PROPUESTA DE UN SISTEMA PARA EL ANÁLISIS BINARIO DE SENTIMIENTOS DEL LENGUAJE NATURAL EN EL IDIOMA ESPAÑOL

PRESENTADO POR:

HUAMANÍ GONZALES, CARLOS ALBERTO

UNIVERSIDAD CATÓLICA SANTA MARÍA

FACULTAD DE CIENCIAS E INGENIERÍAS FÍSICAS Y FORMALES

PROGRAMA PROFESIONAL DE INGENIERÍA DE SISTEMAS

AREQUIPA

PERU

2015

PRESENTACIÓN

SEÑOR DIRECTOR DEL PROGRAMA PROFESIONAL DE INGENIERÍA DE SISTEMAS DE LA UNIVERSIDAD CATÓLICA SANTA MARÍA

SEÑORES MIEMBROS DEL JURADO EXAMINADOR

De conformidad con las disposiciones del Reglamento de Grados y Títulos del Programa Profesional de Ingeniería de Sistemas, pongo a vuestra consideración el presente trabajo titulado **"Propuesta de un Sistema para el Análisis Binario de Sentimientos del Lenguaje Natural en el Idioma Español",** para optar el Título Profesional de Ingeniero de Sistemas.

Arequipa, octubre de 2015

Carlos Alberto Huamaní Gonzales

DEDICATORIA

Este trabajo está dedicado a Dios por dejarme cumplir con mis proyectos a mi madre y mi hermana quienes me han apoyado en todo el trayecto de esta investigación.

ÍNDICE

ÍNDICE DE TABLAS, FIGURAS Y GRÁFICOS

RESUMEN

El análisis de sentimientos es una técnica muy utilizada actualmente por entidades empresariales a nivel mundial, las cuales ofrecen y demandan productos según la aceptación de una población determinada, para lo cual se debe realizar la medición respectiva.

Se inicia al captar opiniones de potenciales clientes y al procesarlas, establecer que características son favorables de un producto o servicio, así poder incidir en el mejoramiento de la presentación, distribución, costos o productos complementarios con el fin de incrementar la calidad y la rentabilidad.

Los beneficios que brindan estos programas no solo son aplicables al campo industrial o de servicios diversos sino también pueden ser útiles en investigación, educación o aspectos culturales entre otros, ya que permiten la rápida recuperación de información y su respectivo análisis de manera siempre actualizada.

El presente trabajo tiene por objetivo elaborar un sistema capaz de reconocer opiniones vertidas acerca de un elemento u objeto en un contexto determinado dando como resultado un valor aproximado, según la polaridad de una opinión dada.

ABSTRACT

Sentiment analysis is a currently used by business entities worldwide, which offer products according to the demand and acceptance of a given population, for which is due on the respective measurement technique.

It starts by capturing opinions from potential customers and process them, establish that features are favorable for a product or service, and to influence the improvement of the presentation, distribution costs or complementary products in order to increase the quality and profitability.

The benefits provided by these programs are not only applicable to the industrial field or various services, but may also be useful in research, education or cultural aspects among others, allowing rapid retrieval of information and their analysis in a manner continuously updated.

This paper aims to develop a system capable of recognizing opinions expressed about an item or object in a given context resulting in an approximate value, depending on the polarity of an opinion given.

INTRODUCCIÓN

Las opiniones vertidas y captadas a nivel de un grupo experimental para luego ser analizadas y posteriormente extraer resultados de sobre como se mantiene las opiniones subjetivas en una persona puede resultar ser un trabajo demasiado elaborado, puesto que para obtener estos pensamientos subjetivos de este grupo experimental es necesario trabajar con formatos de fácil entendimiento, que permitan obtener puntos claves de un tema específico, dando así mayor complejidad a dichos formatos sumándole además la cantidad de información empapelada que se obtendrá.

El análisis de sentimientos consiste en adaptar y mejorar esta recuperación de información de manera más eficiente, ahorrando tiempo, dinero, análisis, etc. Dado que para la obtención de opiniones o sentimientos se reconoce su patrón o forma de humor de la persona en el momento que la vierte, determinando puntos claves sobre el elemento que opina.

Todas las opiniones son vertidas a una base de datos donde estas están sujetas a un diccionario en el cual se mantiene y considera las palabras potenciales en los temas a tratar, pudiendo tener segmentada la información según su tipo ya sea comercial, investigativo, de entretenimiento, etc.

Estas opiniones ya luego de estar almacenadas son posteriormente analizadas, y en cuanto más grande sea el grupo de opinión vertida, los resultados del tema a investigar serán mejores.

En el primer capítulo se define el planteamiento al problema dado, junto con los objetivos, variables, alcances y limitaciones.

En el segundo capítulo repasamos el estado del arte, para posteriormente introducirnos en el marco teórico donde se definen las bases matemáticas necesaria, y conceptos relevantes con el tema brindado.

En el tercer capítulo se muestra el desarrollo de la propuesta sobre el que se hara el analisis respectivo y temas a considerar para el desarrollo del sistema.

En el cuarto capítulo se considerá el caso de estudio donde se tomara en cuenta los pasos para obtener resutlados sobre un tema dado.

En el quinto capítulo se realizan las evaluaciones necesarias para determinar la eficiencia y características del análisis de sentimientos según los criterios propuestos.

CAPÍTULO 1. PLANTEAMIENTO TEÓRICO

1.1 Título Descriptivo del Proyecto

Propuesta de un Sistema para el Análisis Binario de Sentimientos del Lenguaje Natural en el Idioma Español.

1.2 Descripción del Problema

Con los nuevos enfoques de inteligencia artificial, han aparecido nuevos campos de investigación en interacción de humano con la maquina, uno de los cuales es el Procesamiento del Lenguaje Natural, el cual tiene una parte llamada Minería de Opinión que a su vez tiene una tarea llamada Análisis de Sentimientos (Javi Fernandez, Ester Boldrini, Jose Gómez y Patricio Martinez, 2011).

El análisis de sentimientos tiene como objetivo identificar el sentimiento, opinión o sensación de los usuarios frente a algo como puede ser productos, empresas, lugares, personas y otros; basándose en los contenidos publicados en la web y la propia experiencia.

Mediante la aplicación del sistema adecuado, es posible obtener un informe completo que incluye un resumen de lo que un público percibe sobre un elemento sin la necesidad de buscar y leer todas las opiniones y noticias relacionadas a ella y así poder obtener en números, gráficos, etc. la situación de lo que busca un solicitante (Javi Fernandez, et. al, 2011).

.

En el área del aprendizaje de máquinas, el análisis de sentimientos es un problema de categorización de texto que detecta la polaridad de las opiniones relacionadas con un tema específico, siendo importante la identificación de esos sentimientos u opiniones positivas o negativos (H. Cordobes, A. Fernandez, L. Nuñez, F. Perez, T. Redondo, A. Santos, 2013) .

En el mundo existen numerosas herramientas de detección de sentimientos en idiomas diferentes, de los cuales el idioma inglés es el que abarca numerosas aplicaciones, no siendo compatible con el idioma español debido a su variación sintáctica, semántica y de significado y por tanto no produce los resultados esperados frente a una oración.

Es importante el desarrollo de un software que analice las opiniones en idioma español, a fín de fortalecer las investigaciones de inteligencia artificial en nuestro medio, mejorando la actividad comercial, educativa, y de diversos servicios y en un futuro la interacción de humanos con máquinas.

1.3 Área Científica a la que corresponde el problema

Área: Inteligencia artificial.

Línea: Inteligencia artificial aplicada a la inteligencia de negocios.

1.4 Área Investigativa a la que corresponde el problema.

Tipo de Investigación: Aplicada.

Nivel de Investigación: Experimental.

1.5 Objetivos

1.5.1. Objetivo General

- Implementar un sistema de análisis de sentimientos que detecte la polaridad o sentido correcto de opiniones brindadas en contextos sociales para la correcta detección y clasificación.

1.5.2. Objetivos Específicos

1. Implementar una base de términos léxicos usuales en el lenguaje natural español clasificada por sus tipos semánticos.
2. Elaborar el sistema de análisis de oraciones aplicable a la subjetividad, integridad, veracidad y polaridad de las opiniones.
3. Probar el sistema de análisis de sentimientos en un caso de estudio.

1.6 Criterios

La polaridad dada por el sistema de análisis de sentimientos será medida por los siguientes criterios:

1.6.1. Criterios de la variable dependiente.

CRITERIOS	LO QUE ASUME	METODOS ANALITICOS
INTEGRIDAD DEL MENSAJE	Las palabras o expresiones lingüísticas son identificadas por el sistema de acuerdo a su naturaleza.	POS (Part Of Speech) Tagging: Etiqueta las palabras segmentando la oración o expresión, dando lugar a los n-gramas.
PONDERABILIDAD	Número de veces en que se repiten las palabras en las oraciones.	Term Frecuency: Realiza un conteo de palabra en un total de frases u oraciones.
SUBJETIVIDAD	Expresa los sentimientos en las oraciones brindadas por sujetos.	Teorema de Bayes: Detecta la polaridad delas opiniones.
VERACIDAD	Expresa la realidad del sentido del mensaje.	Indicadores de eficiencia de los algoritmos: Índice de Precisión (P), Índice de Recuperación (R), Índice de eficiencia(F).

Cuadro basado en **Hernández M.; Gómez J. -** Análisis de Sentimientos Aplicado a Referencias Bibliográficas

1.7 Justificación del estudio.

Existe la necesidad de contar con técnicas de evaluación de opiniones vertidas en la web en relación a instituciones, productos o servicios para obtener de manera automatizada la graduación de la aceptación o rechazo de estos, aprovechando lo genuino de la subjetividad vertida en las redes sociales.

El estudio permite procesar diversas situaciones en un sin número de mercados posibles, que muestre una evaluación directa del sentir de gente real en forma verídica, en cuanto a la preferencia de ciertos productos o bienes, tecnologías o métodos más aceptados, reconocimiento de la seguridad de lugares, referéndums virtuales, etc.

Se requiere desarrollar la evaluación de las opiniones mediante métodos clasificatorios tal como el Teorema de Bayes con el objetivo de lograr un grupo de valores los cuales son sometidos al análisis de patrones y probabilidades, que muestren la graduación de estas.

Para esto se utilizará un diccionario de palabras, teniendo una clasificación de positivo o negativo con las cuales será entrenado el sistema y posteriormente se podrá realizar el reconocimiento de las oraciones.

La función de los métodos estaría dada por la segmentación de clases de un grupo universal, de las cuales se introduce un nuevo patrón a comparar, y mediante cercanía se determina a cual grupo pertenece el nuevo objeto.

Los resultados de esta investigación otorgaran una mayor visibilidad y beneficios de como es aplicable la propuesta en el mercado, al evaluar algún producto dado.

1.8 Alcances y Limitaciones.

1.7.1 Alcances Internos

Obtención de resultados verídicos a partir de opiniones espontáneas de usuarios en la web.

1.7.2 Alcances Externos

Propuesta de un sistema de análisis de opinión de los usuarios para poder conocer la situación o estatus de productos, servicios u otros en el mercado o sociedad.

1.7.3 Limitaciones

- Hardware del ordenador
- Tiempo de desarrollo de software
- Análisis en texto correctamente escrito
- Traducción e ingreso de las palabras del diccionario
- Uso de jergas

CAPÍTULO 2. MARCO TEÓRICO

2.1. Estado del Arte.

- Bravo, Mendoza y Poblete (2013) explican en su investigación que el análisis de sentimiento en Twitter o automatización de tareas en la recuperación de opiniones de Tweets ha tenido un creciente interés de la comunidad de minería de la web. Esto es debido a una gran importación de campos de estudios tales como son los negocios y la política. La gente expresa sentimientos acerca de temas específicos o entidades con diferente fuerza e intensidad, donde estos sentimientos son fuertemente relacionados a las personas que tienen estos sentimientos u opiniones.

  Un número de métodos y recursos léxicos han sido propuestos para el análisis de sentimientos desde textos de lenguaje natural, direccionando diferentes dimensiones de opiniones.

  En su artículo proponen un enfoque para impulsar la clasificación de sentimientos en Twitter usando diferentes dimensiones de sentimientos como características meta niveles.

  Donde combinan aspectos tales como fuerza de la opinión, emoción e indicadores de la polaridad, generados por métodos y recursos de análisis de sentimientos existentes, mostrando que la combinación de sentimientos en dimensiones provee una mejora significativa en las tareas de clasificación de sentimientos en Twitter tales como polaridad y subjetividad.

  Tomando en cuenta estos aspectos, en esta propuesta se procederá a definir metas y límites donde se podrá realizar el análisis correspondiente, tales como

sería el análisis subjetivo determinando la polaridad en forma negativa y positiva.

