sentimientos, el cual podemos definir como el tratamiento computacional de opiniones, sentimientos y otros elementos subjetivos presentes en texto [2].

Analizar opiniones en redes sociales sobre una elección presidencial, que opina la gente sobre una nueva ley, de un nuevo producto o simplemente comentarios de un foro científico, en este tipo de temáticas puede ser imprescindible e importante poder detectar comentarios que carecen de seriedad, al igual que las bromas, ironía o sarcasmos presentes en las opiniones suelen provocan que se alteren los resultados en estudios relacionados con el análisis de sentimientos.

Estas limitaciones propias de la naturaleza del texto generado por los usuarios hacen que sea de suma relevancia poder diseñar, crear un modelo y un software que pueda detectar y reconocer estas opiniones.

Por lo tanto, el objetivo de este proyecto se basa en intentar dar una solución a la problemática que se genera en diferentes estudios realizados con el Análisis de Sentimiento, la cual es la detección de opiniones que carecen de seriedad, tales como ironía, sarcasmo, bromas, y exceso de garabatos.

2. PERFIL DEL PROYECTO

En esta sección se darán a conocer el perfil del proyecto, ya sean el objetivo general, objetivos específicos, alcances de la investigación y metodología del proyecto a desarrollar.

2.1. Objetivo General.

Construir un modelo, mediante la aplicación de técnicas de Machine Learning, que permita detectar comentarios serios realizados por los usuarios en español.

2.2. Objetivos Específicos.

- Revisar la literatura y trabajos relacionados con la aplicación de técnicas computacionales para el análisis de seriedad de opiniones, tales como humor, sarcasmo, ironía, exceso de garabatos, entre otros aspectos.
- Recolectar opiniones publicadas en twitter en un dominio en particular.
- Construir un corpus, que contenga opiniones etiquetadas y que sirva para el entrenamiento y aprendizaje del modelo.
- Aplicar técnicas de machine learning, analizando características utilizadas en la clasificación y las métricas de rendimiento obtenidas.
- Construir un software que sirva de prototipo y que permita validar los resultados obtenidos del modelo.

2.3. Alcances de la Investigación.

Actualmente no se han encontrado trabajos relacionados con la detección de seriedad en comentarios en español, por lo cual se estudiarán las características de trabajos similares de detección de ironía existentes para idioma español para así poder aplicar los métodos más eficaces en la creación del nuevo modelo.

Se construirá un corpus de comentarios basados en política chilena extraídos desde twitter debidamente etiquetado dentro del dominio de serio y no serio.

Considerando que no es modelo dedicado exclusivamente a la detección de ironía y/o sarcasmo se incluirán las bromas, y exceso de garabatos como una forma de expresarse de una manera carente de seriedad para así lograr el objetivo de crear el nuevo modelo propuesto.
Finalmente, se espera que el resultado final de este trabajo de investigación sea un nuevo modelo de detección de seriedad en textos en español basado en las características de los modelos ya creados en la detección de ironía y sarcasmo mencionados en el marco teórico.

2.4. Metodología.

El desarrollo de este proyecto se dará en forma alineada a los objetivos específicos de esta investigación.

1. Realizar una revisión de literatura donde se recopilen trabajos similares de detección de comentarios poco serios, bromas, ironía, etc. en textos, sus marcos teóricos, sus entradas, resultados y aplicaciones.
2. Recopilación de comentarios y opiniones sobre la temática de política chilena desde la red social twitter.
3. Etiquetado y clasificado de los comentarios en twitter para construcción de corpus que sirva como entrada para el aprendizaje y entrenamiento del modelo de detección.
4. Aplicación de técnicas de machine learning con el corpus construido.
5. Analizar métricas de rendimiento obtenidas.
6. Analizar de forma crítica y comparativa los resultados obtenidos, realizando gráficos y discusiones al respecto.
7. Diseño y construcción de prototipo de software para validar métricas obtenidas.

3. MARCO TEÓRICO

El siguiente marco teórico busca dar un acercamiento general al lector, con respecto a los temas en la cual se da el desarrollo de este proyecto.

3.1. Revisión de literatura.

Se realizó una revisión de la literatura con la finalidad de recopilar el marco teórico sobre de la temática de esta investigación. Dicha revisión, consistió en el análisis de textos digitales, los cuales, fueron revisados e incorporados de acuerdo a filtros y criterios de selección de acuerdo

a la temática de esta investigación. La mayoría de los textos son en relación a detección de ironía y sarcasmo, ya que utilizan enfoques y características similares a las que se podrían implementar en la detección de seriedad.

Los documentos en su totalidad son en formato digital, recopilados por medio de internet en buscadores oficiales de documentos científicos tales como Google Académico, Springer, IEEE.

3.2. Definiciones

3.2.1 Definición de ironía

Podemos decir que ironía es un recurso del lenguaje en la cual existe una contradicción entre el significado y el sentido literal de las palabras sin expresarlo de manera explícita o directa. Se puede dividir en dos categorías: la ironía situacional y verbal [3]. La primera, por ejemplo, un aviso publicitario de bebidas alcohólicas que tiene un afiche en su etiqueta del producto diciendo *no beba alcohol*. La segunda se define comúnmente como "diciendo lo contrario de lo que quiere decir", donde se supone que la diferencia entre el dicho y el significado debe ser clara.

Ejemplos de ironía verbal de este proyecto podrían ser:
"Muy bien Piñera, sube el precio de los combustibles todos los días, llegaron los tiempos mejores! (?)"

Como se aprecia, la ironía verbal se interpreta como un recurso el cual se asocia a lo poca seriedad y credibilidad. En el trabajo a realizar se desarrollará sólo la ironía verbal como un recurso que representa poca seriedad.

3.2.2 Definición de sarcasmo

Actualmente, no existe un consenso respecto a las diferencias conceptuales entre ironía y sarcasmo. La RAE (Real Academia Española) define la ironía como "Expresión que da a entender algo contrario o diferente de lo que se dice, generalmente como burla disimulada" mientras que el sarcasmo es definido como "Burla sangrienta, ironía mordaz y cruel con que se ofende o maltrata a alguien o algo."[4]. El sarcasmo, a diferencia de la ironía, tiene la característica de ser burlesco y cruel, además de estar más enfocado en la ofensa de algo o alguien. Por lo cual en este trabajo se aplicará la ironía y/o sarcasmo como un único recurso ya que no es relevante detectar cada uno por separado, sino la poca seriedad que representan estos dos recursos.

3.2.3 Definición de Humor

Según el Diccionario de la Real Academia Española [4], humor es un "modo de presentar, enjuiciar o comentar la realidad, resaltando el lado cómico, risueño o ridículo de las cosas". Este trabajo se basa en detectar la poca seriedad por parte del emisor que representa el humor que es expresado por escrito. Suelen confundirse los términos ironía, sarcasmo y humor ya que se tiende a la confusión ya que varias figuras humorísticas que se escuchan o leen frecuentemente vienen acompañadas de los recursos nombrados anteriormente. Pero cabe recordar que para efectos de este estudio se analizará la presencia de humor o chistes como efecto de representar poca seriedad en los comentarios.

3.2.4 Definición de Groserías.

Las groserías poseen una carga semántica única, la cual no lograríamos expresar si las reemplazamos con alguna otra expresión, por ejemplo, si en una situación determinada nos

molesta algún comportamiento inadecuado o lo dicho por alguna persona, y nos sentimos con toda la libertad de ofenderla, tendremos dos opciones, o bien le decimos "eres una persona que posee poca inteligencia" o recurrimos a una grosería: "eres un idiota". Aunque en ambas formas lo que se está señalando es la poca capacidad intelectual del individuo, la segunda expresión refleja mayor énfasis en ese defecto. Según la real academia española groserías se puede definir como "falta grande de atención y respeto", "Rusticidad, ignorancia." o simplemente "Dicho ofensivo, indecente o grosero" [4].

3.2.5 Definición de seriedad

Según la real academia española, se define serio como "Real, verdadero y sincero, sin engaño o burla, doblez o disimulo" [4]. Esta definición nos señala la intención directa de este proyecto, la cual es la detección de seriedad, detectar comentarios sin engaños ni burlas.

Al decir que alguien expresa algo poco serio en un comentario significa que es valorado negativamente, no inspira confianza y genera poca credibilidad.

3.3. Problema a solucionar

Aun existiendo múltiples aplicaciones e investigaciones sobre el tema de la detección de ironía y sarcasmo, no existen estudios sobre algo más amplio como la detección de seriedad.

Seerat y Azam en [5] señalan, que las expresiones irónicas o sarcásticas llevan a opiniones engañosa al igual que la presencia de bromas, lo que se reafirma con la definición anterior de poca seriedad la cual representa poca credibilidad y genera poca confianza y a su vez el exceso de garabatos hace que sea valorado negativamente la opinión. Además la existencia de estas expresiones puede cambiar el sentido de una opinión, por lo que un texto con palabras positivas

construido en base a expresiones irónicas, sarcásticas o con bromas en realidad representan una expresión negativa, por lo que sí es clasificado como una opinión positiva debido a las palabras que la componen, la clasificación sería errónea.

Toda definición ya sea de groserías, ironía, sarcasmo, y humor nos lleva a la representación de algo carente de seriedad, Por este motivo, la detección de seriedad podría ser un campo abierto de investigación al incluir desafíos tan complejos como es la detección de ironía y sarcasmo , e incluir detección de bromas y exceso de garabatos presentes en las opiniones, sobre todo cuando el objeto del análisis corresponde al contenido generado por los usuarios, no solamente en redes sociales, en lugares o sitios web donde se requiera que los usuarios se expresen con comentarios serios , que generen confianza y credibilidad.

3.4. Que es twitter

Twitter corresponde a una red social la cual permite a sus usuarios publicar actualizaciones o estados en forma de textos cortos. Este tipo de servicios se denominan servicios de Microblogging, el cual corresponde a una forma de blogging (forma de comunicación basada en la Web, en la cual los usuarios a través de blogs personalizados publican contenido de distinto tipo según sus preferencias y de forma frecuente [7]) en el que los usuarios registrados publican breves mensajes de texto plano acerca de sus intereses o vida cotidiana los cuales pueden ser visualizados por otros usuarios registrados como sus amigos u observadores interesados en el contenido. La temática de lo expresados a través de Twitter varía en distintos aspectos, ya sean de la vida cotidiana hasta temas de contingencia como noticias y otros intereses [8].

Esta red social por la amplitud de temáticas, los tipos de mensajes y la disponibilidad que entrega la aplicación para desarrollar estudios y extraer información en forma masiva de la página se convierte en una fuente valiosa para la extracción de información y estudios sobre minería de datos, como ya se mencionó se pueden publicar mensajes de texto cortos (280

caracteres) identificados por un Hashtags que categoriza su contenido lo cual facilita la búsqueda por contenido generado.

3.5. Opinion Mining

Opinion Mining o Sentiment Analysis (Análisis de Sentimientos) corresponde al estudio computacional de opiniones, sentimientos, subjetividad, evaluaciones, actitudes, emociones, etc. expresados en texto [6].

Actualmente las opiniones impactan en gran medida en las decisiones y comportamientos que tomamos, como por ejemplo al momento de reservar en un hotel o ver una película, al analizar comentarios de este tipo, si observamos que la mayoría son negativos no reservamos en ese hotel o no vemos esa película, con este simple ejemplo podemos apreciar como un simple comentario de una persona puede influir en la decisión de otra.

Gracias a las características de la Web, las personas pueden compartir sus opiniones respecto a diversos temas con otros usuarios ya sea en redes sociales, páginas de internet o blogs.

Este tipo de información va en creciente aumento, por lo cual los mayores interesados suelen ser las empresas las cuales desean analizar las opiniones generadas por sus clientes para obtener conocimiento relevante a partir de esta para ser usadas en su toma de decisiones.

3.6. Machine Learning

Como ya se mencionó, la importancia y la cantidad de información generada por los usuarios va en creciente aumento, por lo cual el análisis de esta gran cantidad de información demanda técnicas complejas. Detectar seriedad de forma automática es un problema complejo sin

estudios aún realizados que demuestren soluciones, solo acercamientos de soluciones en la detección de ironía y sarcasmo.

Para las personas percibir un mensaje serio o no serio puede ser difíciles identificar por lo subjetivo que puede llegar a ser, para una máquina la tarea es aún más complicada. Sin embargo existen técnicas derivadas de la inteligencia artificial que pueden resolver estas problemáticas para así facilitar y automatizar tareas que un humano podría realizar.

3.6.1 Definición

Hoy en día la inteligencia artificial resuelve cientos de problemas que para un humano sería difícil de realizar, como ya se mencionó el procesar miles de comentarios y reconocer su polaridad o clasificarlas según un contexto sería una ardua labor y tomaría mucho tiempo y trabajo. El machine learning da una solución a esta problemática entregando herramientas para el análisis masivo de información y clasificación automática dentro de un contexto predeterminado, de esta forma mediante una cierta entrada podemos llegar a determinar su polaridad en distintos ámbitos que se requieran. El concepto de Machine Learning, se puede definir como el proceso de programar una máquina o computador para que sea capaz de entregar resultados lo suficientemente útiles en base al uso de datos de ejemplo o experiencia pasada [11]. Este proceso se basa en entrenar la máquina mediante un input de datos predeterminado dentro de un contexto, facilitando ejemplos para los cuales el algoritmo pueda identificar patrones complejos en millones de datos de entrada.