- Bal Krishna Bal (2009) en su investigación hace conocimiento que las editoriales representan las opiniones de los artículos escritos por un publicista, editor o columnista de un periódico. Desde esta perspectiva, las editoriales son recursos de puntos de vistas o eventos para ser analizados, como son percibidos.

  El trabajo propuesto tiene por objeto determinar los criterios lingüísticos, en la creación de un modelo para el referido fin, que hace uso de un conjunto adecuado de las etiquetas semánticas que se definieron para anotar los textos a partir del cual, puede llevarse a cabo un detallado análisis.

  El trabajo se centró principalmente en la definición de las variables necesarias para el análisis semántico-pragmático, en paralelo con la definición de las etiquetas y anotaciones de texto, donde han explorado maneras de construir la síntesis de opiniones sobre un determinado evento de varias editoriales.

  Uno de los retos que se plantearon consiste en organizar los puntos de vista positivos y negativos, y la de los argumentos asociados y su fuerza.

  Han tomado nota de que las opiniones en los editoriales no son tan evidentes, lo que hace que la construcción de la síntesis sea una tarea difícil.

  De modo que en la propuesta del sistema se tomará en cuenta el modo de etiquetado de la palabra como así en la forma de identificación de la subjetividad complementadose con la referencia anterior.

- Alexander Pak y Patrick Paroubek (2010), afirman en su investigación que el Microblogging actualmente es una popular herramienta de comunicación entre los usuarios de internet. Millones de usuarios buscan opiniones de diferentes aspectos de vida diaria. Entonces los sitios web de microblogging son recursos ricos en información para la minería de opinión y el análisis de sentimientos.

  En su artículo, se enfocan en el uso de Twitter, la plataforma más popular de microblogging, para las tareas de análisis de sentimientos.

  Muestran como automáticamente se colecciona un corpus para propósitos del análisis de sentimientos y minería de opinión, realizan análisis lingüístico de corpus, y explican sus fenómenos descubiertos.

  Usando el corpus, se puede construir un clasificador de sentimientos, que permite determinar los sentimientos positivos, negativos o neutros en un documento.

  También afirma en sus evaluaciones experimentales el propósito de las técnicas usadas son eficientes con rendimiento mejorado que otros métodos previamente propuestos. El trabajo realizado está basado en el idioma ingles y según el autor puede ser utilizado esta propuesta en otro idioma.

  La propuesta del sistema a implementar, tendrá como objetivo almacenar la mayor cantidad posible de palabras en el corpus previamente etiquetadas según su tipo, lo cual servirá de base para el sistema.

.

- Theresa Wilson, Janyce Wiebe y Paul Hoffman (2005), afirman en su trabajo un nuevo enfoque para determinar el análisis de sentimiento según frase y niveles, donde primero determina si una expresión es neutral o polar y desambigua la

polaridad de las expresiones polares. Con este enfoque es permitido identificar automáticamente la polaridad contextual para un conjunto largo de expresiones de sentimientos, logrando resultados significativamente mejores que un punto inicial al analizar la polaridad contextual.

- Para elaborar el sistema se considerará una parte los artículos mencionados, generando la idea principal de como trabaja el sistema propuesto de análisis de sentimientos, se armará la propuesta del sistema teniendo en cuenta la polaridad, sus límites, la forma en que consideran el POS y el almacenamiento de las palabras para su correcto filtrado, y posteriormente el análisis que les otrogan resultados favorables.

2.2. Marco conceptual

2.2.1. Análisis de Sentimientos

El análisis automático de criterios subjetivos presentes en un texto se conoce como Análisis de Sentimientos (AS), es un tema actual de investigación en el área de procesamiento del lenguaje natural en el campo de extracción de la información; su campo de aplicación es muy amplio en el monitoreo de emociones en disciplinas tan diversas como el marketing, ciencias políticas y economía (Hernandez M. y GomezJ., 2014).

2.2.1.1. Conjuntos

Un conjunto es una colección de objetos o elementos, ya sean números, letras, ciudades, etc.

Se pueden realizar operaciones con conjuntos, como la unión, la intersección y el complemento ya sea por combinación, determinación de elementos comunes o por los elementos que constituyen el conjunto universal.

2.2.1.2. Espacio Muestral

Es el conjunto compuesto por todos los resultados posibles de un experimento. A cada elemento de un espacio muestral se le llama punto muestral o evento simple.

2.2.1.3. Evento simple

Es un subconjunto de un espacio muestral cuando aparece el resultado requerido.

2.2.1.4. Probabilidad del evento simple

Es la ponderación que mide la posibilidad de ocurrencia de un evento simple, cumpliendo dos reglas: la probabilidad que se asigna a cualquier evento simple varía de 0 a 1 y la suma de las probabilidades asignadas a todos los eventos simples del espacio muestral debe ser igual a 1.

Una ponderación o probabilidad próxima a 0 es cuando la ocurrencia de un evento simple no es probable, y si es próxima a 1 su ocurrencia es probable.

2.2.1.5. Probabilidad Condicional

Es la probabilidad de que ocurra un evento *A*, sabiendo que también sucede otro evento *B*. La probabilidad condicional se escribe *P*(*A*|*B*), y se lee "la probabilidad de *A* dado *B*".

$$P(B|A) = \frac{P(A \cap B)}{P(A)}$$

2.2.1.6. Teorema de Bayes

La ecuación de Bayes constituye la base de todos los sistemas modernos de inteligencia artificial para inferencia probabilista.

La técnica de Bayes permite abordar en forma diferente el área de toma de decisiones, dando varios resultados del cual se podría obtener la mejor y peor probabilidad (S. B. Kotsiantis, 2007).

2.2.2. Robustez

Capacidad de los sistemas software de reaccionar apropiadamente ante condiciones excepcionales (Cristina Cachero y Pedro J. Pomce de Leon, 2011).

2.2.3. Algoritmo supervisado

El Aprendizaje supervisado es la búsqueda de algoritmos que suplementen externamente a instancias que producen ciertas hipótesis, con los cuales podemos hacer predicciones sobre futuras instancias. En otras palabras, el objetivo del aprendizaje supervisado es construir un

modelo conciso de distribución de clases de etiquetas en términos de las características de una predicción (S. B. Kotsiantis et. al, 2007).

2.2.4. Minería de opinión

Dado un conjunto de documentos de texto a evaluar D que contienen opiniones (o sentimientos) acerca de un objeto, la minería de opinión tiene por objetivo extraer atributos y componentes de un objeto que ha sido comentado en cada documento d ∈ D y determinando si los comentarios son positivos, negativos o neutros donde las características de la minería de opinión vendrían a ser (Bing Liu, 2007):

- Identificar las características del objeto.
- Determinar la orientación de la opinión.
- Agrupar sinónimos.

2.2.5. Sentimientos y subjetividad

Un estado positivo o negativo que muestra el momento actual de una persona (Alec Go, Richa Brayani, Lei Huang, 2009).

2.2.6. Lenguaje Natural

Lengua o idioma hablado o escrito por humanos para propósitos generales de comunicación. Por ejemplo, en el diálogo se usan mucho las oraciones incompletas o hasta recortadas a una sola palabra (como “ajá”, “pues”), las cuales no contempla un lenguaje formal (Alexander Gelbukh, 2010).

2.2.7. Diccionario de palabras o corpus

El diccionario contribuye al desarrollo de las claves estructurales de la información para garantizar una precisión extremadamente alta del significado de los mensajes recuperados en las redes sociales, por lo tanto es necesario un número masivo de términos o palabras (Nobuhiro Kaji y Masaru Kitsuregawa, 2010).

Para realizar el análisis de sentimientos es fundamental el reconocimiento de la polaridad de las palabras teniendo en cuenta el enfoque del aprendizaje semi supervisado y no supervisado con palabras etiquetadas y no etiquetadas respectivamente (S. B. Kotsiantis et. al, 2007).

El diccionario contiene los tipos de elementos de una oración como adjetivos, verbos y adverbios que son los que describen el sentido de los acontecimientos.

El lenguaje es el puente hacia la realidad, ya que las maneras que describimos los eventos definen los significados de los acontecimientos y estos significados nos ayudan a mantener contacto con la realidad donde estos eventos pueden ser opiniones o hechos (Janes W. Pennebaker, Matthias R. Mehl y Kate G. Niederhoffer, 2003).

Las opiniones a menudo expresan una actitud hacia algo. Esto puede ser un juicio, una opinión o una conclusión o incluso una opinión acerca de una opinión, en estos casos los adverbios o adjetivos captan mejor la polaridad de las expresiones (Bal Krishna Bal, 2009).

Los hechos son característicos de la presencia de ciertos verbos, de diferentes formas de tiempo y número del verbo que proporciona la evidencia de la afirmación.

2.2.8. Bag of Words

Bag of Words es también considerado como el repositorio donde se tomara en cuenta las palabras y oraciones almacenadas.

El modelo Bag of Words es una representación simplificada usada en el procesamiento del lenguaje natural (PLN) y en la obtención de información ó Information Retrieval (IR). En este modelo un texto (tal como puede ser una oración o un documento) es representado como un conjunto o multiconjunto de palabras, sin tener en cuenta su sentido gramático (Yin Zhang, Rong Jin y Zhi-Hua Zhou, 2010).

En el sistema propuesto el Bag of Words vendría a conformar todo lo que es la base de datos o Corpus y una función que se encarga del filtrado de las palabras principales evitando tener en consideración las palabras como artículos, pronombre, conjunciones y preposiciones dado que estas palabras no denotan significado alguno para la oración o no tienen relevancia en una oración y su repetición de estas puede ser demasiada pudiendo interferir en las clases de análisis positivas y negativas y como resultado se tendría soluciones erróneas (Margaret M. Bradley and Peter J. Lang, 1999).

2.2.9. N-Gramas o Cadenas de Markov

El más simple modelo de N-gramas es la estimación de probabilidad máxima (MLE), que tiene la probabilidad de una palabra Wn, dado un contexto anterior W1. . . W(n-1), al ser la relación entre el número de ocurrencias en un corpus de entrenamiento de N-gramas de W1. . . Wn para el número total de ocurrencias de cualquier palabra en el mismo contexto (Robert C. Moore y Chris Quirk, 2009).

Dicho de otra manera los N-gramas vienen a ser un conjunto de palabras tanto de una palabra como dos o tres palabras que tienen la finalidad de lograr un ocurrencia de estas en un corpus dado, generando un probabilidad entre más n-gramas sean identificados su probabilidad difiere o es mayor a cero, caso contrario si el n-grama no existe en el corpus (Hanna M. Wallach, 2006).

El N-grama tanto se aplica como para el corpus como para la oración a ser analizada, en el sistema propuesto ya habiendo descompuesto las oraciones en N-gramas lo que queda es hacer que el sistema descomponga la oración a ser analizada en gramas, y posteriormente su comparación.

Un n-grama esta compuestos por varios tipos de gramas ya sea un grama, dos gramas, o tres gramas ejemplo.

Oración: “la música es el bienestar del alma y cuerpo”

3 gramas: {“la música es”, “música es el”, “es el bienestar”, “el bienestar del”, “bienestar del alma”, “del alma y”, “alma y cuerpo”}

2 gramas: {“la música”, “música es”, “es el”, “el bienestar”, “bienestar del”, “del alma”, “alma y”, “y cuerpo”}

1 grama: {“la”, “música”, “es”, “el”, “bienestar”, “del”, “alma”, “y”, “cuerpo”}

Posterior a haber realizado la descomposición de la oración en gramas se procede a ver su probabilidad y frecuencia de cada termino (TF) en el corpus que será mencionado más adelante.

2.2.10. Term Frecuency o Frecuencia de Términos

Es un peso usualmente usado en recuperación de información y minería de datos. Este peso es una medida estadística que se utiliza para evaluar la importancia de una palabra en un documento en una colección o corpus. La importancia aumenta en proporción al número de veces que una palabra aparece en el documento, pero se compensa con la frecuencia de la palabra en el de corpus. El algoritmo de Tf-idf se utiliza a menudo en el motor de búsqueda, la extracción de datos web, computación similitud de texto y otras aplicaciones. Estas aplicaciones a menudo se enfrentan con el procesamiento masivo de datos (Bin Li y Yuan Guoyong, 2012).

Es decir la frecuencia de cada término encontrado en el corpus es analizada y brinda la posibilidad de calcular la probabilidad en que cada palabra puede aparecer a lo largo de todo un documento o conjunto de oraciones.

Ejemplo en un corpus de 1000 oraciones podemos encontrar las siguientes frecuencias en la oración “No me sentí mal”:

PALABRA	FRECUENCIA
No	20
Me	50
Sentí	30
Mal	15

Fuente: Elaboración Propia

Para tener en consideración un mejor filtrado de las palabras más resaltantes o palabras claves no se considerara el conteo de Pronombres, artículos, conjunciones y preposiciones.