No existen algoritmos específicos para distintas tareas, cada algoritmo aprende dependiendo del contexto y lo que se requiera entrenar, existen abundancia de datos de todo tipo, y mientras más datos se le entregan al algoritmo mejor aprenderá de una tarea en específico y así llegar a una buena aproximación de los resultados [11].

3.6.2 Aprendizaje supervisado

El aprendizaje supervisado es uno de los enfoques de Machine Learning, en el que para generalizar como ejemplo se puede decir que es una tarea en la cual se le entregan ciertas preguntas al algoritmo junto con sus respuestas por parte del supervisor (Set de entrenamiento), esto con la finalidad de que el algoritmo encuentre patrones para realizar una predicción sobre una pregunta nueva.

Entonces durante el aprendizaje supervisado, los parámetros que provee el supervisor son optimizados dado los datos de entrada, para reducir al máximo el error del ajuste y que la predicción del resultado se acerque lo más posible a los valores correctos indicados por el set de entrenamiento utilizado.

Para la construcción del set de entrenamiento, se deben incluir datos con etiquetas indicando las distintas clases de datos que la máquina requiera aprender a identificar. Un caso más simple podría ser cuando se pretende realizar una clasificación de datos, se cuenta con dos clases diferentes, una clase se denomina set positivo y corresponde a un conjunto de datos que representan aquello que se pretende identificar o clasificar. La otra clase corresponde al set negativo el cual incluye ejemplos contrarios a lo que se pretende identificar [11]. Una vez que se cuenta con estos sets de entrenamiento, se debe identificar los atributos que caracterizan las distintas clases para poder realizar el entrenamiento del algoritmo de clasificación, lo que le permitirá posteriormente diferenciar estas clases para poder realizar predicciones sobre nuevos datos de entrada.

3.6.3 Algoritmos de aprendizaje supervisado

El enfoque más comúnmente utilizado es el de aprendizaje supervisado, debido a que permite verificar los resultados de clasificación midiendo el rendimiento y evaluando cada algoritmo con la finalidad de encontrar el mejor modelo.

Los algoritmos supervisados de clasificación automática son divididos en clasificadores Generativos o Discriminativos (también llamados condicionales). Los clasificadores generativos aprenden a partir de la probabilidad conjunta P(C, D) (Donde D es el documento y C la clase) y realizan la predicción maximizando la probabilidad condicional P(C|D) que se calcula utilizando el teorema de Bayes. Al contrario de los ya mencionados, los clasificadores discriminativos modelan directamente la probabilidad condicional P(C|D) para realizar la predicción, es decir, a partir de la estructura oculta de los datos sin tener en cuenta la forma en que se generan. La desventaja de los algoritmos generativos es que resuelven un problema más general como paso intermedio, en cambio, los algoritmos discriminativos resuelven el problema de clasificación directamente [10].

Según la literatura estudiada, el algoritmo generativo más utilizado es el algoritmo Naive Bayes, y dentro de los modelos discriminativos más relevantes se encuentra el algoritmo de Árboles de Decisión y el algoritmo Support Vector Machine (SVM).

3.6.3.1 Arboles de decisión

Los árboles de decisión son diagramas de flujos en forma de árbol, el cual realiza una predicción de la clase a la que pertenece un vector de atributos dado X utilizando sus elementos. El árbol inicia con un nodo raíz, para dividirse de forma progresiva en distintos nodos de decisión por medio de ramas. Cada rama representa los datos resultantes del nodo de decisión (esto generalmente divide el conjunto de datos para discriminar entre las distintas clases) y a medida que los datos recorren el árbol, van trazando un camino a lo largo de este. Eventualmente, los caminos posibles a lo largo del árbol desembocaran en un nodo hoja, el cual representa la clase a la que pertenecen los datos que llegan ahí [14].

3.6.3.2 Support Vector Machine

Corresponde a un método en el cual los datos de entrada son llevados a un espacio dimensional mayor, con el fin de que sean más separables (comparado con el espacio original). Para separarlos, se construye un hiperplano que divide los datos. Generalmente existen infinitos hiperplanos posibles para realizar esta división, por lo que se busca un hiperplano que mantenga una distancia de separación máxima, y para obtenerlo se utilizan otros dos hiperplanos paralelos a este de tal forma que al optimizar la distancia de separación de estos hiperplanos con el utilizado para la separación, se obtendrá un hiperplano con el máximo margen entre los datos y por lo tanto, la distancia entre este y el dato más cercano será máxima. El hiperplano de margen máximo se ubica entre los dos hiperplanos paralelos cercanos a los datos. La distancia entre ellos es optimizada, siendo estos dos hiperplanos paralelos quienes establecen las restricciones al problema de optimización. Todas aquellas tuplas de datos que se encuentren en estos hiperplanos paralelos se denominan vectores de soporte [15].

3.6.3.3 Naive Bayes

El clasificador Naive Bayes consiste en la predicción de la probabilidad de que un elemento pertenezca a cierta clase basándose en el teorema de Bayes [25]. En la siguiente se muestra los fundamentos del algoritmo Naive Bayes, donde dado un vector de atributos X, se obtiene la probabilidad de que pertenezca a una clase *ci*.

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

3.6.3.4 Random Forest

Random Forest es una combinación de árboles predictivos (clasificadores débiles); por lo cual podemos decir que es una modificación del Bagging, el cual trabaja con una colección de árboles incorrelacionados y los promedia [13], en el cual se tiene que cada árbol depende de los valores de un vector aleatorio de la muestra de manera independiente y con la misma distribución de todos los árboles en el bosque. La generalización de error para los bosques converge a un límite en cuanto el número de árboles en el bosque sea grande.

El método Random Forest se basa en un conjunto de árboles de decisión, es decir, una muestra entra al árbol y es sometida a una serie de test binarios en cada nodo, llamados split, hasta llegar a una hoja en la que se encuentra la respuesta. Esta técnica puede ser utilizada para dividir un problema complejo en un conjunto de problemas simples.

3.7. Métricas de rendimiento y evaluación de los resultados

3.7.1 Métricas

Para analizar los resultados de un clasificador, existen estadísticas que permiten medir que tan bueno puede llegar a ser el modelo clasificador al momento de predecir la etiqueta de cierta clase de datos. Como se mencionó anteriormente el objetivo de este proyecto es detectar cierta polaridad en una clase determinada de datos, en este caso Serio o no Serio las cuales para efectos de la siguiente definición son representadas como una polaridad Positivo y negativo.
Con estos términos de positivo y negativo es posible describir cuatro conceptos básicos, que nos permiten generar métricas de precisión para los clasificadores. Estas son descritas a continuación [14]:

- Verdaderos positivos (VP): Son aquellos datos del set positivo, que son etiquetados correctamente como positivos por el clasificador.

- Verdaderos negativos (VN): Son aquellos datos del set negativo, que son etiquetados correctamente como negativos por el clasificador.

- Falsos positivos (FP): Son aquellos datos del set negativo, que son etiquetados de forma errónea por el clasificador. Es decir, estos datos son en realidad del set negativo, pero fueron etiquetados como positivos.

- Falsos negativos (FN): Son aquellos datos del set negativo, que son etiquetados de forma errónea por el clasificador. Es decir, estos datos son en realidad del set positivo, pero fueron etiquetados como negativos.

Estos datos suelen ser resumidos en una tabla llamada matriz de confusión, la cual permite ver a grandes rasgos que tan bien se está comportando el clasificador al predecir los datos pertenecientes a las distintas clases y mostrar cómo estos han sido clasificados. En la siguiente tabla se muestra un ejemplo de dos clases en el que la clase de interés corresponde a Serio y las otras clases corresponden a No Serio así siguiendo un ejemplo relacionado a cómo quedaría una tabla de confusión de este proyecto.

		CLASES PREDECIDAS	
		Serio	No Serio
CLASES REALES	Serio	VP	FN
	No Serio	FP	VN

Tabla 1 Ejemplo matriz de confusión

Ahora, teniendo estas definiciones básicas es posible caracterizar las medidas o métricas que nos permiten evaluar el desempeño del clasificador. A continuación se describen las más significativas para el análisis [14]:

- Precisión: Corresponde a una medida de exactitud, que refleja el porcentaje de datos positivos que fueron etiquetados correctamente. Su cálculo se muestra en la siguiente ecuación donde se comparan los verdaderos positivos con todos los datos etiquetados como positivos por el clasificador.

$$\frac{VP}{VP + FP}$$

- Recall : Medida la cual representa que tan completa fue la clasificación. Refleja el porcentaje de datos positivos que fueron clasificados como tal. Como se ve en la siguiente ecuación, se obtiene la cantidad que representan los verdaderos positivos respecto al total de datos positivos reales.

$$\frac{VP}{VP + FN} = \frac{VP}{P}$$

- F-measure: Medida que combina la precisión y recall en una sola. Se define en la ecuación donde β es un número real no negativo. Corresponde a la *media armónica* entre la medida de precisión y recall (medida utilizada cuando se quiere obtener el promedio entre tasas). Generalmente se asigna el mismo peso para ambas medidas, definiendo β = 1.

$$\frac{1 + \beta^2 * \text{PRECISIÓN} * RECALL}{\beta^2 * \text{PRECISIÓN} * RECALL}$$

3.7.2 K-fold Cross Validation

Una de las metodologías más frecuentadas para la validación de modelos de predicción corresponde a K-fold Cross Validation el cual consiste en que el set completo de datos es dividido en k sub-sets de tamaño similar. Luego, el clasificador es entrenado y probado durante k iteraciones, y en cada una se aparta uno de los k sub-sets y se utiliza como set de pruebas, mientras todo el resto se utiliza como set de entrenamiento [15].

La siguiente imagen es una representación de cómo trabajaría un algoritmo con el set de entrenamiento y prueba con valor k=4.

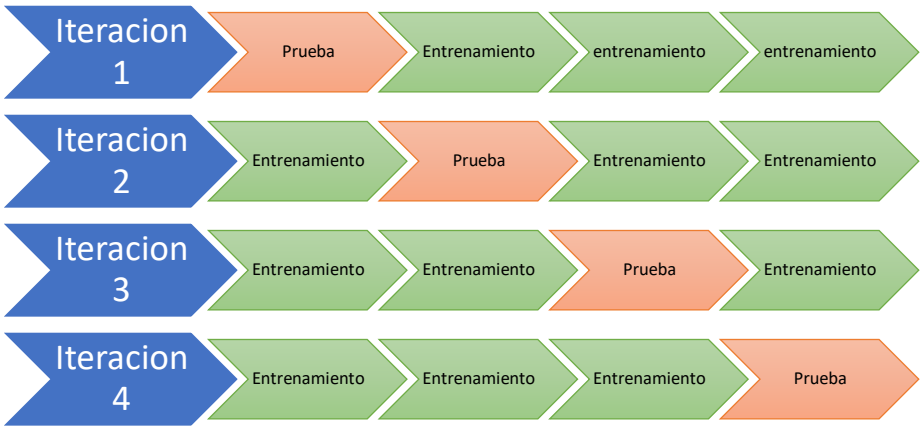

Ilustración 1: Ejemplo funcionamiento K-fold Cross Validation

3.8. Features etiquetado POS

Un feature es una característica del texto la cual puede o no resultar relevante para una tarea de procesamiento en particular. Existen features que resultan de gran utilidad en la mayoría de las tareas de clasificación y otros que resultan de utilidad seleccionando para la tarea específica a realizar. Por ejemplo: si la tarea consiste en identificar nombres propios, un feature trivial sería identificar las palabras que comienzan con mayúscula.

Uno de los features más utilizados de manera general en las tareas de procesamiento de textos es el etiquetado POS (del inglés, Part Of Speech Tagging). Este feature consiste a asignar a cada término de en este caso un corpus, una etiqueta la cual identificará si la palabra representa un sustantivo, verbo, adjetivo, artículo, adverbio, etc., dependiendo del contexto.

En clasificación de textos, suele ser de utilidad agregar a cada término un sufijo con la etiqueta asignada realizando etiquetado POS, con el fin de diferenciar aquellos términos idénticos pero que expresan significados distintos dependiendo del contexto del documento. El siguiente ejemplo muestra un comentario ya procesado con etiquetado POS.

FRASE SIN ETIQUETADO POS	FRASE CON ETIQUETADO POS
'Me los paseo con las AFP'	'Me_pp000000 los_da0000 paseo_nc0s000 con_sp000 las_da0000 AFP_np00000 '

Tabla 2: Ejemplo comentario etiquetado POS

Como se aprecia en el ejemplo, cada palabra en su continuación se agrega un sufijo predefinido dependiendo de su categoría gramatical, para así poder identificar qué representa cada palabra ya sea un sustantivo, verbo, adjetivo, etc. Las etiquetas asignadas son derivadas de la librería de etiquetado POS Standford Pos Tagger , la cual utiliza un conjunto de etiquetas diferentes para el idioma español, como por ejemplo la etiqueta ‘da0000’ son atribuidas a los artículos.[24]

3.9. StopWords

Stopwords o también conocidas como palabras vacías es el nombre que reciben aquellas palabras la cual no representan ningún significado válido como artículos, pronombres, preposiciones, etc y las cuales son eliminadas o filtras en trabajos relacionados con el procesamiento del lenguaje natural. Las palabras ya mencionadas no proporciona información relevante al clasificador por lo tanto como ya se mencionó son eliminadas durante o después

del procesamiento. Algunos ejemplos de StopWords en español son: un, desde, cierto, quien, solo, dentro, a, etc. El siguiente ejemplo muestra un comentario con Stopwords eliminados.