2.2.11. Part Of Speech (Tagging)

Part of Speech consiste en asignar un identificador a cada palabra en una oración (Eugene Charniak).

Por ejemplo:

PRONOMBRE	VERBO	PREPOSICION	ARTICULO	SUSTANTIVO
ELLA	AMA	A	LOS	ANIMALES

Fuente: Elaboración Propia

Con esta identificación en el sistema propuesto se reconoce las palabras más importantes, y se puede eliminar palabras que al momento de la

clasificación influirán de una manera muy negativa en lo que es procesamiento y asi poder identificar la polaridad más rápidamente (Peter D. Turney, 2002).

En la figura 4 se muestra donde el algoritmo consulta con el corpus la palabra si es de la categoría enunciada, se comprueba que esta siga las reglas según la gramática del español y posteriormente se etiqueta.

2.2.12. Teorema de Bayes

El teorema de Bayes usado es la versión General, donde Bayes es un resultado en un teoria de probabilidad que relata condiciones probabilísticas. Sea A y B denota dos eventos, donde P(A|B) denota la probabilidad condicional de A ocurriendo dado que B ocurrió. Las dos probabilidades condicionales P(A|B) y P(B|A) son diferentes. El Teorema de Bayes da una relación entre P(A|B) y P(B|A).

Una aplicación importante del Teorema de Bayes es que da una regla de la forma de actualizar o revisar las fortalezas basadas en la evidencia que da a la luz a nuevas pruebas a posteriori.

Como un teorema formal, el teorema de Bayes es válido en todas las interpretaciones de la probabilidad. Sin embargo, desempeña un papel central en el debate sobre los fundamentos de la estadística: interpretaciones de frecuencia y bayesianas que están de acuerdo sobre el

tipo de cosas a las que las probabilidades deben ser asignados en las aplicaciones.

Mientras las frecuentistas asignan probabilidades a sucesos aleatorios según sus frecuencias de ocurrencia o para subconjuntos de poblaciones como proporciones del conjunto, los bayesianos asignan probabilidades a las proposiciones que son inciertos. Una consecuencia es que los bayesianos tienen ocasión más frecuente de utilizar el Teorema de Bayes. Los artículos sobre la probabilidad bayesiana y probabilidad frecuentista discuten estos debates con mayor extensión (Stuart Russell and Peter Norvig, 1999).

El teorema de Bayes relaciona las probabilidades condicionales y marginales de eventos estocásticos A y B.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Donde cada término en el Teorema de Bayes tiene un nombre convencional:

- P(A) es la probabilidad a priori o probabilidad marginal de A. Es es a priori en el sentido que esto no toma en cuenta cualquier información de B.
- P(A|B) es la probabilidad condicional de A, dado B. También es llamado la probabilidad aposteriori, porque se deriva de o depende del valor especificado de B.
- P(B|A) es la probabilidad de B, dado A.

- P(B) es la probabilidad a priori o marginal, y actua como una normalización constante (Stuart Russell et. al, 1999).

Aplicando el Teorema de Bayes a nuestra aplicación denotamos por A y B como nuestro conjunto de oraciones o de palabras (A) dado que la polaridad a ser detectada sea (B).

Donde la formula seria:

$$P(o|p) = \frac{P(p|o)P(\mathrm{o})}{P(\mathrm{p})}$$

- P(o) es la probabilidad de oraciones tanto positivas como negativas.
- P(p|o) probabilidades de las frecuencias de las palabras de las oraciones según clase ya sea positiva o negativa.
- P(p) es la suma de las probabilidades de ambas clases
- Y la fórmula de Bayes se ejecuta en ambas clases tanto para positivas como para negativas.

2.2.13. Metodología Scrum

La metodología Scrum de desarrollo de software ágil marca una salida dramática de la gestión en cascada. De hecho, Scrum y otros procesos ágiles fueron inspirados por sus deficiencias. La metodología Scrum hace hincapié en la comunicación y la colaboración, el software de funcionamiento y la flexibilidad para adaptarse a las nuevas realidades de

negocios y todos los atributos que sufren en el paradigma de la cascada (Danube, 2010)

.

3. DESARROLLO DE LA PROPUESTA

Para el desarrollo de la aplicación propuesta se tuvo en cuenta aspectos de análisis y desarrollo del software, previa planificación con una estructuración de tiempo determinada de los cuales se detallaran a continuación.

3.1. Esquema

El esquema mostrado resume la manera en como interactua el sistema desarrollado, donde para poder realizar un análisis de sentimientos es tanto necesario el corpus, los modulos de POS, Bag of words y el teorema de bayes, generando una sinergia entre ellos.

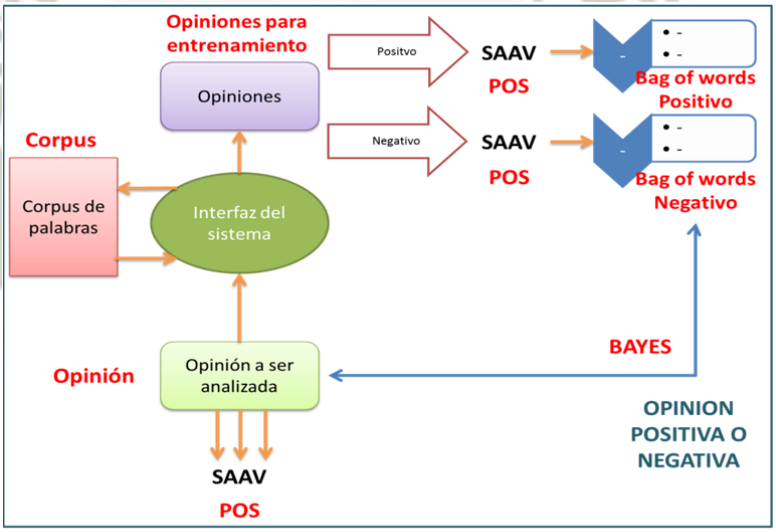


Figura 1.- Esquema interacción sistema de análisis de sentimientos basado en Kunpeng Zhang, Yu Cheng, Yusheng Xie, Daniel Honbo (2014)

El sistema (Figura 1) contará con una interfaz la cual interactuara con el corpus, las opiniones de entrenamiento, y la opinion de prueba, donde se describe a continuación el proceso, el cual es considerado y esquematizado de los artículos mencionados en el estado del arte:

1. La interfaz procesa y consulta las opiniones de entrenamiento.
    a. Separa cada opinión de entrenamiento en gramas.
    b. Luego de separar aplica POS para identificar los términos de cada grama según polaridad de opinión.
    c. Almacena y separa las palabras en dos listas, llamadas bag of words una de opiniones postivas y otra de negativas.
2. La interfaz por cada consulta y proceso con las opiniones ingresa al corpus para determinar la existencia.
3. La interfaz procesa la opinión almacenada
    a. Separa la opinión almacenada en gramas.
    b. Indentifica el POS de los gramas.
4. Teniendo separado la opinión ingresada en gramas e identificadas por POS y también teniendo los bag of words negativos y positivos, se procede a realizar el análisis correspondiente con el teorema de bayes el que arrojara un resultado aproximado.

3.2. Aplicación de la metodología

SCRUM es la metodología idónea para el desarrollo del software propuesto, al carecer de equipo de trabajo se realizará una modificación de la metodología SCRUM, el cual será la metodología SCRUM Personal, para la construcción del

software donde se realizará el desarrollo respectivo, tomando en cuenta la gestión y priorización de requisitos, la implementación y pruebas, depuración del sistema y buen funcionamiento. (John Pruitt, 2011)

3.2.1. Gestión de requisitos

- Prototipo de requisitos iniciales
    - Corpus
    - Sistema de análisis de sentimientos
    - Colección de sentimientos
- Priorizacion de requisitos
    1. Corpus
    2. Colección de sentimientos
    3. Sistema de análisis de sentimientos

3.2.2. Requisitos a implementar

Requisitos implementados según necesidad inmediata.

- Recolección de corpus
- Implementación de sistema de análisis de sentimientos
    - Interfaz.
        - Desarrollo de n-gramas.
        - Desarrollo de POS.
        - Consulta de palabras de base de datos y determinación del bag of words.
        - Term Frequency.
        - Bayes.

- Análisis con opiniones experimentales y determinación de éxito.
- Planificación
    - Diagrama de Gantt.
    - Esquematización del proyecto según casos de uso, clases, paquetes.

3.2.3. Preguntas de autoanálisis

- ¿Qué se hizo ayer?
- ¿Qué falta hacer?
- ¿Se encontró algún problema inesperado?

3.2.4. Entrega del sistema

- Incremento

  Según necesidad verificada se añade ciertos módulos mejorando la funcionalidad del sistema.
- Incidencias resueltas
    - Codificación del sistema.
    - Recolección datos tanto palabras como opiniones
    - Pruebas con Bayes.

3.3. Casos de uso

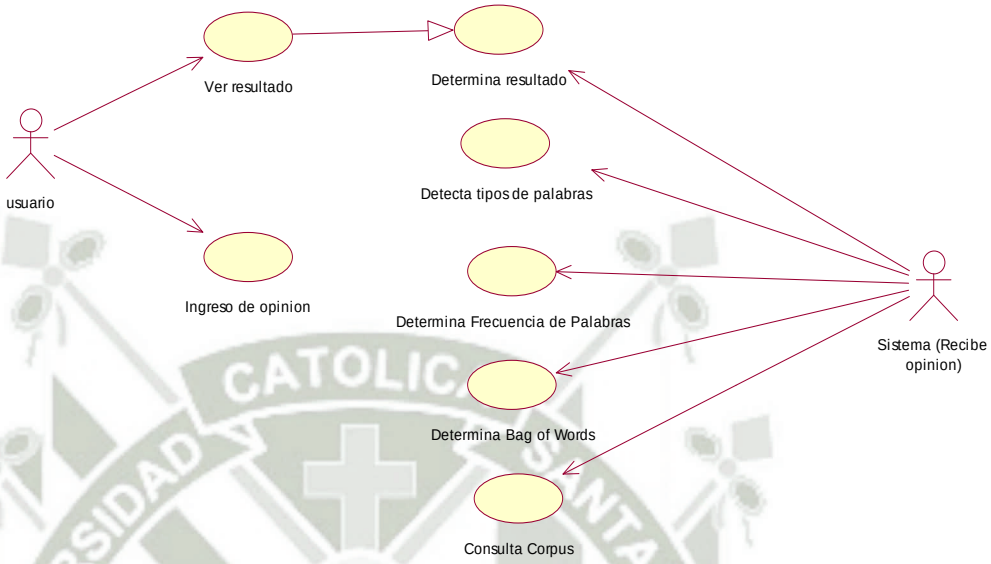


Figura 2.- Caso de uso interacción sistema

Fuente: Elaboración propia

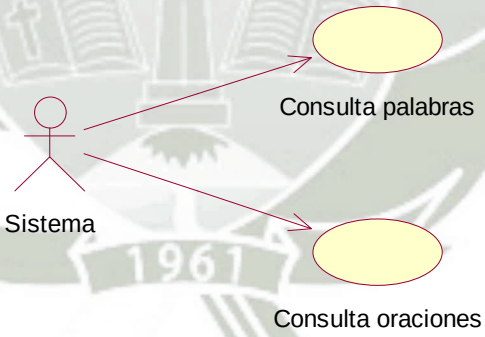


Figura 3.- Caso de uso Corpus

Fuente: Elaboración propia

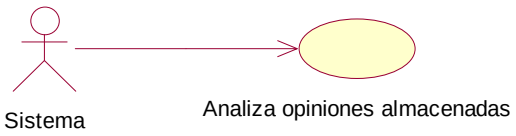


Figura 4.- Caso de uso Entrenamiento con opinioness

Fuente: Elaboración propia

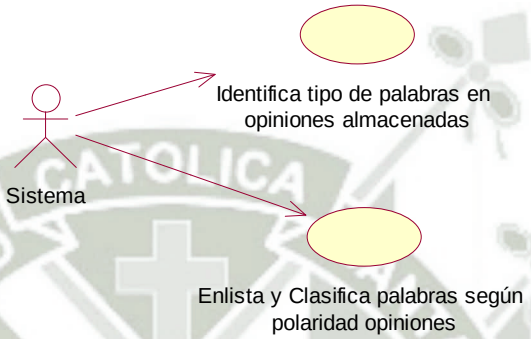


Figura 5.- Caso de uso POS y Bag of Words

Fuente: Elaboración propia

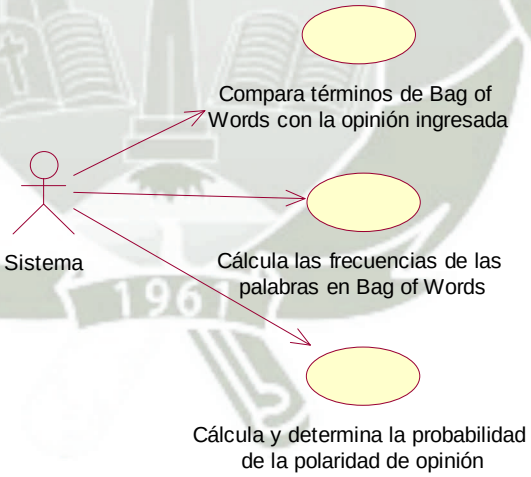


Figura 6.- Caso de uso Bayes

Fuente: Elaboración propia

3.3.1. Descripción de casos de uso interacción de sistema figura 2.

Nombre	Ingreso de opinión-CU1	
Actores	Usuario	
Función	Ingresar la opinon al sistema para ser analizada	
Descripción	El usuario será la persona encargada de ingresar las opiniones al sistema donde posteriormente estas serán analizadas, según lo especificado.	
Flujo Principal:	**EVENTO ACTOR**	**EVENTO SISTEMA**
	1.- Activa el sistema para analizar opinión.	1.- Invoca los métodos de análisis de opinión.
Alternativa:	1.- Activa el sistema para analizar opinión.	1.- Invoca los métodos de análisis de opinión.
		2.-Verifica error de existencia de opinión a ser analizada
Precondición:	El usuario habilita el sistema de análisis de sentimientos, para realizar la clasificación.	
Poscondición:	El sistema muestra en pantalla un resultado aproximado	