SIN ELIMINAR STOPWORDS	ELIMINACIÓN DE STOPWORDS
'Me los paseo con las AFP'	'Me paseo AFP'

Tabla 3: Ejemplo comentario con StopWords eliminados

El diccionario de Stopwords utilizado se encuentra adjunto en Anexo A.

3.10. Software Weka

Weka es un software de código libre desarrollado en Java, el cual se puede decir que es una colección de algoritmos de aprendizaje automático para tareas de minería de datos desarrollada por un grupo de investigadores del área del PLM de la Universidad de Waikato [16]. El paquete de distribución de Weka contiene la mayoría de los algoritmos de aprendizaje automático disponibles, pudiendo ser aplicados directamente a un conjunto de datos o llamados desde su propio código Java. Además como otras características disponibles este software contiene herramientas para pre-procesamiento de datos, clasificación, regresión, clustering, reglas de asociación y visualización.

La herramienta permite ingresar para la lectura un set de datos, la cual debe estar en formato ARFF (formato propio de Weka) para luego poder realizar las tareas requeridas. El formato ARFF solicita definir en su estructura los atributos y el tipo de dato de los elementos por los que está compuesto el set de datos.

Además Weka provee de distintas herramientas como lo son tokenizadores, stemmers, filtrado de StopWords, entre otros, todos ellos disponibles exclusivamente para el idioma inglés sin tener compatibilidad en otros idiomas.

Como ya se mencionó, el software incorpora diversos algoritmos de clasificación automática, los cuales mediante la entrada de un set de datos usados para el entrenamiento del modelo, los algoritmos realizan la clasificación, entregando métricas y rendimiento de la misma.

4. DESARROLLO DE CORPUS DE COMENTARIOS

Para poder crear un modelo de detección de seriedad usando técnicas de machine learning, uno de los detalles más importantes es una correcta construcción de un corpus de comentarios, este con la finalidad de usarlos como un set datos de entrenamiento para que los algoritmos realice el aprendizaje de una manera eficiente, junto con su correcta clasificación y a su vez siendo importante contar con una gran cantidad de datos para que el corpus sea lo suficientemente extenso y permita generar los patrones necesarios para la clasificación y detección en este caso de seriedad.

La construcción del corpus se basó en comentarios extraídos desde la red social Twitter sobre la temática de política chilena, ya que en este tipo de temáticas es relevante detectar comentarios de carácter serio.

Una ventaja es la existencia de buscadores y APIs de programación para Twitter la cual facilitan búsqueda y extracción de comentarios utilizando Hashtags para buscar y realizar una recolección rápida y masiva de datos que se atribuyen a la temática ya mencionada.

Los datos extraídos desde twitter requieren ser previamente etiquetados en función del fenómeno que se quiere identificar, en este caso Serio o No Serio, por lo cual conociendo lo extenso que puede resultar el corpus para el entrenamiento resultaría muy difícil etiquetar este gran número de datos, por lo que se realizó un plan de etiquetado involucrando a diferentes personas de diferentes rangos etarios que cumplieran la función de evaluadores con la finalidad que analicen y etiqueten el corpus de comentarios.

4.1. Hashtags utilizados en la recolección de datos

Aprovechando una de las ventajas de la red social Twitter que es la existencia de Hashtag para la categorización del contenido, se utilizó esta herramienta para la recolección de los datos.

Se escogieron Hashtags que fueron tendencia nacional los días que se recopilaron datos y que a su vez generan gran contenido en la red social , siendo todos estos relacionados a políticos o partidos políticos influyentes en chile, todo esto recopilado durante el periodo 2018 segundo semestre.

Los Hashtags que fueron Trending topic con sus respectivas fechas de extracción fueron los siguientes:

HASHTAGS	FECHA
#Piñera	12/11/2018
#Boric	12/11/2018
#Walker	13/11/2018
#Presupuesto2019	13/11/2018
#Bachelet	13/11/2018
#AulaSegura	14/11/2018
#RicardoLagos	14/11/2018
#Chadwick	15/11/2018
#Cheyre	15/11/2018
#RenovacionNacional	15/11/2018
#Udi	15/11/2018
#NuevaMayoria	20/11/2018
#FrenteAmplio	20/11/2018
#SubsecretarioUbilla	20/11/2018

#AcusacionConstitucional	20/11/2018
#CamilaVallejos	21/11/2018
#CadenaNacional	28/11/2018
#MejoresPensiones	29/11/2018
#BolsonaroaChile	29/11/2018
#SueldoMinimo	29/11/2018
#Isapres	30/11/2018

Tabla 4: Hashtags utilizados en recolección de datos

4.2. Extracción de tweets

Para la extracción de los tweets se utilizaron dos métodos, Inicialmente se utilizó la API Developer Twitter en su versión 4.0 [23] la cual al suscribirse de forma gratuita entrega credenciales standard para desarrollar aplicaciones y así permite extraer contenido masivo sobre los Hashtag deseados, siendo este contenido desde la fecha actual de publicación hasta fechas antiguas. El problema que presentó esta API, es que el contenido que generaba no filtraba si era un retweet, noticia, publicidad o tweets repetidos, ya que se necesitan solo comentarios realizados por usuarios naturales.

Teniendo en cuenta este problema de igual manera se buscó cada uno de los Hashtags seleccionados, se desarrolló una pequeña aplicación javascript con la cual se obtuvo un total de 7816 tweets.

El código de la aplicación JavaScript es el siguiente:

```
var express = require("express"),
    app = express(),
    twitter = require('ntwitter');


//configuración de los datos de nuestra app de twitter
var credentialsTwitter = new twitter({
                    //credenciales facilitadas por twitter
  consumer_key: 'GhxDbxHDN1s65HAQWTNs3voDx',
  consumer_secret: 'BceT1Q6LLjrrszTNatYdyVQWyW99kA8NQw1WpaXgdUTEqUpGSH',
    access_token_key: '2147846869-BUDgzRI3ZHuuEP71r3eJFW90YocZtNNCgnaUXV6',
  access_token_secret: '7ogtMhbmkMZfR7DApNqw5pAwY1c2GfSHB9xgSqunnIJSj'

});

credentialsTwitter.stream('statuses/filter',
    { track: ['@AulaSegura'] },
    function (stream) {
        stream.on('data', function (tweet) {
            console.log(tweet.text + "\n");
        });
        stream.on('error', function (type, info) {
            console.log(type + " " + info + "\n");
        });
    }
);
```

Ilustración 2: Código JavaScript Api Developer Twitter

El segundo método fue la extracción manual buscando cada Hashtag de forma directa en el buscador de la página de Twitter, la cual requirió mucho más tiempo pero da la posibilidad de filtrar en el momento el comentario que se almacena siendo mucho más selectivo en la calidad del comentario y en lo que se busca. De igual manera se buscó cada uno de los hashtags seleccionados y se obtuvo un total de 854 comentarios teniendo en cuenta que todos estos formarían ya definitivamente del corpus al estar seleccionado y filtrado manualmente.

4.3. Filtro manual para comentarios extraídos de API Developer Twitter

La desventaja de utilizar la API de twitter descrita anteriormente fue que sólo se podía obtener el mensaje del tweet, y no saber si es un retweet, una conversación, si es publicidad, etc. Por esta razón, fue necesario realizar una limpieza exhaustiva inicial a los datos para poder trabajar con ellos por lo cual se realizó un filtrado manual, el cual consistió en revisar, leer y seleccionar cada uno de los tweets obtenidos y así declararlos aptos o no aptos para formar parte del corpus.

Los tweets que fueron desechados son los que cumplían los siguientes criterios de exclusión:

- Tweets publicitarios.
- Tweets de noticias.
- Retweets.
- Tweets de idiomas diferentes al español.
- Tweets que no sean sobre temática de política chilena.
- Tweets de política extranjera.
- Tweets repetidos.

Una vez realizado este proceso de filtrado manual se obtuvieron los siguientes resultados:

TIPO DE TWEETS	NÚMERO DE TWEETS DESECHADOS
Publicidad	824 tweets
Noticia	3192 tweets
Retweet	1071 tweets
Idioma	19 tweets
Fuera de contexto de temática política chilena	1122 tweets
Política extranjera	565 tweets
Repetidos	449 tweets
Total tweets desechados	**7242 tweets**
Total tweets incluidos	**574 tweets**

Tabla 5: Número de Tweets eliminados filtrado manual:

4.4. Resultados extracción de comentarios.

Una vez obtenidos los comentarios filtrados, se obtuvo un set de comentarios de 1428 comentarios de diferentes temáticas sobre política chilena los cuales se encuentran listos para formar parte de la siguiente etapa de clasificado.

4.5. Etiquetado de comentarios.

De los 1428 tweets que componen el corpus, para la siguiente etapa se requiere tener etiquetados cada uno de estos tweets dentro del dominio que se desea construir el modelo, en este caso 'Serio' o 'No Serio' , por lo cual se seleccionó a un grupo de 12 personas de diferentes rangos etarios con la finalidad que etiqueten cada uno de estos comentarios dentro del dominio de 'Serio' o 'No Serio' , grupos de 3 personas etiquetaron 250 Tweets cada uno, por lo cual cada tweet fue etiquetado 3 veces. Una vez obtenido los resultados del plan de etiquetado se obtuvieron 3 etiquetas para el mismo comentario.

La concordancia entre las respuestas de los evaluadores se analizó utilizando la medida Kappa de Cohen que permite realizar un análisis para así obtener el grado de concordancia que tienen un par de evaluadores al etiquetar o responder algo [17]. El coeficiente de kappa de cohen se obtuvo mediante un complemento estadístico de Excel llamado RealStats en el cual utilizan tablas dinámicas para analizar los datos y calcular el coeficiente K.

El cálculo se realizó en base a 3 grupos diferentes, la respuesta 1 con la 2, respuesta 2 con 3 y respuesta 1 con 3, luego se realiza el cálculo del promedio de estos 3 resultados y así se obtuvo el coeficiente K.

Una vez analizados los datos se obtuvo un coeficiente de concordancia entre las 3 respuestas de los diferentes grupos de evaluadores de un k= 0,73 el cual indica un porcentaje del 73% de concordancia, lo cual podemos decir que es un resultado óptimo considerando las medidas

utilizadas en [9] la cual nos indica un 65% de concordancia como un resultado óptimo para este tipo de estudios. Los datos específicos del cálculo del coeficiente Kappa se adjuntan en Anexo C.

Ya teniendo realizada la encuesta y con un coeficiente kappa aceptable, el resultado definitivo del etiquetado es aquella respuesta que más se obtuvo en este caso Serio o no Serio, y se obtuvieron los siguientes resultados:

RESULTADOS	
Serio	782 Tweets
No Serio	646 Tweets

Tabla 6: Número de comentarios etiquetado corpus final

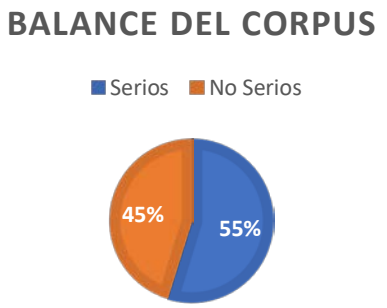

Ilustración 3: Balance del corpus

DENSIDAD DE PALABRAS DEL CORPUS	
que	3.15% (1093)
los	1.52% (526)
con	1.05% (365)
por	1.00% (345)
bachelet	0.93% (323)
piñera	0.83% (288)
del	0.80% (278)

para	0.76% (264)
las	0.65% (225)
una	0.57% (199)
como	0.50% (174)
chadwick	0.49% (171)
mas	0.49% (170)
gobierno	0.41% (143)
chile	0.41% (142)

Tabla 7: Densidad de palabras corpus

Ilustración 4: Nube de palabras corpus

COMENTARIO	ETIQUETA
Hay un mal olor en Talcahuano.... Ah llego Bachelet No A Bachelet	No Serio
Excelente noticia! legado de Bachelet avanza y pinera retrocede!!!!	Serio

Tabla 8: Extracto corpus resultante

4.6. Preprocesamiento

Antes de realizar las pruebas para poder generar un modelo de detección de seriedad, resulta necesario modificar el corpus creado, otorgándole ciertas características que han resultado exitosas en estudios anteriores relacionados con la detección de ironía y que se probaran los resultados si son efectivas para la creación del modelo de detección de seriedad

En algunos estudios [18][19][20] se hace relevante reconocer los signos de puntuación ya que en la detección de ironía o sarcasmo por lo general estos signos se hacen bastante presente en los textos, por lo cual es de suma relevancia mantener estos signo al estos estar incluidos en la detección de seriedad, lo mismo sucede con los signos de exclamación e interrogación presentes en los comentarios. Además se incluirá eliminación de Stopwords y etiquetado POS.

Las pruebas se realizarán con el software Weka en su versión 3.8.3, por lo cual el corpus debe estar en formato .arff para poder ser reconocidos directamente por el software.

Para poder realizar el preprocesamiento se utilizó un software desarrollado en [21], el cual logra eliminar signos de puntuación , exclamación, interrogación , agregar etiquetado POS y eliminar Stopwords , además tiene la ventaja de al ingresar un Excel como entrada devuelve como salida el corpus en formato .arff listo para ser leído por el software Weka.