Fuente: Elaboración Propia

Nombre	Ver resultado –CU2	
Actores	Usuario	
Función	Verificar resultado de la opinión.	
Descripción	El usuario será capaz de verificar y contrastar los resultados arrojados por el sistema.	
Flujo Principal:	**EVENTO ACTOR**	**EVENTO SISTEMA**
	1.- Verifica resultado en pantalla	1.- Muestra resultado procesado por sistema en pantalla
Alternativa:	1.- Verifica resultado en pantalla	1.- Muestra resultado procesado por sistema en pantalla
	2.- Muestra un resultado desconocido	2.- Ocurrio error de procesamiento de la opinión existente.

Precondición:	El usuario ingreso y permitió que el sistema de análisis de sentimientos procesara la opinión.
Poscondición:	El sistema muestra en pantalla un resultado aproximado

Fuente: Elaboración Propia

Nombre	Consulta corpus –CU3	
Actores	Sistema	
Función	Verificar el corpus contenido	
Descripción	Al momento de ingresar y someter la opinión a analisis el primer paso será la constrastación de las palabras de la opinión con las que se tienen en el corpus.	
Flujo Principal:	**EVENTO ACTOR**	**EVENTO SISTEMA**
	1.- Invoca la función para consultar las palabras de la opinión a ser analizada.	1.- Separa la opinión en gramas
	2.-	2.-Compara con la base de datos y comptrueba la existencia de las palabras identificadas en la BD en el corpus
	3.- Recibe conformidad de analisis con corpus.	
Alternativa:	1.- Invoca la función para consultar las palabras de la opinión a ser analizada.	1.- Separa la opinión en gramas
		2.- Muestra error y envía a pantalla de usuario palabras no existentes en la BD
Precondición:	El sistema separó la opinión en gramas.	
Poscondición:	El sistema detectó correctamente todos los gramas analizados.	

Fuente: Elaboración Propia

Nombre	Detecta tipo de palabras – CU4	
Actores	Sistema	
Función	Identificar los tipos de palabras contenidas en la opinión	
Descripción	Posterior a ser comparado con el corpus, el sistema detectara los tipos de palabras contenidas en la opinión.	
Flujo Principal:	**EVENTO ACTOR**	**EVENTO SISTEMA**
	1.- Invoca la función para detectar el tipo de palabras contenidas en la opinión.	1.- Los gramas separados anteriormente los compara la BD
	2.-	2.-Determina el tipo de palabra de cada grama analizado
	3.- Recibe conformidad de analisis y verifica etiquetado de palabras.	
Alternativa:	1.- Invoca la función para detectar el tipo de palabras contenidas en la opinión.	1.- Los gramas separados anteriormente los compara la BD
		2.- Muestra error y ejecuta excepción que no existe tipo de palabra definido
Precondición:	El sistema separó la opinión en gramas y verifico la existencia de estas.	
Poscondición:	El sistema detectó correctamente los tipos de palabras de cada grama.	

Fuente: Elaboración Propia

Nombre	Determina Bag of Words – CU5
Actores	Sistema
Función	Clasificar los bag of words
Descripción	El sistema al haber detectado y clasificado la opinión según los tipos de palabras, procederá a descomponer de la misma manera al grupo de opiniones ya clasificadas y almacenadas logrando obtener un conjunto de palabras

	tanto como para opiniones positivas y negativas donde se obtendrá una especie de corpus pero solo de las opiniones ya analizadas.	
Flujo Principal:	**EVENTO ACTOR**	**EVENTO SISTEMA**
	1.- Invoca la función para procesar las opiniones de entrenamiento.	1.- Separa en gramas y verifica la existencia de cada palabra.
	2.-Obtiene existencia e invoca función de etiquetado de palbras	2.-Etiqueta los gramas según tipo de palabra
	3.- Recibe conformidad de analisis y verifica etiquetado de palabras.	3.- El sistema almacena cada palabra detectada en dos listas una positiva y otra negativa según la polaridad de cada opinión de entrenamiento previamente clasificada.
Alternativa:	1.- Invoca la función para detectar el tipo de palabras contenidas en la opinión.	1.- Los gramas separados anteriormente los compara la BD
		2.- Muestra error y ejecuta excepción que no existe tipo de palabra y realiza la petición de ingreso con su tipo de palabra.
Precondición:	El sistema debió haber realizado el analisis a la opinión ingresada.	
Poscondición:	El sistema almaceno y archivo ambas listas (positivo y negativo) de palabras como archivos externos considerándose el bag of words que se utilizara mas adelante.	

Fuente: Elaboración Propia

Nombre	Determina frecuencia de palabras – CU6
Actores	Sistema
Función	Realizar un conteo de las palabras y su presencia en las opiniones
Descripción	Al mismo tiempo que esta analizando bag of words se

	determina la frecuencia de cada palabra en cuanto a su repetición en las opiniones analizadas, obteniendo un valor, el cual es considerado la frecuencia de una palabra que influyen en una opinion positiva o negativa.	
Flujo Principal:	**EVENTO ACTOR**	**EVENTO SISTEMA**
	1.- Invoca la función para procesar las opiniones de entrenamiento.	1.- Separa en gramas y verifica la existencia de cada palabra.
	2.-Obtiene existencia e invoca función de etiquetado de palbras	2.-Etiqueta los gramas según tipo de palabra
	3.- Recibe conformidad de analisis y verifica etiquetado de palabras.	3.- El sistema almacena cada palabra detectada en dos listas una positiva y otra negativa según la polaridad de cada opinión de entrenamiento previamente clasificada.
Alternativa:	1.- Invoca la función para detectar el tipo de palabras contenidas en la opinión.	1.- Los gramas separados anteriormente los compara la BD
		2.- Muestra error y ejecuta excepción que no existe tipo de palabra y realiza la petición de ingreso con su tipo de palabra.
Precondición:	El sistema debió haber realizado el analisis a la opinión ingresada.	
Poscondición:	El sistema almaceno y archivo ambas listas (positivo y negativo) de palabras como archivos externos considerándose el bag of words que se utilizara mas adelante.	

Fuente: Elaboración Propia

Nombre	Determina resultado – CU7
Actores	Sistema
Función	Ejecuta el análisis de bayes y arroja un resultado según cercanía.

Descripción	El Teorema de Bayes, arroja un resultado de identificación de la opinión y esta le es brindada al usuario.	
Flujo Principal:	**EVENTO ACTOR**	**EVENTO SISTEMA**
	1.- Invoca la función para obtener resultados del analisis.	1.- Invoca resultados de la función que lo genera.
	2.-Muestra en pantalla el resultado de la opinión analizada ya sea positivo o negativo.	
Alternativa:	1.- Invoca la función para obtener resultados del analisis.	1.- Invoca resultados de la función que lo genera.
		2.- Muestra error y ejecuta excepción de no encontrar datos en el sistema.
Precondición:	El sistema debió haber realizado el análisis a la opinión ingresada.	
Poscondición:	El sistema muestra en pantalla el resultado lo cual es usado por el caso de uso ver resultado para que lo pueda visualizar el usuario.	

Fuente: Elaboración Propia

3.3.2. Descripción de casos de uso de corpus figura 3.

Nombre	Consulta de palabras – CU8	
Actores	Sistema	
Función	Consulta de palabras	
Descripción	El sistema consultara y revisara todas las palabras contenidas en el corpus para luego almacenarlos en listas según su tipo de palabra.	
Flujo Principal:	**EVENTO ACTOR**	**EVENTO SISTEMA**
	1.- Invoca la función para verificar existencia de palabras	1.- Evalua la existencia de palabras separadas por gramas.
	2.- Da conformidad de	

	existencia de palabras	
Alternativa:	1.- Invoca la función para verificar existencia de palabras	1.- Evalua la existencia de palabras separadas por gramas.
		2.- Muestra error y pide ingreso de palabras al corpus
Precondición:	El sistema separó la opinión en gramas.	
Poscondición:	El sistema detectó correctamente todos los gramas analizados.	

Fuente: Elaboración Propia

Nombre	Consulta de oracíon – CU9	
Actores	Sistema	
Función	Consulta de oracíon u opinión.	
Descripción	El sistema consultara todas las opiniones, haciendo contenerlas en listas para luego poder procesarlas.	
Flujo Principal:	**EVENTO ACTOR**	**EVENTO SISTEMA**
	1.- Invoca la función para verificar existencia de opiniones según polaridad.	1.- Envia según petición a procesar la opinión según polaridad
Alternativa:	1.- Invoca la función para verificar existencia de opiniones según polaridad.	1.- Envia según petición a procesar la opinión según polaridad
		2.- Muestra error y control de excepción al no encontrar opiniones contenidas.
Precondición:	El sistema separó la opinión en gramas y analizo correctamente la opinoi ningresada por el usuario.	
Poscondición:	El sistema grabo archivos externos llamados bag of words.	

Fuente: Elaboración Propia

3.3.3. Descripción de caso de entrenamiento con opiniones figura 4.

Nombre	Analiza opiniones almacenadas – CU10	
Actores	Sistema	
Función	Procesar las opiniones almacenadas en la base de datos	
Descripción	El sistema realizará el proceso de análisis a las opiniones tales como identificación de palabras, tipos de palabras, frecuencias de palabras, clasificación, etc.	
Flujo Principal:	**EVENTO ACTOR**	**EVENTO SISTEMA**
	1.- Invoca la función para separarlas en gramas cada opinión de entrenamiento, etiquetado de palabras y alamcenamiento de estas.	1.- Todas las opiniones analizadas las separa en grupos positivos y negativos, ejecutando los algoritmos ingresados en el sistema.
Alternativa:	1.- Invoca la función para separarlas en gramas cada opinión de entrenamiento, etiquetado de palabras y alamcenamiento de estas.	1.- Todas las opiniones analizadas las separa en grupos positivos y negativos, ejecutando los algoritmos ingresados en el sistema.
		2.- Muestra error y control de excepción al no encontrar opiniones contenidas.
Precondición:	El sistema separó la opinión en gramas y analizo correctamente la opinión ingresada por el usuario.	
Poscondición:	El sistema grabo archivos externos llamados bag of words.	

Fuente: Elaboración Propia

3.3.4. Descripción de casos de uso POS y Bag of Words figura 5.

Nombre	Identificar tipo de palabra en opiniones almacenadas – CU11
Actores	Sistema
Función	Identificar el tipo de palabra
Descripción	La interfaz del sistema comparara e identificar cada palabra de la opinon almacenada según el corpus actual.

Flujo Principal:	**EVENTO ACTOR**	**EVENTO SISTEMA**
	1.- Invoca la función para poder procesar la opinión y separar cada una en gramas.	
	2.- Invoca la función para determinar el tipo de palabra según grama.	2.- Determina el tipo de palabra por grama identificado
Alternativa:	1.- Invoca la función para poder procesar la opinión y separar cada una en gramas..	
		2.- Muestra error y control de excepción al no encontrar opiniones contenidas.
Precondición:	El sistema separó la opinión ingresada por el usuario en gramas	
Poscondición:	El sistema tendrá lista las palabras para grabarlas en el bag of words.	

Fuente: Elaboración Propia

Nombre	Enlista y clasifica palabras según polaridad de opiniones – CU12	
Actores	Sistema	
Función	Clasificara y enlistara las palabras por la polaridad de las opiniones	
Descripción	La interfaz del sistema clasificará y enlistará todas las palabras de las opiniones contenidas según el sentido que tenga cada una.	
Flujo Principal:	**EVENTO ACTOR**	**EVENTO SISTEMA**
	1.- Invoca la función para poder etiquetar las palabras según su tipo	1.- Se determina el tipo de palabra y a su vez cada palabra es enlistada en el bag fo wrds tanto para opinoines positivas y negativas
Alternativa:	1.- Invoca la función para poder etiquetar las	1.- Se determina el tipo de palabra y a su vez cada palabra

	palabras según su tipo	es enlistada en el bag fo wrds tanto para opinoines positivas y negativas
		2.- Muestra error y control de excepción al no encontrar opiniones contenidas.
Precondición:	El sistema separó la opinión ingresada por el usuario en gramas	
Poscondición:	El sistema tendrá los archivos de bag of words grabados.	

Fuente: Elaboración Propia

3.3.5. Descripción de casos de uso Bayes figura 6.

Nombre	Comparación de términos de Bag of Words con opinión ingresada – CU13	
Actores	Sistema	
Función	Comparar bag of words con opinión ingresada	
Descripción	La interfaz del sistema descompondrá en términos la opinión ingresada por el usuario y estos términos serán comparadas con Bag of Words para verificar su existencia	
Flujo Principal:	**EVENTO ACTOR**	**EVENTO SISTEMA**
	1.- Invoca la función para obtener la opinión ingresada por el usuario.	1.- se obtendrá las palabras separadas en gramas e identificadas por su tipo de palabras
		2.- Se compara mediante algoritmo la existencia de las palabras en el bag of words.
Alternativa:	1.- Invoca la función para obtener la opinión ingresada por el usuario.	1.- se obtendrá las palabras separadas en gramas e identificadas por su tipo de palabras
		2.- Muestra error y control de excepción de no existir la

		palabra en bag of words
Precondición:	El sistema analizó las opiniones de entrenamiento conjuntamente con la opinión a analizar para determinar la polaridad.	
Poscondición:	El sistema mostrará resultado de la comparación entre la opinión y el bag of words.	

Fuente: Elaboración Propia

Nombre	Cálculo de frecuencias en Bag of Words – CU14	
Actores	Sistema	
Función	Obtener la frecuencia de una palabra identificada en el bag of words.	
Descripción	La interfaz del sistema luego de verificar la existencia de una palabra en el bag of words realizará un conteo de la palabra y dara como resultado una frecuencia.	
Flujo Principal:	**EVENTO ACTOR**	**EVENTO SISTEMA**
	1.- Invoca la función para obtener la cantidad de frecuencia de palabras en el bag of words.	1.- Ingresara por lista y realizara un conteo por veces que aparece la palabra en todo el grupo de entrenamiento.
Alternativa:	1.- Invoca la función para obtener la cantidad de frecuencia de palabras en el bag of words.	1.- Ingresara por lista y realizara un conteo por veces que aparece la palabra en todo el grupo de entrenamiento.
		2.- Muestra error y control de excepción de no existir los archivos de bag of words
Precondición:	El sistema analizó las opiniones de entrenamiento conjuntamente con la opinión a analizar para determinar la polaridad.	
Poscondición:	El sistema obtendrá un resultado de la cantidad de repeticiones por palabra contenidos en el bag of words.	

Fuente: Elaboración Propia

Nombre	Cálcula y determina la probabilidad de polaridad de la oración.– CU15	
Actores	Sistema	
Función	Obtener la probabilidad más cercana.	
Descripción	La interfaz del sistema calculará la probabilidad de la opinión teniendo en cuenta varias etapas, la cantidad de palabras y cantidad de opiniones.	
Flujo Principal:	**EVENTO ACTOR**	**EVENTO SISTEMA**
	1.- Invoca la función para obtener para calcular la probabilidad de tendencia de polaridad de la opinión ingresada.	1.- Ejecuta el teorema de bayes teniendo en consideracion el bag of words y la opinin ingresada, dando un resultado
	2.- Obtiene en pantalla un resultado del análisis.	
Alternativa:	1.- Invoca la función para obtener para calcular la probabilidad de tendencia de polaridad de la opinión ingresada.	1.- Ejecuta el teorema de bayes teniendo en consideracion el bag of words y la opinin ingresada, dando un resultado
		2.- Muestra error y control de excepción de no existir los archivos de bag of words
Precondición:	El sistema analizó las opiniones de entrenamiento conjuntamente con la opinión a analizar para determinar la polaridad.	
Poscondición:	El sistema obtendrá un resultado de polaridad.	

Fuente: Elaboración Propia

3.4. Diagrama de secuencia

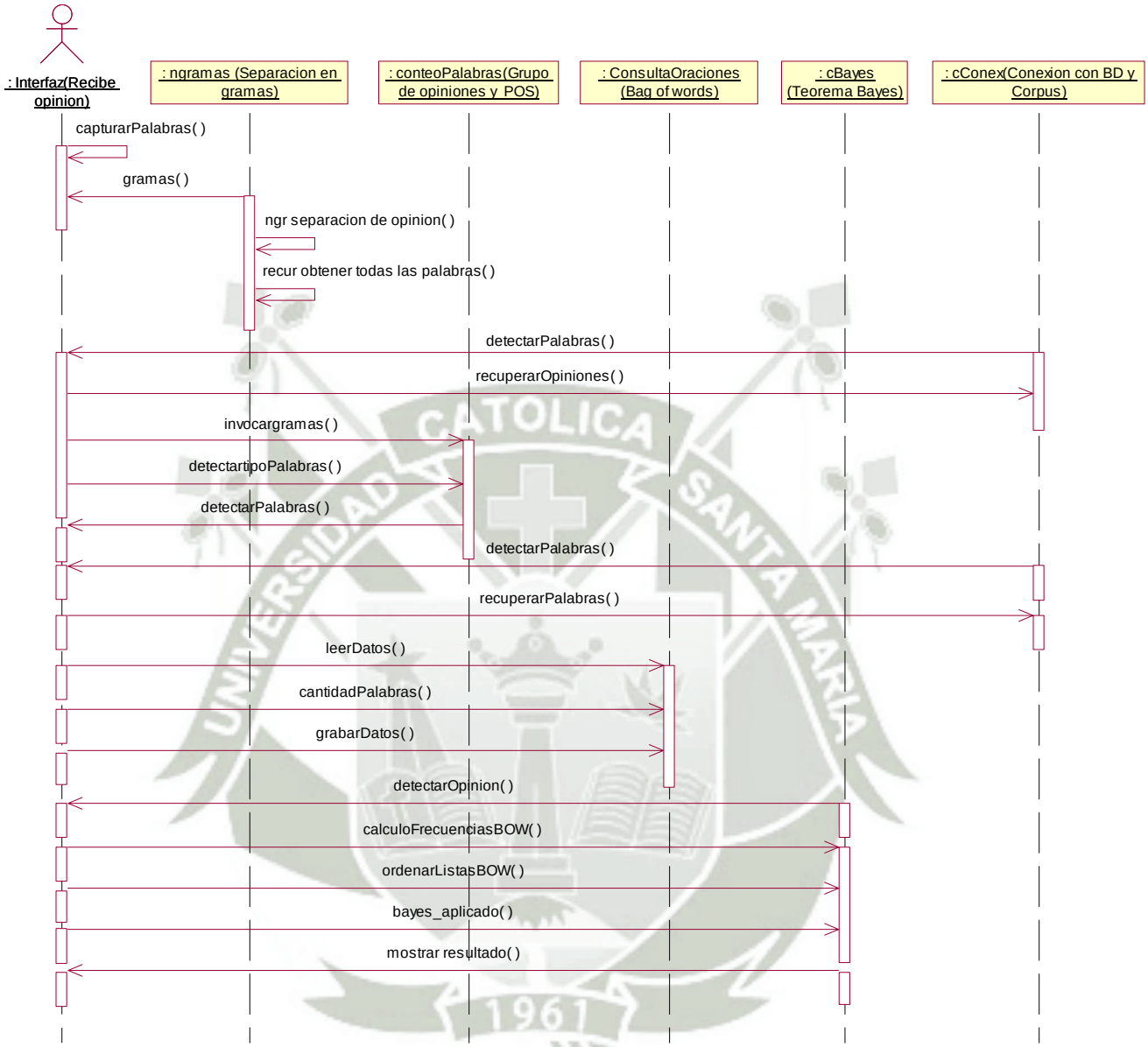


Figura 7.- Diagrama de secuencia Interacción del sistema.

Fuente: Elaboración propia

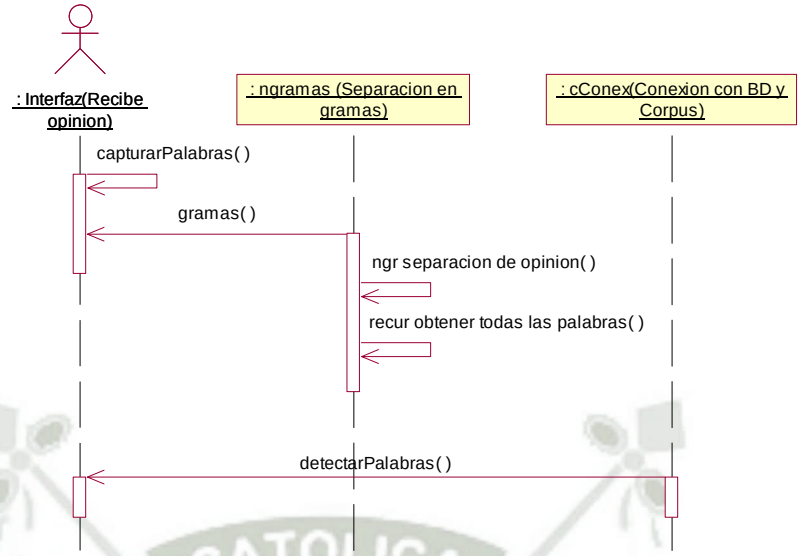


Figura 8.- Diagrama de secuencia N-gramas.

Fuente: Elaboración propia

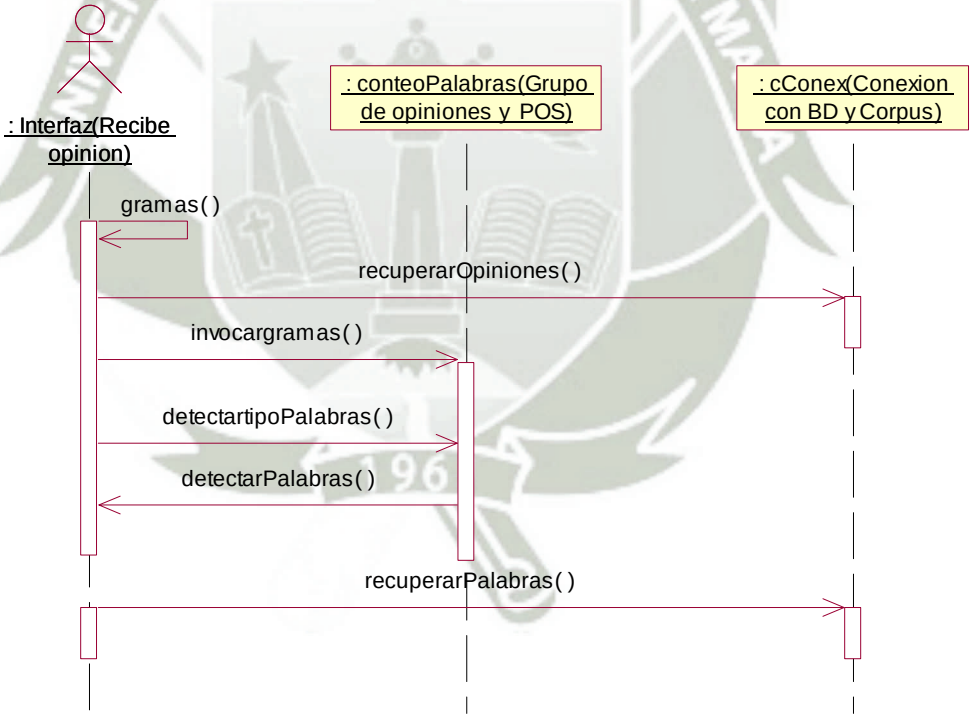


Figura 9.- Diagrama de secuencia de analisis de opinones y POS.