En primer lugar es necesario para poder utilizar correctamente el programa, la instalación de las siguientes librerías:

- Apache POI: Para realizar la lectura del archivo en formato Excel.
- Stanford NLP Tagger: Para aplicar el etiquetado POS para idioma español.

Para realizar la lectura del archivo que contiene al corpus original en formato Excel, el programa permite buscar el archivo en el equipo gracias a una interfaz gráfica implementada con un

JFileChooser. A continuación, también usando una interfaz gráfica, el programa permite al usuario asignar una ruta de destino donde se guardará el archivo de salida en formato ARFF. Luego, utilizando los métodos provistos por Apache POI, se realiza la lectura del Excel para la creación de los distintos corpus, recordando que éstos están conformados por 2 columnas, siendo la primera el comentario y la segunda la clasificación de la polaridad. Ubicándose luego en una fila, se lee la columna correspondiente al comentario, aplicándose los filtros de limpieza y la selección de features de acuerdo a la versión del corpus que se desea crear para la clasificación. Se continúa con la columna de afecto, en la cual sólo se aplica el filtro de limpieza, agregándose al archivo de salida en el apartado de set de datos. Este proceso se itera para cada una de las filas disponibles en el corpus, por último, se cierra el archivo de salida y finaliza el programa.

Para que se ejecuten los filtros de limpieza mencionados anteriormente, se desarrollaron los métodos que se describen a continuación:

- **filterComillasEspacios:** Permite eliminar todo tipo de comillas, además de eliminar el espacio doble o superior.
- **filterTildes:** Permite reemplazar letras con tildes, diéresis y caracteres especiales de letras-
- **filterSignos:** Permite eliminar los signos de puntuación.
- **filterExclInt:** permite eliminar los signos de exclamación e interrogación.
- **filterStopWords:** Permite eliminar los StopWords leídos desde archivo, los cuales se encuentran posteriormente almacenados en un ArrayList. El diccionario de Stopwords se encuentra adjunto en Anexo A.

La principal ventaja del software al utilizar métodos independientes para la limpieza es poder aplicar estos filtros según la configuración de limpieza que se requiera, esto dado por el contexto, los documentos y la necesidad de estudio del investigador pensando en trabajos futuros.

Por otro lado, para poder aplicar el feature correspondiente al etiquetado POS, es necesario configurar en el programa la herramienta provista por el Stanford POS Tagger, utilizando los modelos para el idioma español. Para ello es necesario realizar lo siguiente:

- En la dirección o ruta, donde se ubica el nuevo proyecto se crea una carpeta con el nombre de taggers (puede tener otro nombre).
- Descomprimir archivos provistos por Stanford (//nlp.stanford.edu/software/tagger.shtml) y ubicarlos en la carpeta creada en el paso anterior, estos archivos tienen la extensión tagger y props.
- importar la librería para dejar disponible el modelo en el programa.
- Concluida la configuración, codificar el método principal en el que constará un constructor de la clase MaxentTagger, al cual se le tiene que enviar como parámetro un archivo entrenado, que en este caso corresponde al tagger en español.
- Por último, se etiqueta la cadena a procesar.

El diccionario de etiquetas de Standford se encuentra disponible en la siguiente referencia [24]. El corpus una vez trabajado con el software entrega el siguiente resultado del preprocesamiento como ejemplo:

CORPUS	EJEMPLO COMENTARIO PROCESADO
Sin etiquetas POS y sin eliminar Stopwords	'Pinera es minoria! Representa solo al 27% delos chilens! Chile se equivoco, q renuncie!'
Sin etiquetas POS , eliminados Stopwords	'Pinera minoria! Representa 27% delos chilens! Chile equivoco, q renuncie!'
Se incluyen etiquetas POS y no se eliminaron Stopwords	'Pinera_np00000 es_vsip000 minoria_aq0000 !_fat Representa_vmip000 solo_rg al_sp000 27_z0 %_nc0s000 delos_np00000 chilens_np00000 !_fat Chile_np00000 se_p0000000 equivoco_vmip000 ,_fc q_cs renuncie_vmsp000 !_fat '
Se incluyen etiquetas POS Y se eliminaron stopwords	'Pinera_np00000 minoria_np00000 !_fat Representa_vmip000 27_z0 %_nc0s000 delos_np00000 chilens_np00000 !_fat Chile_np00000 equivoco_vmip000 ,_fc q_cs renuncie_vmsp000 !_fat '

Tabla 9: Extracto corpus con preprocesamiento

5. EXPERIMENTOS

5.1 Introducción a experimento

La principal finalidad de los experimentos que se describen a continuación es buscar el modelo más eficaz para la detección de seriedad, esto incluye el mejor algoritmo de clasificación y las combinaciones de características necesarias para que el algoritmo pueda clasificar de mejor manera.

Las pruebas se realizarán con corpus con diferentes características cada uno, y a su vez cada uno se probará con los algoritmos más usados en la literatura como es Decision tree J48 , Naive

Bayes , Support Vector Machine y Random Forest. Los corpus que se utilizaran serán los siguientes:

CORPUS	CARACTERÍSTICA
C1	Sin etiquetas POS y sin eliminar Stopwords
C2	Sin etiquetas POS , eliminados Stopwords
C3	Se incluyen etiquetas POS y no se eliminaron Stopwords
C4	Se incluyen etiquetas POS Y se eliminaron stopwords

Tabla 10: Versiones de corpus para experimento

5.2 Experimento Nº1:

El primer experimento se realizó con todos los filtros activos, eliminando signos de interrogación, signos de exclamación y puntuación en cada corpus ya sean en C1,C2,C3 y C4. El algoritmo una vez creado el modelo se evaluará utilizando el enfoque Cross Validation K-Folder (k=10) y utilizando el filtro StringToWordVector el cual convierte los atributos de tipo String en un conjunto de atributos representando la ocurrencia de las palabras del texto.

RENDIMIENTO								
Corpus	Decision tree J48		NAIVE BAYES		SVM		RANDOM FOREST	
	Correctas	Incorrectas	Correctas	Incorrectas	Correctas	Incorrectas	Correctas	Incorrectas
C1	56.652%	43.347%	64.495%	35.504%	66.666%	33.333%	66.876%	33.123%
C2	57.142%	42.857%	64.845%	35.154%	67.787%	32.212%	67.997%	32.002%
C3	58.823%	41.176%	64.705%	35.294%	67.996%	32.002%	66.106%	33.893%
C4	57,493%	42.507%	63.585%	36.414%	67.366%	32.633%	66.176%	33.823%

Tabla 11: Experimento Nº 1 Rendimiento

MÉTRICAS OBTENIDAS												
Corpus	**Decision tree J48**			**NAIVE BAYES**			**SVM**			**RANDOM FOREST**		
	Precisión	Recall	F-Messure	Precisión	Recall	F-Messure	Precisión	Recall	F-Messure	Precisión	Recall	F-Messure
C1	0,566	0,567	0,566	0,645	0,645	0,642	0,666	0,667	0,666	0,673	0,669	0,664
C2	0,573	0,571	0,572	0,648	0,648	0,646	0,679	0,678	0,678	0,683	0,680	0,680
C3	0,588	0,588	0,588	0,647	0,647	0,645	0,680	0,680	0,680	0,662	0,661	0,658
C4	0,577	0,575	0,575	0,636	0,636	0,633	0,676	0,674	0,674	0,664	0,662	0,662

Tabla 12: Experimento Nº1 Métricas

MATRIZ DE CONFUSIÓN				
Corpus	**Decision tree J48**	**NAIVE BAYES**	**SVM**	**RANDOM FOREST**
C1	a b <-- classified as 450 300 | a = Serio 319 359 | b = NoSerio	a b <-- classified as 544 206 | a = Serio 301 377 | b = NoSerio	a b <-- classified as 527 223 | a = Serio 253 425 | b = NoSerio	a b <-- classified as 586 164 | a = Serio 309 369 | b = NoSerio
C2	a b <-- classified as 426 324 | a = Serio 288 390 | b = NoSerio	a b <-- classified as 542 208 | a = Serio 294 384 | b = NoSerio	a b <-- classified as 502 248 | a = Serio 212 466 | b = NoSerio	a b <-- classified as 488 262 | a = Serio 195 483 | b = NoSerio
C3	a b <-- classified as 454 296 | a = Serio 292 386 | b = NoSerio	a b <-- classified as 534 216 | a = Serio 288 390 | b = NoSerio	a b <-- classified as 535 215 | a = Serio 242 436 | b = NoSerio	a b <-- classified as 556 194 | a = Serio 290 388 | b = NoSerio
C4	a b <-- classified as 419 331 | a = Serio 276 402 | b = NoSerio	a b <-- classified as 532 218 | a = Serio 302 376 | b = NoSerio	a b <-- classified as 491 259 | a = Serio 207 471 | b = NoSerio	a b <-- classified as 479 271 | a = Serio 212 466 | b = NoSerio

Tabla 13: Experimento Nº1 Matriz de confusión

5.2.1 Resumen experimento Nº1

El mejor rendimiento lo obtuvo el Corpus nº 2 el cual no incluía etiquetas POS y si elimina Stopwords en los comentarios. El corpus obtuvo el mejor rendimiento con el algoritmo Random Forest obteniendo un 67,996% de predicciones correctas, lo cual se condice con sus métricas obtenidas, al igual que su matriz de confusión, el cual señala que 488 comentarios serios fueron

clasificados correctamente como serios (Verdaderos positivos) y 483 comentarios No serios fueron clasificados correctamente como No serios (Verdaderos negativos).

Como se puede observar en el siguiente gráfico todos los algoritmos de clasificación obtuvieron resultados similares en relación a su precisión de clasificación, existiendo solo de un 11,3% de diferencia entre el de peor rendimiento (J48-Corpus 1) con el de mejor rendimiento (Random Forest-Corpus 2)

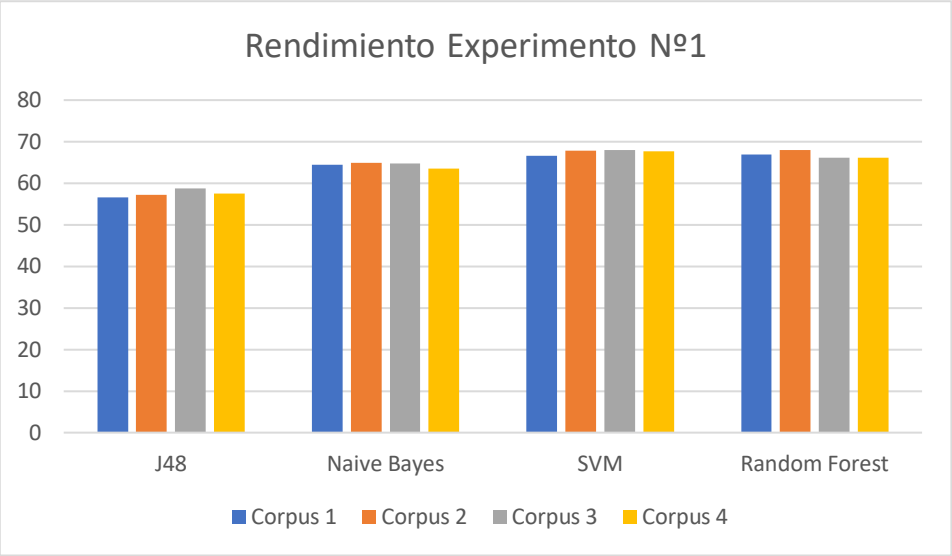

Ilustración 5: Experimento Nº 1 Gráfico de rendimiento

5.3 Experimento Nº 2

El segundo experimento consiste en repetir el experimento nº1 con una modificación adicional. Se modificó el programa en java con la finalidad que no elimina signos de exclamación e interrogación, para así verificar si existen cambios en los resultados y ver si es realmente resulta ser relevante reconocer estos signos para lograr mejores resultados en la clasificación del modelo predictivo.

Por lo cual se usó la misma metodología del experimento Nº 1 la cual consistía en utilizar C1,C2,C3 y C4 , un k=10 y filtro StringToWordVector, para la evaluación del modelo creado pero con la excepción que los corpus utilizados para la creación del modelo si incluían signos de interrogación y exclamación al contrario del experimento anterior.