Fuente: Elaboración propia

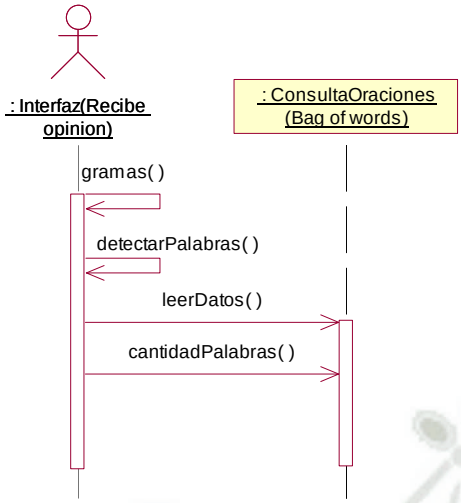


Figura 10.- Diagrama de secuencia de analisis de Bag of Words (BoW).

Fuente: Elaboración propia

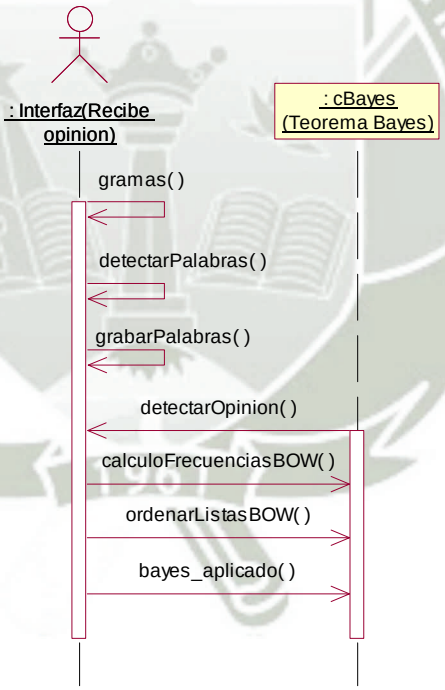


Figura 11.- Diagrama de secuencia de analisis de Bayes.

Fuente: Elaboración propia

3.5. Diagrama de Clases

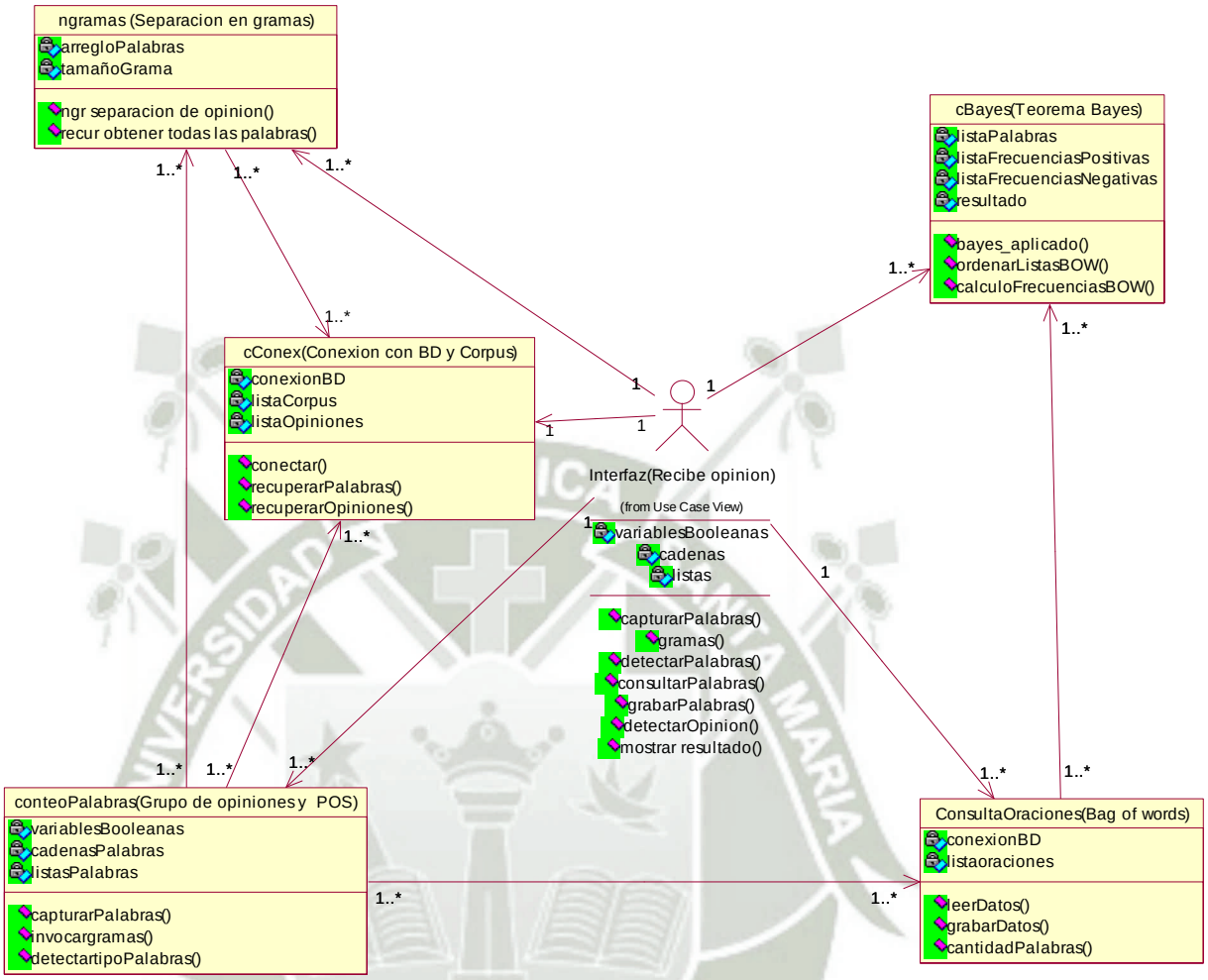


Figura 12.- Diagrama de Clases

Fuente: Elaboración propia

3.6. Pseudocódigo de bag of words

Pseudocódigo de Bag of Words
Datos entrada: Or: Lista de palabras
Datos de Trabajo: Cant: Entero cantidad de palabras Todo: Lista total de palabras palabPo: Lista de palabras por polaridad k: Arreglo auxiliar
Datos salida: palabPo: Retorna lista de palabras según polaridad
INICIO Ordenar_alfanumericamente(or) i=0 Repita Asignar_parecidos(todo) Contar_palabras(todo) i=i+cant-1 asignar_palabra(k[0]) asignar_tipopalabra(k[1]) asignar_polaridad(k[2]) k[3]=cant si k[1]!=”Articulo” && k[1]!=”Pronombre” &&k[1]!=”Conjuncion” && k[1]!=”Preposicion”

```
                palaPo = añadir_palabra(k)
            Fin si
            i=i+1
        hasta i < cantidad_palabras(or)
FIN ALGORITMO
```

Fuente: Elaboración propia

3.7. Pseudocodigo N-Gramas

Pseudocódigo de Ngramas
Datos entrada: Tamañograma: Entero que determina el tamaño del grama Arreglo: Array que contiene palabras
Datos de Trabajo: Tamañoriginal: Entero que obtiene el tamaño de Arreglo tam: Entero que obtiene el tamaño del grama k: Array de gramas count: Entero contador arre: Array auxiliar tamañodelgrama: Entero auxiliar
Datos salida: k: Array de gramas
INICIO Si tamaño(arreglo) < tamañograma FIN ALGORITMO

```
    Fin si
    Tamañoriginal=tamaño(arreglo)
    Tam=tamañograma-1
    c=0
    arre=arreglo
    tamañodelgrama=tamañograma
    i=0
    Repita
        Llamar función recur(c,i)
        Count=count+1
        C=0
        i=i+1
    Hasta i=tamaño(arreglo)-tam
FIN ALGORITMO
```

Fuente: Elaboración propia

3.8. Pseudocodigo Teorema de Bayes

Pseudocódigo de Bayes
Datos de Trabajo: Suma_oraciones:Double sumara cantidad de oraciones con: Entero cantidad oraciones negativas cop: Entero cantidad oraciones postitivas pOrPo: Double probabilidad a priori positiva pOrNe: Double probabilidad a priori negativa

condP: Double probabilidad condicional positiva condN: Double probabilidad condicional negativa probTotal: Double probabilidad total bayesN: Double probabilidad aposteriori negativa bayesP: Double probabilidad aposteriori positiva frecuenciasPositivas: Lista de frecuencias de palabras positiva frecuenciasNegativas: Lista de frecuencias de palabras Negativas
INICIO Suma_oraciones=con+cop pOrPo=cop/suma_oraciones pOrNe=con/suma_oraciones i=0 Repita Si frecuenciasPositivas(i)!=noexiste condP= frecuenciasPositivas(i)+condP Fin si i=i+1 Hasta i=tamaño(frecuenciasPositivas) i=0 Repita Si frecuenciasNegativas(i)!=noexiste condN= frecuenciasNegativas (i)+condN Fin si

```
        i=i+1
    Hasta i=tamaño(frecuenciasNegativas)
    probTotal=(pOrPo*condP)+(pOrNe*condN)
    bayesN=(pOrNe*condN)/probtotal
    bayesP=(pOrPo*condP)/probtotal
    Si bayesN > bayesP
        Mostrar “La oración es Negativa”
    Caso contrario
        Mostrar “La oración es Positiva”
    Fin si
FIN ALGORITMO
```

Fuente: Elaboración propia

4. CASO DE ESTUDIO

Para el caso de estudio se tomó en cuenta la cantidad de opiniones vertidas en internet sobre un tema de opinión específico.

Se tuvo en cuenta las siguientes restricciones:

- Tema de estudio

  El tema a estudiar es aleatorio, en este estudio se escogió varios temas el cual es recolectado de internet.

- Número de opiniones

  El número de opiniones debe ser mayor a 70 opiniones y con una base real de 455 opiniones, es importante el número; ya que entre más opiniones tenga el sistema sobre un tema específico su capacidad de distinción de polaridad será más eficaz.

- Número de prueba de opiniones

  Se tuvo en consideracion 180 opiniones para la prueba de detección de la polaridad binaria.

- Lenguaje de desarrollo C# y gestor de base de datos MySQL

- Número de palabras sobre un tema de opinión

  Al igual que el número de opiniones, también el número de palabras influye sobre los temas elegidos a procesar, debido a que existe una mayor repetición de ciertas palabras características sobre un tema dado.

- El tamaño máximo de los n-gramas

Dado a que el sistema consume recursos, la cantidad de n-gramas permitidos que no afectaran el rendimiento de procesamiento del sistema considerablemente es de 3 gramas como máximo.

- Polaridad de identificación de las oraciones en base binaria

  El sistema planteado solo detectara la polaridad de una oración en base a positivos y negativos.

4.1. Implementación de la propuesta

Etapa	Actividad	Método	Resultado
Primera	Elaboración del diccionario de palabras o corpus (Eladio Blanco, Fernando Martinez y Antonio Pantoja, 2009)	Almacenamiento de términos según sentencias SQL en programa MYSQL. Actualización de términos nuevos.	Base de términos o palabras según clasificación léxica: • Verbo • Adjetivo • Adverbio • Sustantivos
Segunda	Implementación del sistema de análisis de sentimientos.	• Bag of words • N-Gramas o cadenas de markov. • Term Frequency o	Procesamiento de opiniones para obtener la polaridad

		frecuencia de términos. • Part of spech (Tagging) • Teorema de Bayes	
Tercera	Prueba de funcionamiento y regularización en pequeñas muestras (Eladio Blanco, et. al, 2009).		Sistema eficiente

Fuente: Elaboración Propia

4.2. Desarrollo de la propuesta

4.2.1. Diccionario de palabras o corpus

Términos	Cantidad
Verbos	**4667**
Adjetivos	**7748**
Sustantivos	**17470**
Adverbios	**853**
Conjunciones	**44**

Preposiciones	**50**
Pronombre	**840**
Artículos	**11**
Total	**31683**

Fuente: Elaboración Propia

4.2.2. Sistema de análisis binario de sentimientos del lenguaje natural en el idioma español

- Desarrollado en el lenguaje de programación C#
- Programa orientado a objetos
- Programa de la línea de inteligencia artificial supervisado (Bo Pang yLillian Lee, 2008).
- Multicapa (Base de datos, interfaz, usuarios)
- Flexible
- Robusto(Soporte a errores)
- Lenguaje formal.
- Requiere Base de datos inicial de opiniones prototipo positivas o negativas.

4.2.3. Bag of Words

Comprende:

- Recepción opiniones
- Segmentación de términos.
- Proceso por tipo de palabra, polaridad y repetición
- Filtrado de acuerdo a términos de interés.

4.2.4. N-Gramas o cadenas de Markov

Comprende:

- Identificación o descarte de términos compuestos.

4.2.5. Term Frequency o Frecuencia de Términos

Comprende:

- Frecuencia de Verbos, adjetivos, adverbios, sustantivos que aparecieron en todo el grupo de entrenamiento del sistema, para poder realizar el análisis respectivo en un conjunto de 455 opiniones sobre diversos temas.