RENDIMIENTO								
Corpus	**Decision tree J48**		**NAIVE BAYES**		**SVM**		**RANDOM FOREST**	
	Correctas	Incorrectas	Correctas	Incorrectas	Correctas	Incorrectas	Correctas	Incorrectas
C1	57.352%	42.647%	64.355%	35.644%	67.086%	32.913%	65.266%	34.733%
C2	56.862%	43.137%	64.986%	35.014%	68.347%	31.652%	68.557%	31.442%
C3	56.932%	43.067%	64.845%	35.154%	67.997%	32.002%	66.036%	33.963%
C4	57.142%	42.857%	64.355%	35.644%	67.857%	32.142%	66.456%	33.543%

Tabla 14: Experimento Nº 2 Rendimiento

MÉTRICAS OBTENIDAS												
Corpus	**Decision tree J48**			**NAIVE BAYES**			**SVM**			**RANDOM FOREST**		
	Precisión	Recall	F-Messure	Precisión	Recall	F-Messure	Precisión	Recall	F-Messure	Precisión	Recall	F-Messure
C1	0,574	0,574	0,574	0,644	0,644	0,641	0,671	0,671	0,671	0,654	0,653	0,648
C2	0,571	0,569	0,569	0,650	0,650	0,647	0,685	0,683	0,684	0,688	0,686	0,686
C3	0,569	0,569	0,569	0,648	0,648	0,646	0,680	0,680	0,680	0,662	0,660	0,657
C4	0,573	0,571	0,572	0,643	0,644	0,641	0,680	0,679	0,679	0,667	0,665	0,665

Tabla 15: Experimento Nº 2 Métricas

MATRIZ DE CONFUSIÓN				
Corpus	**Decision tree J48**	**NAIVE BAYES**	**SVM**	**RANDOM FOREST**
C1	a b <-- classified as 438 312 | a = Serio 297 381 | b = NoSerio	a b <-- classified as 541 209 | a = Serio 300 378 | b = NoSerio	a b <-- classified as 522 228 | a = Serio 242 436 | b = NoSerio	a b <-- classified as 565 185 | a = Serio 311 367 | b = NoSerio
C2	a b <-- classified as 410 340 | a = Serio 276 402 | b = NoSerio	a b <-- classified as 543 207 | a = Serio 293 385 | b = NoSerio	a b <-- classified as 505 245 | a = Serio 207 471 | b = NoSerio	a b <-- classified as 500 250 | a = Serio 199 479 | b = NoSerio
C3	a b <-- classified as 441 309 | a = Serio 306 372 | b = NoSerio	a b <-- classified as 542 208 | a = Serio 294 384 | b = NoSerio	a b <-- classified as 530 220 | a = Serio 237 441 | b = NoSerio	a b <-- classified as 565 185 | a = Serio 300 378 | b = NoSerio
c4	a b <-- classified as 422 328 | a = Serio 284 394 | b = NoSerio	a b <-- classified as 536 214 | a = Serio 295 383 | b = NoSerio	a b <-- classified as 497 253 | a = Serio 206 472 | b = NoSerio	a b <-- classified as 480 270 | a = Serio 209 469 | b = NoSerio

Tabla 16: Experimento Nº 2 Matriz de confusión

5.3.1 Resumen experimento Nº2

Resultados similares se obtuvieron en este experimento, al igual que el anterior el mejor rendimiento lo obtuvo el Corpus nº 2 el cual no incluía etiquetas POS y si elimina Stopwords en los comentarios. El corpus obtuvo el mejor rendimiento con el algoritmo Random Forest obteniendo un 68,557% de predicciones correctas el cual representa un aumento de un 0,56% de en relación al experimento anterior lo que demuestra que incluir signos de exclamación e interrogación en el entrenamiento del modelo si representa mejoras en el mismo.

Como se presenta en el gráfico el algoritmo Random Forest sigue siendo el que presenta los mejores resultados en cuanto a rendimiento.

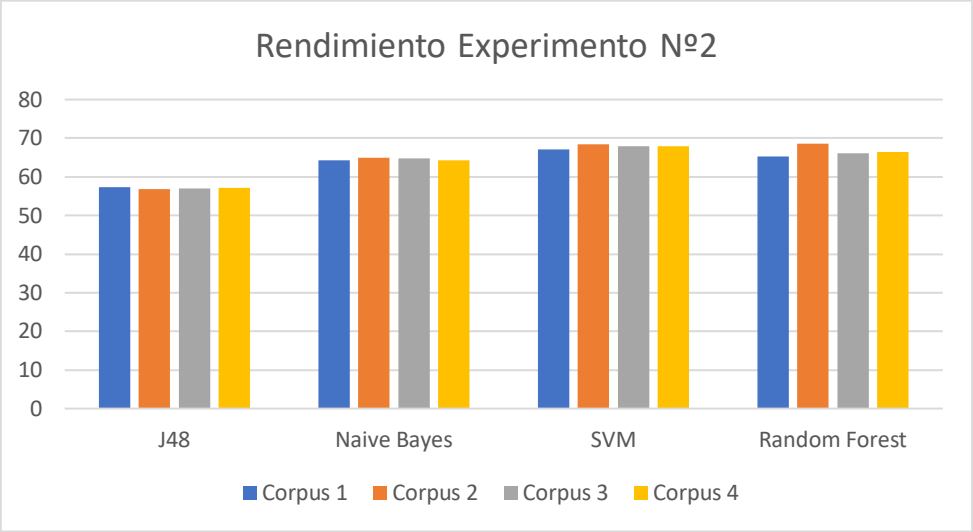

Ilustración 6: Experimento Nº2 Gráfico de rendimiento

5.4 Experimento Nº3

Siguiendo la misma lógica lo de los experimentos anteriores, nuevamente se modifica el programa en java para que no elimine ni los signos de exclamación, interrogación y puntuación, para así verificar si el rendimiento aumenta o disminuye con estas modificaciones realizadas.
Se vuelve a utilizar C1, C2, C3 y C4 con las modificaciones ya señaladas, con un k=10 para la validación del modelo y filtro StringToWordVector.

RENDIMIENTO								
Corpus	**Decision tree J48**		**NAIVE BAYES**		**SVM**		**RANDOM FOREST**	
	Correctas	Incorrectas	Correctas	Incorrectas	Correctas	Incorrectas	Correctas	Incorrectas
C1	59.453%	40.546%	64.775%	35.224%	67.016%	32.983%	67.366%	32.633%
C2	57.843%	42.156%	64.775%	35.224%	68.697%	31.302%	66.876%	33.123%
C3	54.551%	45.448%	64.565%	35.434%	69.537%	30.462%	68.137%	31.862%
C4	56.582%	43.417%	65.336%	34.663%	69.047%	30.952%	67.787%	32.212%

Tabla 17: Experimento Nº 3 Rendimiento

MÉTRICAS												
Corpus	**Decision tree J48**			**NAIVE BAYES**			**SVM**			**RANDOM FOREST**		
	Precisión	Recall	F-Messure	Precisión	Recall	F-Messure	Precisión	Recall	F-Messure	Precisión	Recall	F-Messure
C1	0,594	0,595	0,594	0,648	0,648	0,645	0,670	0,670	0,670	0,674	0,674	0,672
C2	0,581	0,578	0,579	0,648	0,648	0,646	0,688	0,687	0,687	0,673	0,669	0,669
C3	0,546	0,546	0,546	0,646	0,646	0,643	0,695	0,695	0,695	0,685	0,681	0,677
C4	0,566	0,566	0,566	0,653	0,653	0,652	0,692	0,690	0,691	0,678	0,678	0,678

Tabla 18: Experimento Nº 3 Métricas

MATRIZ DE CONFUSIÓN				
Corpus	**Decision tree J48**	**NAIVE BAYES**	**SVM**	**RANDOM FOREST**
C1	a b <-- classified as 473 277 | a = Serio 302 376 | b = NoSerio	a b <-- classified as 546 204 | a = Serio 299 379 | b = NoSerio	a b <-- classified as 520 230 | a = Serio 241 437 | b = NoSerio	a b <-- classified as 559 191 | a = Serio 275 403 | b = NoSerio
C2	a b <-- classified as 418 332 | a = Serio 270 408 | b = NoSerio	a b <-- classified as 538 212 | a = Serio 291 387 | b = NoSerio	a b <-- classified as 512 238 | a = Serio 209 469 | b = NoSerio	a b <-- classified as 474 276 | a = Serio 197 481 | b = NoSerio
C3	a b <-- classified as 416 334 | a = Serio 315 363 | b = NoSerio	a b <-- classified as 546 204 | a = Serio 302 376 | b = NoSerio	a b <-- classified as 548 202 | a = Serio 233 445 | b = NoSerio	a b <-- classified as 592 158 | a = Serio 297 381 | b = NoSerio
C4	a b <-- classified as 434 316 | a = Serio 304 374 | b = NoSerio	a b <-- classified as 539 211 | a = Serio 284 394 | b = NoSerio	a b <-- classified as 505 245 | a = Serio 197 481 | b = NoSerio	a b <-- classified as 516 234 | a = Serio 226 452 | b = NoSerio

Tabla 19: Experimento Nº 3 Matriz de confusion

5.4.1 Resumen experimento Nº3

En este experimento queda en evidencia que al eliminar los filtros del programa en java aumentó la precisión del clasificador. El corpus con mejor rendimiento fue el C3 aquel que incluía etiquetas POS y no se eliminaron Stopwords, el corpus obtuvo el mejor rendimiento con el algoritmo Support Vector Machine obteniendo un 69,537% de precisión.

Al contrario de los experimentos anteriores el que obtuvo mejor rendimiento fue Support Vector Machine, pero de igual manera como se presenta en el gráfico el algoritmo Random Forest mantiene altos porcentajes de rendimiento.

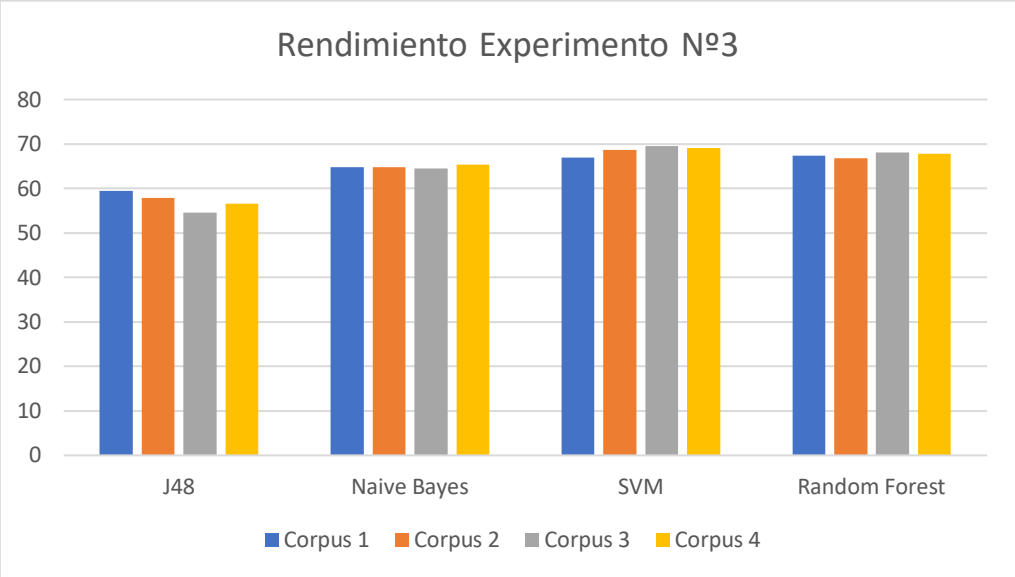

Ilustración 7: Experimento Nº 3 Grafico de rendimiento

6. ANÁLISIS DE RESULTADOS EXPERIMENTOS

En términos generales, nuestro modelo según las métricas y rendimientos señalados debería ser capaz de detectar la presencia de Seriedad y no Seriedad en textos en idioma español con un rendimiento de aproximadamente 69,5%. El resultado es aceptable en comparación a todos los trabajos revisados en esta investigación y más aun considerando la nula existencia de trabajos de detección de seriedad. Mediante el análisis que surge a la luz tras los resultados del experimento, podemos observar que a medida que agregamos ciertas características al corpus el rendimiento va en aumento, obteniendo los mejores resultados manteniendo signos de puntuación, exclamación , interrogación , agregando etiquetas POS y sin eliminar Stopwords obteniendo así un rendimiento de hasta un 69,5 por otro lado el peor rendimiento fue de un

54,5% el cual se obtuvo al no agregar ni Stopwords ni etiquetas POS pero si manteniendo signos de puntuación , exclamación e interrogación .En definitiva, nuestro modelo de detección de Seriedad tiene un rendimiento general de un 69,5 %, y se deduce que al agregar nuevas características al corpus aumenta el rendimiento de la clasificación.

El rendimiento de los 3 experimentos se resume en la siguiente tabla.

MEJOR RENDIMIENTO EXPERIMENTOS 1, 2 Y 3				
EXPERIMENTO	**ALGORITMO**	**CORPUS**	**RENDIMIENTO**	
			%CORRECTAS	**%INCORRECTAS**
Nº1	Random Forest	C2	67.9%	32.0%
Nº2	Random Forest	C2	68.5%	31.4%
Nº3	SVM	C3	69.5%	30.4%

Tabla 20: Resumen mejor rendimiento experimentos 1,2 y 3

PEOR RENDIMIENTO EXPERIMENTOS 1, 2 Y 3				
EXPERIMENTO	**ALGORITMO**	**CORPUS**	**RENDIMIENTO**	
			%CORRECTAS	**%INCORRECTAS**
Nº1	J48	C1	56.6%	43.3%
Nº2	J48	C2	56.8%	43.1%
Nº3	J48	C3	54,5%	45.4%

Tabla 21: Resumen peor rendimiento experimentos 1,2 y 3

7. APLICACIÓN DESARROLLADA PARA USO DEL MODELO.

Ya obteniendo los resultados sobre los rendimientos de los algoritmos se dispone a diseñar una pequeña aplicación en lenguaje de programación Python que permite implementar 4 modelos diferentes a partir del corpus con mayor índice de rendimiento obtenido en Weka, para así poder generar predicciones en base a un texto ingresado en la aplicación. La implementación se da de esta manera ya que al no obtener gran certeza de que un algoritmo en particular realice la predicción de manera correcta, se opta por implementar 4 algoritmos diferentes y verificar su predicción. Los algoritmos a utilizar son Naive Bayes, Support Vector Machine, Decision Tree y Random Forest.

Para comenzar se utilizó la distribución de Python llamada Anaconda en su versión 3.7, la cual facilita instalar, correr y actualizar los diferentes paquetes de aprendizaje automático que ya vienen incluidos en la distribución.

7.1 Librerías utilizadas

- **Pandas:** Es una biblioteca de código abierto con licencia BSD que proporciona estructuras de datos de alto rendimiento y fáciles de usar y herramientas de análisis de datos para el lenguaje de programación Python.

- **Scikit-learn (sklearn) :** Es una biblioteca para aprendizaje de máquina de software libre para el lenguaje de programación Python.1 Incluye varios algoritmos de clasificación, regresión y análisis de grupos entre los cuales están máquinas de vectores de soporte, bosques aleatorios, Gradient boosting, K-means y DBSCAN. Está diseñada para interoperar con las bibliotecas numéricas y científicas NumPy y SciPy.

- **Nltk:** Es un conjunto de bibliotecas y programas para el procesamiento del lenguaje natural (PLN) simbólico y estadísticos para el lenguaje de programación Python. NLTK

incluye demostraciones gráficas y datos de muestra. NLTK está destinado a apoyar la investigación y la enseñanza en PLN o áreas muy relacionadas, que incluyen la lingüística empírica, las ciencias cognitivas, la inteligencia artificial, la recuperación de información, y el aprendizaje de la máquina.

- **Tkinter:** Es una biblioteca gráfica Tcl/Tk y orientada a objeto para el lenguaje de programación Python. Se considera un estándar para la interfaz gráfica de usuario (GUI) para Python y es el que viene por defecto con la instalación para Microsoft Windows.

- **NumPy :** Más que una librería se podría definir como una extensión de Python, que le agrega mayor soporte y características para el manejo de vectores y matrices, constituyendo una biblioteca de funciones matemáticas de alto nivel para operar con esos vectores o matrices.

7.2 Funciones

- **tokenize():** Mediante la entrada de un texto, se dispone a dividir ese texto en palabras separadas en un arreglo, quedando indexada cada palabra de la oración en un índice del arreglo, como se señala en el siguiente cuadro:

TEXTO DE ENTRADA	VARIABLE PYTHON TOKENIZADA
“Que divertido es jugar futbol”	txt= [‘Que’,’divertido’ , ’es’, ‘estudiar’]

Tabla 22: Ejemplo tokenización

- **clean_corpus():** Función la cual recibe un set de comentario como parámetro, y revisa cada carácter buscando los que son diferentes dentro del dominio de *a hasta z* o *A hasta Z* definida por la expresión regular, para así desecharlos del corpus de comentarios.

- **train_test_vector():** Prepara el corpus para el entrenamiento del modelo, calculando el TF (Frecuencia de términos) en la cual representa cuántas veces se repite la palabra en el corpus por lo cual si un término aparece con relativa frecuencia en un documento, lo más probable es que sea un término importante y tendrá un TF alto. ITF (Frecuencia de datos inversa) representa la frecuencia de aparición de palabras raras o poco usadas en el texto, por ejemplo la palabra 'el' sí es bastante frecuente encontrarla en el set de comentarios tendrá un ITF bajo por su alta frecuencia y si existe una palabra única que solo se señala una vez tendrá un ITF alto.

- **ingreso_teclado ():** Función que es activada mediante un evento de un botón de la interfaz, el cual toma un comentario escrito en un campo de texto para así ser evaluado por los modelos y realizar su predicción , la cual la respuesta es enviada en formato de texto a otro campo de texto definido dentro de la interfaz.

- **lectura_txt():** Función que se activa mediante un evento de un botón de la interfaz, la cual procede a abrir una interfaz gráfica que permite seleccionar un archivo en formato .txt con la finalidad de leer su contenido y extraer cada línea del texto y realizar su predicción.

- **Escritura():**Función que se activa mediante un evento de un botón de la interfaz, la cual procede a abrir una interfaz gráfica que permite seleccionar donde se almacenará el archivo .txt el cual contendrá los resultados de la predicción actual presente en la pantalla.

7.3 Creación del Software en Python

El programa diseñado en Python se basa en los siguientes pasos señalados a continuación:

1. Lectura de archivo .csv el cual contiene el corpus
2. Limpieza de corpus.
3. Dividir el corpus en Set de entrenamiento y Set de prueba.
4. Cálculo TF-ITF del corpus.
5. Llamada a los modelos predictivos.
6. Entrenamiento de los modelos entregando como parámetros el set de entrenamiento correspondiente al corpus.
7. Imprime por pantalla rendimiento de los algoritmos usando la técnica Cross validation K fold =10.
8. Permite mediante una pequeña interfaz gráfica el ingreso de una oración la cual es evaluada por los modelos señalando su etiqueta de Serio o No serio, así obteniendo 4 respuestas de etiquetado
9. Se realiza un pequeño cálculo de porcentaje de la respuesta de predicción de Serio y No Serio para así verificar que predicción es la más repetida.
10. El software da la posibilidad de leer un archivo de texto en el cual dentro de su contenido cada línea es un comentario nuevo, así el software lee el archivo y evalúa cada uno de ellos
11. Se da la opción de guardar en un archivo de texto los resultados que están presentes en pantalla.

El código de la aplicación queda adjunto en Anexo D.

7.4 Interfaz gráfica

La interfaz gráfica fue diseñada mediante la librería Tkinter y el diseño sigue el siguiente modelo de maqueta:

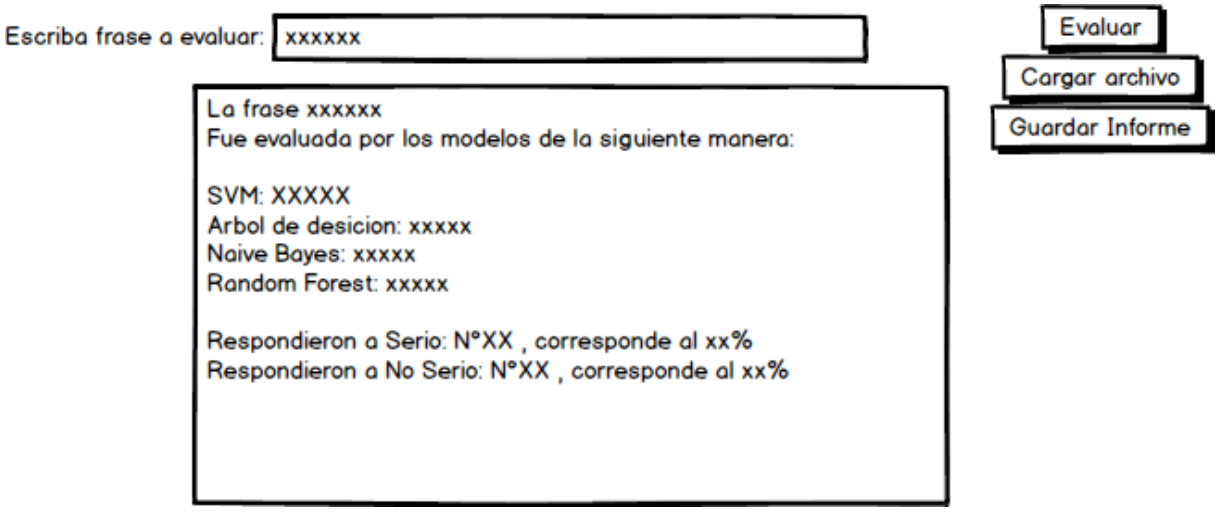

Ilustración 8: Mockup diseño interfaz de software

7.5 Prototipo de software

Una vez desarrollado el software se probó su funcionamiento con 2 diferentes comentarios ingresados por teclado para probar la predicción que nos entrega.

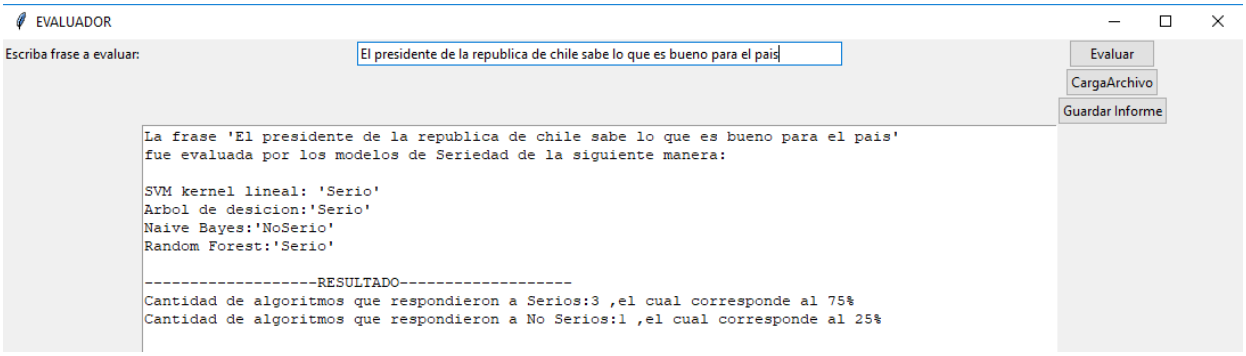

Ilustración 9: Recorte de pantalla prueba evaluación Serio

Como se puede apreciar según los modelos, 3 de ellos da como predicción Serio representando el 75% , y solo uno da como predicción No Serio, así que por voto mayoritario podemos decir que la predicción definitiva según el software es Serio.

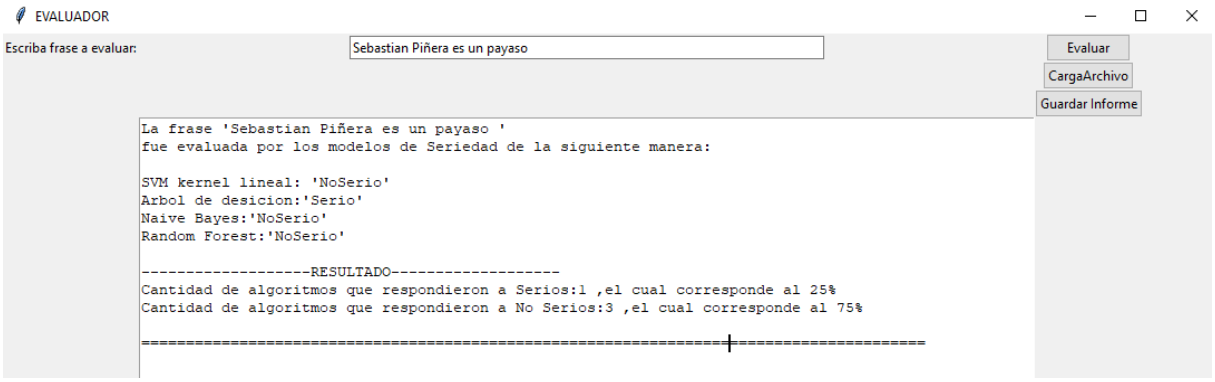

Ilustración 10: Recorte de pantalla prueba evaluación No Serio

Al contrario de la evaluación anterior, se da el caso de que 3 de los algoritmos dan como resultado No Serio, y solo uno Serio, y al igual que el ejemplo anterior por voto mayoritario podemos decir que la predicción definitiva según el software es No Serio.

7.6 Prueba de predicción de prototipo de software

Para evaluar el rendimiento al poner en práctica el prototipo de software, se realizará una breve prueba la cual consiste en extraer manualmente 15 comentarios aleatorios desde Twitter, atribuidos a la temática de política chilena y se solicitará a 1 evaluador para que etiquete los comentarios en función del dominio de ‘Serio’ y ‘No Serio’ para así comparar los resultados con la predicción realizada por el software.