Ejemplo:

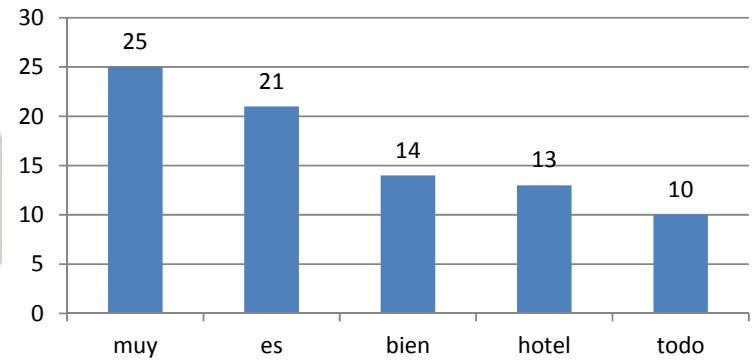


Figura 13. Frecuencia de términos

Fuente: Elaboración propia

4.2.6. Part of Spech (Tagging)

Comprende la segmentación e identificación de las palabras ya procesadas ya sea cómo adjetivo, sustantivo, adverbio, etc, ejemplo:

PRONOMBRE	VERBO	PREPOSICION	ARTICULO	SUSTANTIVO
ELLA	AMA	A	LOS	ANIMALES

Fuente: Elaboración Propia

4.2.7. Teorema de Bayes

El teorema de Bayes se encargará de la categorización de las opiniones, sobre un grupo determinado..

5. EVALUACIÓN Y RESULTADOS

5.1. Ambiente de Desarrollo

El sistema de análisis binario de sentimientos en el idioma español, está desarrollado en el entorno de programación de C# de Visual Studio .NET y con un gestor de base de datos MySQL.

5.2. Características del Ambiente de Pruebas

5.2.1. Equipos de Cómputo

Las pruebas se realizaron utilizando 1 computador con la siguiente característica:

Computadora 1 (PC 1)	
Sistema Operativo:	Microsoft Windows 7 Professional Service Pack 1
Procesador:	Pentium(R) Dual-Core CPU E5400 @ 2.70GHz 2.69GHz
Memoria RAM:	2 GB DDR2
Tarjeta de Video:	NVIDIA GeForce 7300 SE/7200 GS 256.00 MB
Disco Duro:	SAMSUNG HD502HI ATA Device

Tabla 5-1 Características Computadora 1

Fuente: Elaboración propia

5.2.2. Casos de Evaluación

Para la evaluación del análisis de sentimientos se tuvo en cuenta:

5.2.2.1. Caso de Evaluación 1.

En una oración o conjunto de palabras donde todas las palabras existen y son comparables con el corpus, su fácil identificación para una mejor fiabilidad en el sistema.

	PRONOMBRE	VERBO	PREPOSICION	ARTICULO	SUSTANTIVO
CORPUS CONTIENE	SI	SI	SI	SI	SI
PALABRAS	ELLA	AMA	A	LOS	ANIMALES

Fuente: Elaboración Propia

5.2.2.2. Caso de Evaluación 2.

En una oración o conjunto de palabras, palabras importantes que considera el sistema para identificar no existen en la base de datos o corpus, impidiendo una identificación correcta, por lo cual solicita completar estos términos.

	VERBO	ADVERBIO	PREPOSICION	SUSTANTIVO
CORPUS CONTIENE	NO	NO	SI	SI
PALABRAS	LOGRÓ SALIR	SATISFACTORIAMENTE	DEL	EXAMEN

Fuente: Elaboración Propia

5.3. Evaluación de Indicadores.

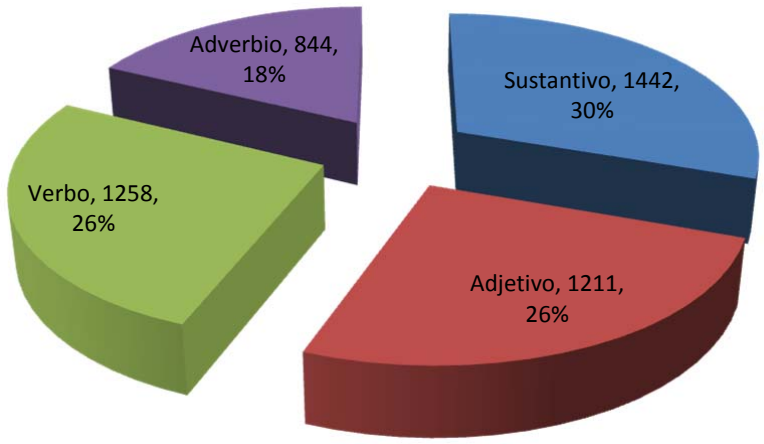
5.3.1. Integridad del mensaje

Cuadro Nº1. Integridad del mensaje por tipo de palabras identificadas

Tipo de palabra	Positivos	Negativos	Total	Porcentaje
Sustantivo	674	768	1442	30.33
Adjetivo	627	584	1211	25.47
Verbo	510	748	1258	26.46
Adverbio	305	539	844	17.75
Total	2116	2639	4755	100

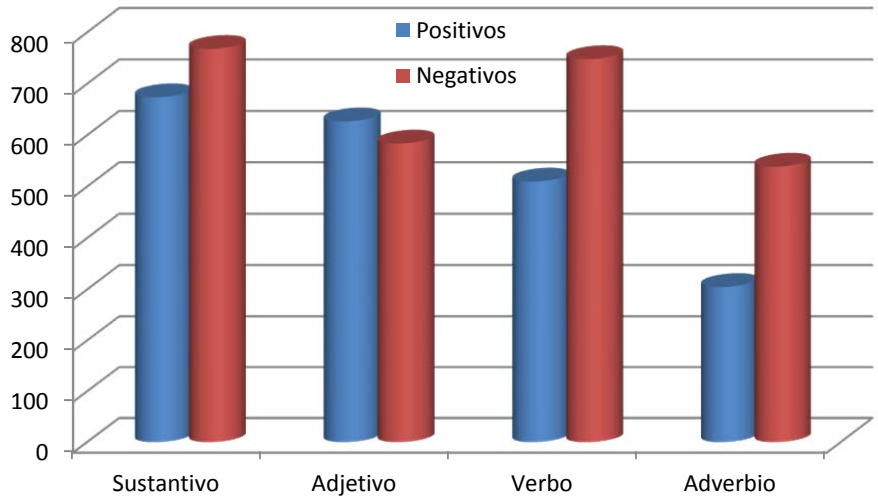
De un total de 455 opiniones el sistema ha identificado 4755 términos relevantes para la investigación de los cuales 30.33% son sustantivos, 25.47% adjetivos, 26.46% verbos y 17.75% adverbios tal como se aprecia en el gráfico siguiente.

Gráfico Nº1. Integridad del mensaje por tipo de palabras identificadas según porcentaje.



A pesar que los sustantivos tienen la mayor frecuencia en la conformación de las expresiones no son determinantes del sentido o polaridad de estas, a diferencia de los adverbios, verbos o adjetivos, sin embargo se ha tomado en cuenta a los sustantivos porque algunos expresan polaridad en sí mismos como “Amor, Felicidades, Cumpleaños, Soluciones, etc.”.

Gráfico Nº2. Integridad del mensaje por tipo de palabras identificadas según sentido de la oración.

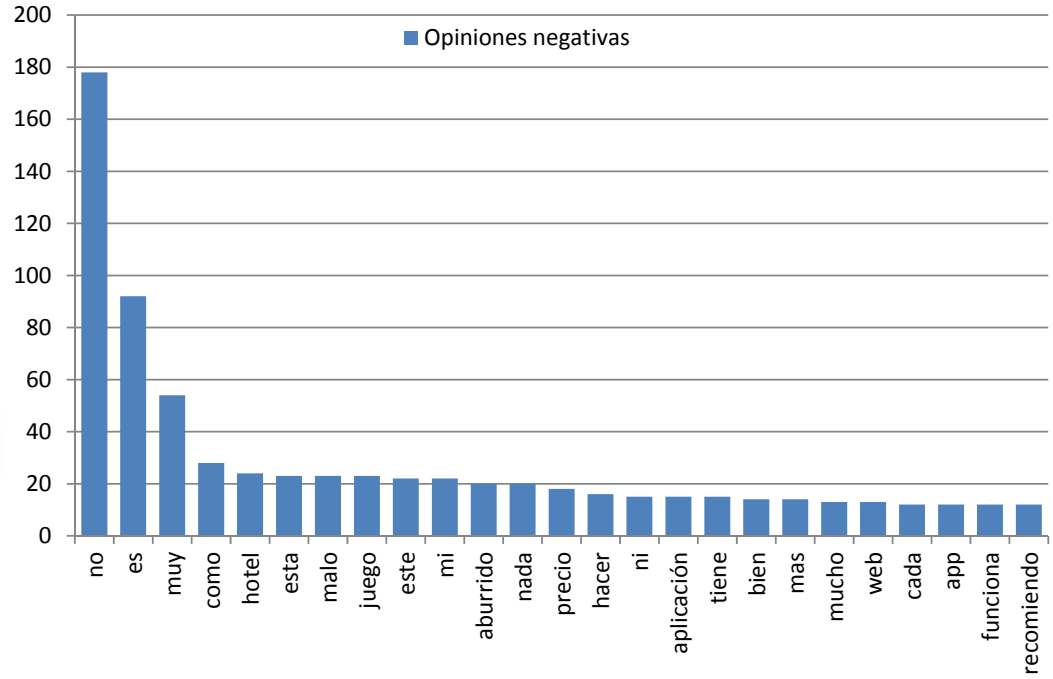


El gráfico muestra la predominancia de oraciones negativas por el mayor uso de verbos (es, tiene, está, viajar), adjetivos (excelente, buena, todo, esta), verbos (es, recomiendo, estuve, ir) y adverbios (no, muy, pero, como).

Donde se puede inferir que a mayor cantidad de palabras se obtendrá un mejor resultado al momento de realizar el análisis.

5.3.2. Ponderabilidad

Gráfico Nº3. Términos comunes en opiniones negativas

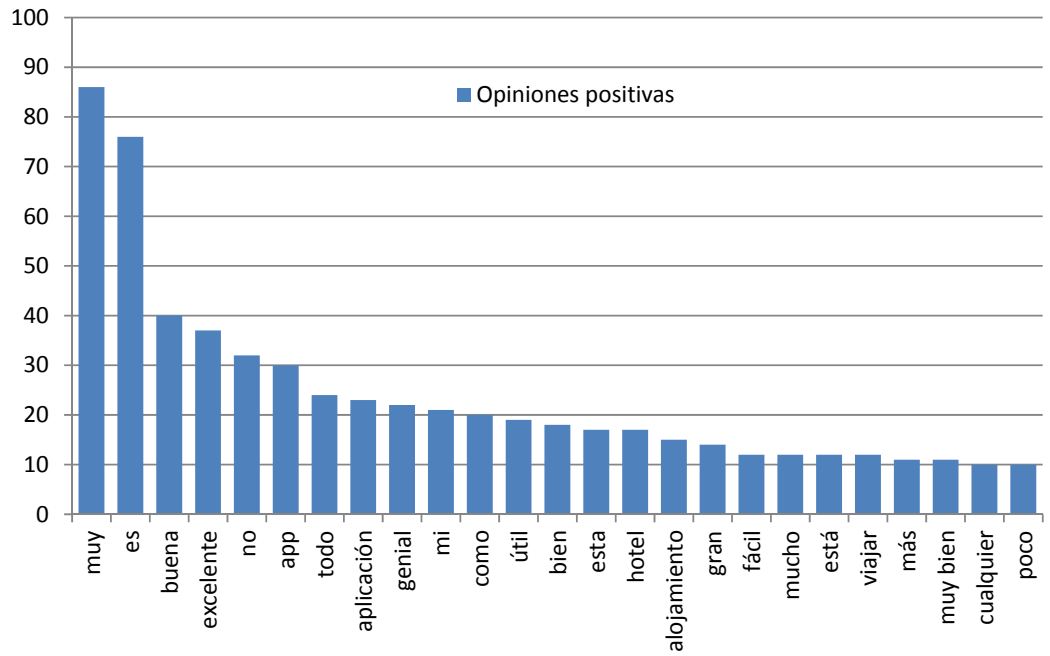


El gráfico 3 muestra los términos mas usados al momento de vertir opiniones de diferente contexto. Se puede apreciar las palabras predominantes en las opiniones negativas, lo que demarca la aceptación de cierto público objetivo en una determinada área.

En las oraciones negativas destaca mucho los adverbios (no, muy, como).

En este tipo de gráfico se pude inferir la idea principal de que temas esta abarcando las opiniones, según las palabras principales para las opiniones negativas.