Una vez realizado el experimento se obtuvieron los siguientes resultados:

Comentario	Etiqueta	Predicción del software		
		%Serio	%No Serio	Voto Mayoritario
Me imagino que este gobierno tan acostumbrado a hecharle la culpa al anterior, reconocerá el trabajo del Gobierno de la Pdta. Bachelet.	Serio	50%	50%	Indeterminado
Tanto revuelo por este pelagato, nihay con bolsonaro	No Serio	0%	100%	No Serio
Le guste o no Bolsonaro es democracia pura. Aborrezco su mente pero amo la democracia	Serio	0%	100%	No Serio
Piñera debería regalarle a Bachelet una entrada a un concierto de Miguel Bosé	No Serio	50%	50%	Indeterminado
Leo que la pifiadera a Piñera en el concierto fue monumental. ¿Para qué se expone, digo yo? ¿O de verdad cree que la gente lo quiere?	Serio	25%	75%	No Serio
Gente que aplaude pifias a Piñera, pero condena críticas a Bachelet por incompetencia. Siendo ambos igual de incompetentes	Serio	50%	50%	Indeterminado
Después d tanto desastres d la naturaleza y la lenta respuesta del Gbno Bachelet, pienso q los 33 mineros se les habrían muerto.	No Serio	25%	75%	No Serio
Bachelet, para los pueblos, para los trabajadores, para las masas, fue tan nefasta como lo es actualmente Piñera, pero los bacheletistas no quieren reconocer esto, aunque la realidad porfiada se los muestre en la cara una y otra vez.	Serio	75%	25%	Serio
Crecen las arcas de piñera y sus amigos empresarios... El pueblo seguirá miserable, cómo lo quieren	Serio	75%	25%	Serio
En el gobierno de chile no son blancas palomas nuevamente pillaron a estos corruptos!	Serio	0%	100%	No Serio
Si Carola Urrejola no es capaz de controlar su odio contra un invitado, no esta calificada para participar de un programa politico. Pesimo desempeño	Serio	50%	50%	Indeterminado
Mientras la imagen de Michelle Bachelet comienza a elevarse en la ONU. Viene su hijo, Sebastián Davalos, con su prepotencia de siempre y reflota un tema pasado.	Serio	100%	0%	Serio
Medios de comunicación ligados a la izquierda haciendo ridículo intentando lavar la imagen de davalos y su mami la inepta Bachelet.Creen qué la gente es wna?	No Serio	0%	100%	No Serio
Chadwick es la peor basura de la política chilena .El 1 lugar no se lo quita nadie.	No Serio	25%	75%	No Serio
Andrés Chadwick definitivamente es el Señor Smithers de este gobierno.	No Serio	0%	100%	No Serio

Tabla 23: Resultado experimento Prototipo de software

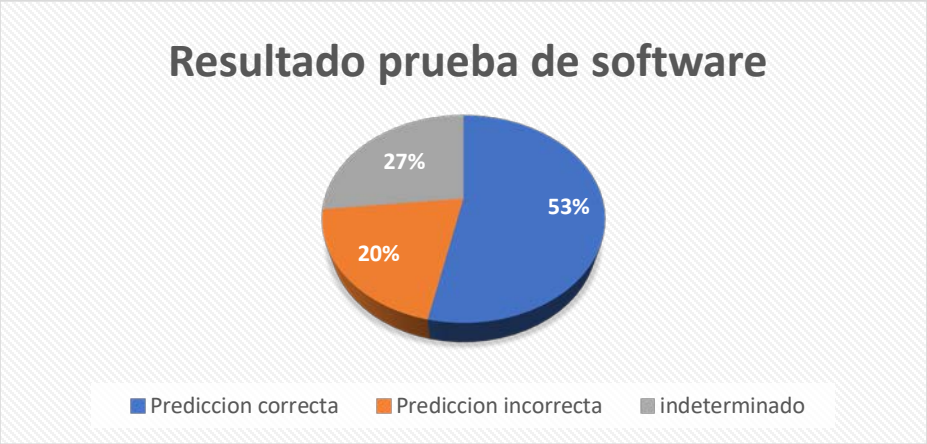

Ilustración 11: Gráfico resultado prueba de software

7.7 Conclusión prueba de software

El prototipo de software diseñado crea el modelo de forma correcta y sus funcionalidades cumplen con lo previsto. Una vez llevándolo a la práctica, según el experimento realizado podemos decir que solo existe un acuerdo mutuo de un 53% entre la predicción del software y el etiquetado del evaluador, un desacuerdo de un 20% y un 27% de una respuesta indeterminada en la cual la mitad de los algoritmos dieron una respuesta y los demás otra.

En relación al prototipo de software podemos decir que sus funcionalidades son una ventaja para desarrollar nuevas pruebas, evaluar de forma masiva varios Set de pruebas, ensayar diferentes mejoras a los algoritmos de predicción e ir verificando sus rendimientos.

8. CONCLUSIONES

Mediante este trabajo se propuso un modelo de detección Seriedad en textos escritos en español, a partir de los enfoques estudiados en una revisión sistemática de la literatura en la cual se revisaron artículos relacionados con la detección de ironía, sarcasmo, humor y minería de opinión, así tomando como base estos trabajos para crear el nuevo modelo de detección de seriedad.

Para el desarrollo de este nuevo modelo se creó un corpus de 1428 comentarios extraídos desde la red social Twitter enfocados directamente en la temática de política chilena por la relevancia que tiene la seriedad en estos tipos de temas. Estos comentarios fueron analizados por evaluadores humanos y fueron etiquetados como serios y no serios con los cuales se procedió a evaluar el modelo presentado.

Para el experimento de evaluación utilizamos 4 algoritmos de clasificación más usados en la literatura como es Árboles de decisión (J48), Naive Bayes , Support Vector Machine y Random Forest , por cada uno de estos algoritmos se realizaron 3 pruebas distintas con 4 corpus diferentes cada uno usando el enfoque de validación Cruzada.

El corpus C1 contenía los comentarios sin etiquetas Pos y sin eliminar Stopwords, el corpus C2 contenía los comentarios sin etiquetas Pos pero se eliminaron todos los Stopwords dentro de un listado predefinido, el corpus C3 se agregaron etiquetas Pos pero no se eliminaron los Stopwords y finalmente el C4 se agregaron las etiquetas Pos y se eliminaron los Stopwords

Como ya se mencionó cada uno de estos comentarios se probó en 3 experimentos diferentes. El primer experimento se basó en probar cada uno de estos 4 corpus pero con la característica que se eliminaron todos los signos de puntuación, exclamación e interrogación de los corpus, el segundo experimento solo se eliminaron signos de puntuación y el tercer experimento se mantuvieron los signos de puntuación exclamación e interrogación.

Las pruebas realizadas con el software Weka obtuvieron resultados de un 69,5% de precisión como el más alto obtenido en el experimento Nº3 con el Corpus C3.

Por otro lado, al llevar el modelo a la práctica mediante el prototipo de software de detección de seriedad desarrollado, se refleja que el modelo no es tan certero como lo señalan las métricas, una hipótesis de esto, es que el modelo creado es demasiado dependiente del dominio, específicamente del corpus con el cual se entrenó, el cual debería podría ser más extenso para lograr más exactitud en la predicción del modelo.

Por otro lado, al llevar el modelo a la práctica mediante el prototipo de software de detección de seriedad desarrollado, se refleja que el rendimiento es menor cuando se evalúa con comentarios externos al corpus. En la prueba realizada solo se obtuvo un 53% de certeza lo que demuestra que no es modelo certero.

Finalizando podemos concluir que al agregar diferentes características probadas anteriormente en los trabajos de detección de ironía o sarcasmo son la base para la detección de Seriedad, entregando resultados prometedores para futuros trabajo y abriendo posibilidades de futuras investigaciones aplicando nuevas características al corpus como por ejemplo análisis de N-Gramas , Skipgramas, reconocimiento de emoticones, etc., tal vez aumentando el tamaño de comentarios del corpus o agregar nuevos algoritmos en el prototipo de software.

8.1. Trabajos Futuros

Basándose en toda la investigación realizada se dan recomendaciones e ideas de trabajos futuros relacionados a este proyecto.

- Mejorar y aumentar el tamaño del corpus de comentarios, tal vez incluir diferentes temáticas de dominio, no solo política chilena, sino diferentes dominios donde es relevante detectar seriedad.
- Desarrollar experimentos con nuevos algoritmos de aprendizaje automático.
- Expandirse a otros idiomas.
- Utilizar nuevas características desarrolladas para la detección de ironía como el análisis de N-Gramas, Skipgramas, reconocimiento de emoticones, etc. con la finalidad de aumentar el rendimiento de clasificación.

REFERENCIAS

[1] R. T. S. Project, Internet live stats." http://www.internetlivestats.com/internet-users/", Consultado el 15/12/2018.

[2] B. Pang and L. Lee, "Opinion mining and sentiment analysis," Foundations and trends in information retrieval, vol. 2, no. 1-2, 2008. Páginas: 1-135.

[3] Erik Forslid y Niklas Wikén. "Automatic irony- and sarcasm detection in Social media. *Student thesis Master Programme in Engineering Physics*", 2015.

[4]Real Academia Española (2001). Diccionario de la lengua española. url: http://www.rae. es/recursos/diccionarios/drae, Consultado el 12/12/2018

[5] B. Seerat and F. Azam, "Opinion mining: Issues and challenges (a survey)," International Journal of Computer Applications , 2012.

[6] B. Liu, "Sentiment analysis and opinion mining," Synthesis Lectures on Human Language Technologies, vol. 5, no. 1, 2012. Páginas: 1-167.

[7] B. A. Nardi, D. J. Schiano, and M. Gumbrecht, "Blogging as social activity, or, would you let 900 million people read your diary?" , 2004.,Paginas 222-231.

[8] A. Java, X. Song, T. Finin, and B. Tseng, "Why we twitter: understanding microblogging usage and communities," in Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis, Páginas. 56-65.

[9] Cristina Bosco, Viviana Patti, Andrea Bolioli, y Departamento de Informática. Developing Corpora for Sentiment Analysis : "The Case of Irony and Senti – TUT (Extended Abstract)."

[10] a Ng and M. Jordan, "On generative vs. discriminative classifiers: A comparison of logistic regression and naive bayes," *Proc. Adv. Neural Inf. Process.*, vol. 28, no. 3 2002,Paginas. 169–187, Universidad del Bío-Bío.Sistema de Bibliotecas – Chile

[11] E. Alpaydin, "Introduction to Machine Learning". The MIT Press, 2010.

[12] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten, "The WEKA data mining software: An update," vol. 11, no. 1 2009, Páginas 10–18.

[13] Hastie, T., Friedman, J., & Tibshirani, R." The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer.", 2001.

[14] J. Han and M. Kamber, "Data Mining: Concepts and Techniques 3rd Ed. Morgan kaufmann", 2012.

[15]D. L. Olson and D. Delen, "Advanced data mining techniques. Springer Science & Business Media", 2008.

[16] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten, "The WEKA data mining software: An update," vol. 11, no. 1, 2009, Páginas 10–18.

[17] B. Levin J. L. Fleiss y M. C. Paik, "Statistical methods for rates and proportions." ,2013.

[18]Antonio Reyes, Paolo Rosso, y Tony Veale. “A multidimensional approach for detecting irony in Twitter. *Language Resources and Evaluation”*, 2013, Páginas: 239–268.

[19]Irazu Hernandez-Farıas, Jose Benedi, y Paolo Rosso. “Applying Basic Features from Sentiment Analysis for Automatic Irony Detection.”, 2015, Páginas: 121–129.

[20] P Carvalho, L Sarmento, M Silva, y E de Oliveira. “Clues for detecting irony in usergenerated contents: oh...!! It’s “so easy”; -)”, 2009, Páginas: 53–56.

[21] Jorge Andres Elgueta Morales,” Comparación de rendimiento de técnicas de aprendizaje automático para análisis de afecto sobre textos en español” , 2017.

[22] Landis, R., & Koch, G. (1977). An Application of Hierarchical Kappa-type Statistics in the Assessment of Majority Agreement among Multiple Observers. *International Biometric Society, 33*(2), 363-374.

[23] Getting started. Developer Twitter https://developer.twitter.com/en/docs/basics/getting-started .Consultado el 27/03/2019

[24] Spanish FAQ for Stanford CoreNLP https://nlp.stanford.edu/software/spanish-faq.shtml Consultado el 27/03/2019

[25] Micheline Kamber Jiawei Han y Jian Pei. *Data Mining—Concepts and Techniques, 2nd ed., Morgan Kaufmann, 2006.* 2011. URL http://hanj.cs.illinois.edu/bk3/.

ANEXOS

A. Listado de Stopwords

Un, una, unas, unos, uno, sobre, todo, también, tras, otro, algún, alguno, alguna, algunos, algunas, ser, es, soy, eres, somos, sois, estoy, esta, al, se, estamos, estais, están, como, en, para, atrás, porque, por qué, estaba, ante, antes, siendo, ambos, pero, por, fui, fue, fuimos. fueron, hacer, hago , hace, hacemos, haceis, hacen, cada, fin, incluso, primero, desde, conseguir, consigo, consigue, consigues, conseguimos, ir, voy ,va, vamos, vais, van, vaya, ha, tener, tengo, tiene, tenemos, teneis, tienen, el, la, lo, las, los, su, aquí, mio, tuyo, ellos, ellas, nos, nosotros, vosotros, vosotras, si, dentro, solo, solamente , sabemos, sabeis, saben, ultimo, largo, bastante, haces, muchos, aquellos, aquellas, sus, entonces, cierto, ciertos, cierta, ciertas, dos, bajo, arriba, encima, usar, uso, usas, usa ,usamos, usais, usan, muy, era, eras, eramos, eran, modo, bien, cual, cuando, donde , quien, con, entre, sin, podríais, yo, aquel.

B. Entorno de desarrollo

Se utilizó durante todos los experimentos y desarrollo un ordenador marca Asus X556U , con procesador Intel Core i5 7200U 2.71 GHZ, memoria ram 12 GB. El ordenador cuenta con un sistema operativo Windows 10 Single lenguage de 64 bits.

C. Cálculo coeficiente Kappa de Cohen.

NIVEL DE ACUERDO EVALUADOR 1 VS 2			
Etiquetas	No Serio	Serio	**Total general**
No Serio	608	68	**676**
Serio	129	623	**752**
Total general	**737**	**691**	**1428**

Tabla 24 Anexo: Matriz de confusión cálculo kappa evaluador 1 vs 2

COHEN'S KAPPA	
Alpha	0,05
kappa	**0,72456185**
std err	0,0181977
lower	0,68889502
upper	0,76022868

Tabla 25 Anexo: Coeficiente kappa evaluador 1 vs 2

NIVEL DE ACUERDO EVALUADOR 2 VS 3			
Etiquetas	No Serio	Serio	**Total general**
No Serio	576	161	**737**
Serio	70	621	**691**
Total general	**646**	**782**	**1428**

Tabla 26 Anexo: Matriz de confusión cálculo kappa evaluador 2 vs 3

COHEN'S KAPPA	
Alpha	0,05
kappa	**0,67746011**
std err	0,01939677
lower	0,63944313
upper	0,71547708

Tabla 27 Anexo: Coeficiente kappa evaluador 2 vs 3

NIVEL DE ACUERDO EVALUADOR 1 VS 3			
Etiquetas	No Serio	Serio	**Total general**
No Serio	589	87	**676**
Serio	57	695	**752**
Total general	**646**	**782**	**1428**

Tabla 28 Anexo: Matriz de confusión cálculo kappa evaluador 1 vs 3

COHEN'S KAPPA	
Alpha	0,05
kappa	**0,79729186**
std err	0,01600835
lower	0,76591608
upper	0,82866765

Tabla 29 Anexo: Coeficiente kappa evaluador 1 vs 3

Promedio Índice kappa: 0,73310461

DESDE	HASTA	FUERZA CONCORDANCIA
0,0	0,0	Pobre (Poor)
0,01	0,20	Leve (Slight)
0,21	0,40	Aceptable (Fair)
0,41	0,60	Moderada (Moderate)
0,61	0,80	Considerable (Substantial)
0,81	1,00	Casi perfecta (Almost perfect)

Tabla 30 Anexo: Rangos concordancia coeficiente kappa

El coeficiente kappa (*k*) toma valores entre -1 y +1; mientras más cercano a +1, mayor es el grado de concordancia inter-observador. Por el contrario, un valor de *k*= 0 refleja que la concordancia observada es precisamente la que se espera a causa exclusivamente del azar [22].

D. Código aplicación Python prototipo de software.

```
import numpy as np
import pandas as pd
from sklearn import svm
from sklearn.metrics import classification_report
import nltk
import nltk.data
#nltk.download('punkt')
from sklearn.cross_validation import train_test_split as tts
from sklearn import cross_validation
import re
from tkinter import *
from tkinter.ttk import *
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn import tree
from sklearn import preprocessing
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import BernoulliNB
from sklearn.ensemble import RandomForestClassifier
from tkinter.filedialog import asksaveasfilename

# FUNCIONES
# ------------------------------------------------------------------------
#cualquier caracter que sea diferente de az o AZ  se elimina
def clean_corpus(corpus):
  xcorpus = corpus.get_values()

  for i in range(len(corpus)):
    xcorpus[i] = re.sub("[^a-zA-Z]"," ", corpus[i])
    xcorpus[i] = ' '.join(xcorpus[i].split())
  return xcorpus

#Stringtowordvector
def tokenize(text):
    tokens = nltk.word_tokenize(text,  language='spanish')
    return tokens
```

```
#calculo tf(Term Frequency) idf(Inverse Data Frequency)
def train_test_vector(xtrain, xtest):
   vectorizer = TfidfVectorizer(tokenizer=tokenize, max_df = 0.80, use_idf=True,min_df=1)
   vector_train  = vectorizer.fit_transform(xtrain)
   vector_test   = vectorizer.transform(xtest)
   return vector_train, vector_test, vectorizer



data = pd.read_csv('corpus/CORPUS_PRUEBA.csv')

corpus = clean_corpus(data['comentario'])

xtrain, xtest, ytrain, ytest = tts(corpus, data['etiqueta'], train_size=1427,random_state=0)
xtrain, xtest, vectorizer = train_test_vector(xtrain=xtrain, xtest=xtest)


#MODELOS

SVMLINEAR = svm.SVC(kernel='linear')
SVMLINEAR.fit(X=xtrain, y=ytrain)

ARBOL = tree.DecisionTreeClassifier()
ARBOL.fit(xtrain, ytrain)

NAIVEBAYES = BernoulliNB()
NAIVEBAYES.fit(xtrain.toarray(),ytrain);

RANDOMFOREST = RandomForestClassifier()
RANDOMFOREST.fit(xtrain,ytrain)
```

```
print('--------------------------------------------------------------------------')
precisionSVM = cross_validation.cross_val_score(estimator=SVMLINEAR,
                                        X=xtrain, y=ytrain,
                                        cv=10)
print('Precision SVM LINEAL: {}'.format(precisionSVM))
print('Precision SVM LINEAL promedio: {0: .3f} +/- {1: .3f}'.format(np.mean(precisionSVM),
                                       np.std(precisionSVM)))
print('--------------------------------------------------------------------------')


print('--------------------------------------------------------------------------')
precisionNAIVE = cross_validation.cross_val_score(estimator=NAIVEBAYES,
                                        X=xtrain, y=ytrain,
                                        cv=10)
print('Precision NAIVEBAYES: {}'.format(precisionNAIVE))
print('Precision NAIVEBAYES promedio: {0: .3f} +/- {1: .3f}'.format(np.mean(precisionNAIVE),
                                       np.std(precisionNAIVE)))
print('--------------------------------------------------------------------------')

print('--------------------------------------------------------------------------')
precisionARBOL = cross_validation.cross_val_score(estimator=ARBOL,
                                        X=xtrain, y=ytrain,
                                        cv=10)
print('Precision ARBOL: {}'.format(precisionARBOL))
print('Precision ARBOL promedio: {0: .3f} +/- {1: .3f}'.format(np.mean(precisionARBOL),
                                       np.std(precisionARBOL)))
print('--------------------------------------------------------------------------')

print('--------------------------------------------------------------------------')
precisionFOREST = cross_validation.cross_val_score(estimator=RANDOMFOREST,
                                        X=xtrain, y=ytrain,
                                        cv=10)
print('Precision RANDOM FOREST: {}'.format(precisionFOREST))
print('Precision RANDOM FOREST promedio: {0: .3f} +/- {1: .3f}'.format(np.mean(precisionFOREST),
                                       np.std(precisionFOREST)))
print('--------------------------------------------------------------------------')
```

```
def ingreso_teclado():
    T.delete('1.0', END)
    print("----------------------PREDICCION INGRESO POR TECLADO-----------------------")
    ENTRADA = [txt.get()]
    vector_new    = vectorizer.transform(ENTRADA)
    PREDICCION_SVM_LINEAR = SVMLINEAR.predict(vector_new)
    PREDICCION_ARBOL = ARBOL.predict(vector_new)
    PREDICCION_NAIVEBAYES=NAIVEBAYES.predict(vector_new.toarray())
    PREDICCION_RANDOMFOREST=RANDOMFOREST.predict(vector_new)
    txt.delete(0,END)

    Serio=0
    NoSerio=0


    if PREDICCION_SVM_LINEAR[0] == "Serio":
        Serio=Serio+1
    else:
        NoSerio=NoSerio+1

    if PREDICCION_ARBOL[0] == "Serio":
        Serio=Serio+1
    else:
        NoSerio=NoSerio+1

    if PREDICCION_NAIVEBAYES[0] == "Serio":
        Serio=Serio+1
    else:
        NoSerio=NoSerio+1

    if PREDICCION_RANDOMFOREST[0] == "Serio":
        Serio=Serio+1
    else:
        NoSerio=NoSerio+1



    porcentajeSerio=(Serio*25)
    porcentajeNoSerio=(NoSerio*25)



    for i in range(len(ENTRADA)):

            print("SVM LINEAR: X=%s, Predicted=%s" % (ENTRADA[i], PREDICCION_SVM_LINEAR[0]))
            print("ARBOL DE DESICION: X=%s, Predicted=%s" % (ENTRADA[i], PREDICCION_ARBOL[i]))
            print("NAIVE BAYES: X=%s, Predicted=%s" % (ENTRADA[i], PREDICCION_NAIVEBAYES[i]))
            print("RANDOM FOREST: X=%s, Predicted=%s" % (ENTRADA[i], PREDICCION_RANDOMFOREST[i]))
            print("----------------------------------------------------------")


            texto="La frase '%s'\nfue evaluada por los modelos de Seriedad de la siguiente manera:\n\nSVM kernel
            separador=("\n-------------------RESULTADO-------------------\n")
            texto2="Cantidad de algoritmos que respondieron a Serios:%s " % Serio
            texto3=",el cual corresponde al {:.0f}%\n".format(porcentajeSerio)

            texto4="Cantidad de algoritmos que respondieron a No Serios:%s "% NoSerio
            texto5=",el cual corresponde al {:.0f}%\n".format(porcentajeNoSerio)
            separadorvacio=("\n=============================================================================================


            T.insert(END,texto)
            T.insert(END,separador)
            T.insert(END,texto2)
            T.insert(END,texto3)
            T.insert(END,texto4)
            T.insert(END,texto5)
            T.insert(END,separadorvacio)
```

```
def lectura_txt():
   T.delete('1.0', END)

   filename =  filedialog.askopenfilename(title = "Seleccione el archivo con los comentarios",filetypes = (("Arc

   print(filename)

   archivo = open(filename, "r")
   lectura=archivo.read()

   Nuevas_entradas=lectura.split(sep='\n')

   indice=0
   print("----------------------PREDICCION INGRESO POR ARCHIVO-----------------------")
   while indice < len(Nuevas_entradas):

       ENTRADA = [Nuevas_entradas[indice]]
       vector_new   = vectorizer.transform(ENTRADA)
       PREDICCION_SVM_LINEAR = SVMLINEAR.predict(vector_new)
       PREDICCION_ARBOL = ARBOL.predict(vector_new)
       PREDICCION_NAIVEBAYES=NAIVEBAYES.predict(vector_new.toarray())
       PREDICCION_RANDOMFOREST=RANDOMFOREST.predict(vector_new)
       txt.delete(0,END)

       Serio=0
       NoSerio=0


       if PREDICCION_SVM_LINEAR[0] == "Serio":
           Serio=Serio+1
       else:
           NoSerio=NoSerio+1

       if PREDICCION_ARBOL[0] == "Serio":
           Serio=Serio+1

       else:
           NoSerio=NoSerio+1

       if PREDICCION_NAIVEBAYES[0] == "Serio":
           Serio=Serio+1
       else:
           NoSerio=NoSerio+1

       if PREDICCION_RANDOMFOREST[0] == "Serio":
           Serio=Serio+1
       else:
           NoSerio=NoSerio+1


       porcentajeSerio=(Serio*25)
       porcentajeNoSerio=(NoSerio*25)



       for i in range(len(ENTRADA)):
           print("-----------------------------------------------------------")
           print("SVM LINEAR: X=%s, Predicted=%s" % (ENTRADA[i], PREDICCION_SVM_LINEAR[0]))
           print("ARBOL DE DESICION: X=%s, Predicted=%s" % (ENTRADA[i], PREDICCION_ARBOL[i]))
           print("NAIVE BAYES: X=%s, Predicted=%s" % (ENTRADA[i], PREDICCION_NAIVEBAYES[i]))
           print("RANDOM FOREST: X=%s, Predicted=%s" % (ENTRADA[i], PREDICCION_RANDOMFOREST[i]))
           print("-----------------------------------------------------------")
           texto="\nLa frase '%s'\nfue evaluada por los modelos de Seriedad de la siguiente manera:\n\nSVM kern
           separador=("\n-------------------RESULTADO-------------------\n")
           separadorvacio=("=================================================================================
           texto2="Cantidad de algoritmos que respondieron a Serios:%s " % Serio
           texto3=",el cual corresponde al {:.0f}%\n".format(porcentajeSerio)

           texto4="Cantidad de algoritmos que respondieron a No Serios:%s "% NoSerio
           texto5=",el cual corresponde al {:.0f}%\n\n".format(porcentajeNoSerio)

           T.insert(END,texto)
           T.insert(END,separador)
```

```
            T.insert(END,texto)
            T.insert(END,separador)
            T.insert(END,texto2)
            T.insert(END,texto3)
            T.insert(END,texto4)
            T.insert(END,texto5)
            T.insert(END,separadorvacio)
            T.insert(END,separadorvacio)

        indice=indice+1


def Escritura():
    filename = asksaveasfilename(defaultextension = 'txt',title = "Seleccione donde guardar el informe",filetype

    if filename:
        f = open(filename, 'a')
        SALIDA = [T.get('1.0', END)]
        SALIDAAUX = ''.join(SALIDA)
        f.write(SALIDAAUX)
        f.close()



window = Tk()
window.title("EVALUADOR ")
window.geometry('1090x600')

lbl = Label(window, text="Escriba frase a evaluar:")
lbl.grid(column=1, row=1)

txt = Entry(window,width=70)
txt.grid(column=2, row=1)

btn = Button(window, text="Evaluar", command=ingreso_teclado)
btn.grid(column=3, row=1)


btn2 = Button(window, text="CargaArchivo", command=lectura_txt)
btn2.grid(column=3, row=2)

btn3 = Button(window, text="Guardar Informe", command=Escritura)
btn3.grid(column=3, row=3)

T = Text(window, height=20, width=100)
T.grid(column=2, row=6)

firma = Label(window, text="Clasificador de Seriedad desarrollado por: Nelson Retamal Silva\nAlumno tesista de
firma.grid(column=2, row=7)

imagen=PhotoImage(file="ubb.png")
labelimagen=Label(window,image=imagen).place(x=140,y=430)

imagen2=PhotoImage(file="somosubb.png")
labelimagen2=Label(window,image=imagen2).place(x=610,y=440)

window.mainloop()
```

Ilustración 12: Código aplicación Python