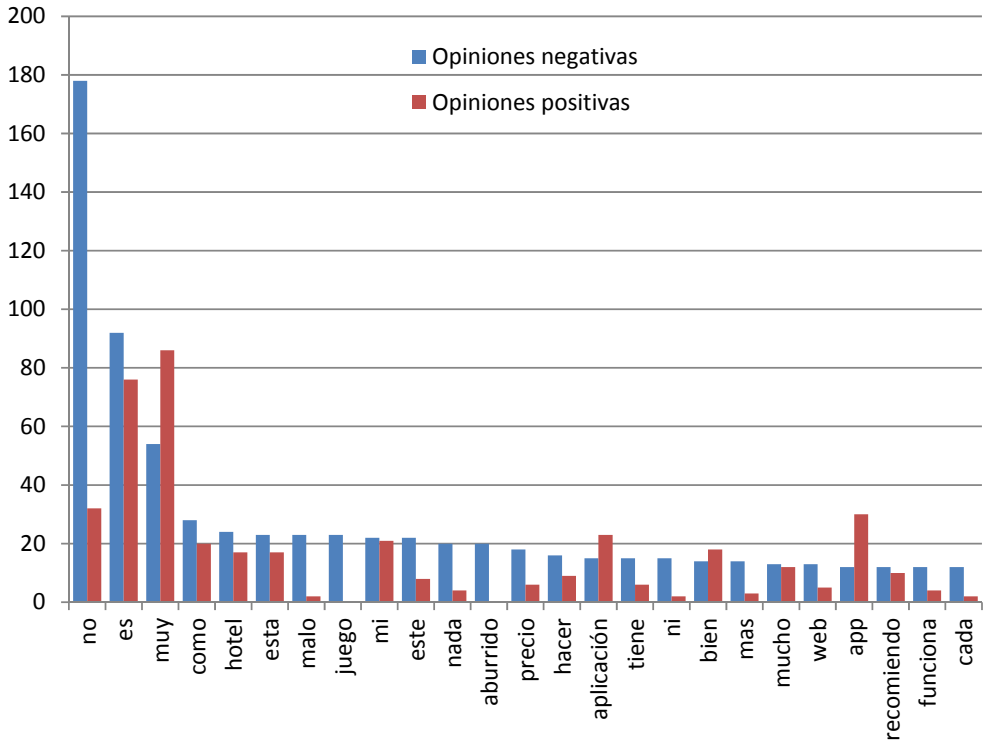
Gráfico Nº4. Términos comunes en opiniones positivas.



En el grafico Nº4 se observa los términos usados en las opiniones positivas que al igual que las opiniones negativas destacan aspectos del contexto al cual se trata ya sea de hoteles o aplicaciones web. Por ser opiniones positivas destacan los adverbios (muy, no) y adjetivos (excelente, buena, todo, útil, mi, genial, bien, gran).

Al igual que el gráfico en la pagina 72 se puede hacer una inferencia de la idea principal sobre que temas influyen en las opiniones brindadas o capturadas.

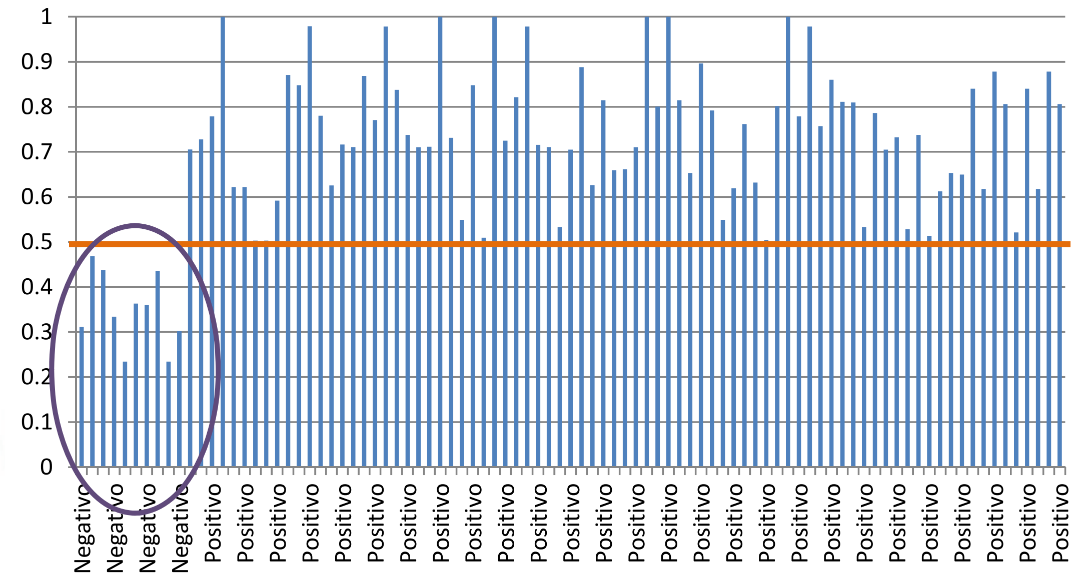
Gráfico Nº5. Contraste de palabras comunes en oraciones positivas y negativas



En el gráfico Nº5 se observa los términos comunes vertidos entre opiniones positivas y negativas los cuales varían en sus frecuencias, destacando las palabras, “muy”, “hotel”, “mi” entre oraciones positivas y “no”, “es” y “como” entre oraciones negativas.

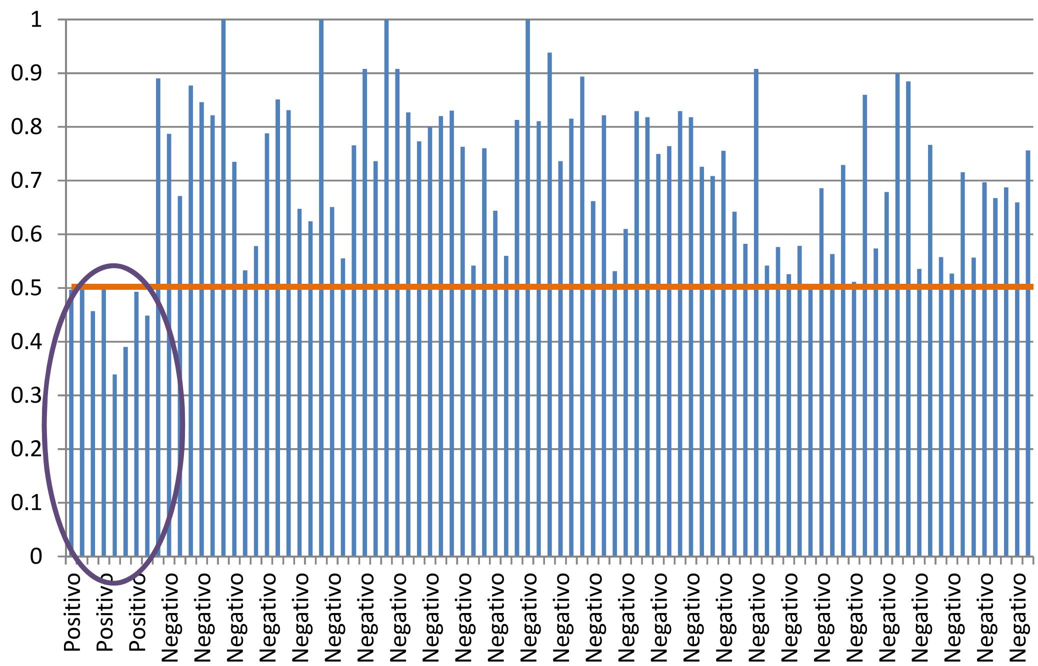
5.3.3. Subjetividad

Gráfico Nº6. Oraciones positivas según aplicación del Teorema Bayes.



Al aplicar el teorema de Bayes al grupo de oraciones positivas el sistema ha reafirmado la polaridad positiva en la mayoría de ellas (88.89%), mientras que el 11.11% ha variado la polaridad de positiva a negativa.

Gráfico Nº7. Oraciones negativas según aplicación del Teorema Bayes



La aplicación del teorema de Bayes a las opiniones negativas reafirma la polaridad en un 91.11% donde 8.89% han variado de negativas a positivas.

Se puede afirmar que la polaridad negativa tanto como positiva se vuelven mas consistentes en sus resultados debido a que el corpus obtenido en la base de datos es aceptable y contiene un buen grupo de palabras que permiten detectar en la mayoría las palabras contenidas en cada opinión pudiendo disgregarlos al momento del análisis teniendo en ambos casos un acierto de 88.89% y 91.11%; sin embargo estos resultados no son concluyentes ya que a mayor data, las detecciones de acierto pudieran ser más altas.

La variación de la polaridad de las oraciones al aplicar el Bayes indica la existencia de oraciones falsas positivas y falsas negativas.

5.3.4. Veracidad

Para poder determinar la veracidad del sistema se aplica una métrica que son los índices de precisión (P), recuperación (R) y eficiencia (F). También es conocido como los falsos positivos, donde la formula se muestra a continuación.

	Maq. Si	Maq. No
Hum. Si	A	b
Hum. No	C	d

Indicador de Precisión:

$$P = \frac{a}{a + c}$$

Indicador de Recuperación:

$$R = \frac{a}{a + b}$$

Indicador de eficiencia:

$$F = \frac{2 * R * P}{R + P}$$

Aplicado esto a nuestro sistema vendría a ser:

	Maq. Si	Maq. No
Hum. Si	79	11
Hum. No	8	81

Precisión:

$$P = \frac{79}{79 + 8} = 0.91 = 91\%$$

Recuperación:

$$R = \frac{79}{79 + 11} = 0.88 = 88\%$$

Eficiencia:

$$F = \frac{2 * 0.91 * 0.88}{0.91 + 0.88} = \frac{1.4432}{1.7} = 0.90 = 90\%$$

Para realizar estos cálculos, se tuvo en consideración el número de opiniones evaluadas tanto positivas como negativas, que en este caso son 90 opiniones positivas y 90 negativas y el número de error de clasificación realizada por la máquina que seria 11 opiniones mal clasificada en la polaridad positiva y 8 opiniones mal clasificadas en la polaridad negativa, teniendo estos datos y su análisis realizado podemos aplicar las fórmulas, dando como resultado una precisión de análisis de 0.91 ó 91%, de recuperación frente a lo determinado por el humano de 0.88 ó 88% y de eficiente al evaluar las opiniones de un 0.90 ó 90%.

Según Alexander Pak y Patrick Paroubek (2010) una buena clasificación sobre un grupo experimental esta en 81% y Peter D. Turney (2002) que el rango de clasificación esta en 78% a 92%, concluyendo que se obtuvo resultados aceptables cabe destacar que a medida se aumenten más opiniones a la base de datos o corpus estos valores variaran.

CONCLUSIONES

1. El sistema de análisis de sentimientos puede ejecutar el análisis de opiniones formales en castellano vertidas en la web u obtenidas por otros medios, según su polaridad.
2. Se logró implementar una base de términos léxicos o corpus con 31683 términos, como fuente de informacion del sistema de análisis de sentimientos binaria, tamaño que garantiza la precisión, recuperación y eficiencia del proceso.
3. El sistema de análisis de sentimientos cumple con cuatro criterios: de la Integridad al disgregar la oración por sus componentes semánticos; la Ponderabilidad al identificar la idea principal de la opinión captada por la frecuencia de términos; la Subjetividad al identificar la polaridad; la Veracidad al identificar el sentido correcto de la oración.
4. Para el análisis se tuvo en consideración 455 opiniones como entrenamiento y 180 opiniones como conjunto de detección, siendo un total de 600 oraciones.
5. En 180 Opiniones procesadas con 4755 terminos relevantes, se obtuvo 30.33% de sustantivos, 25.47% adjetivos, 26.46% verbos y 17.75% adverbios; se detectó 25 terminos mas frecuentes; se identificaron 88.89% de oraciones positivas y 91.11% de negativas; con 91% de presicion, 88% de recuperación y 90% de eficiencia, contrastado con Alexander Pak y Patrick Paroubek (2010) en 81% y Peter D. Turney (2002) en un rango de 78% a 92%.

RECOMENDACIONES

1. Se recomienda utilizar los sistemas de analisis de sentimientos en instituciones y/o empresas que trabajen en base a opiniones de usuarios, en evaluación de productos o servicios diversos.

2. Procurar un corpus actualizado de palabras, debido a que entre mayor sea el corpus mayor tasa de éxito hay que sea identificada en el bag of words consecutivamente mayor será la precisión dada por el sistema.

3. Se recomienda incidir en la motivación a los alumnos en construcción de sistemas automatizados y su investigación.

4. Se puede mejorar el presente sistema de análisis de sentimiento innovando los algoritmos o aplicar en otro idioma al cambiar el corpus y el sistema de etiquetado POS.

APLICACIONES A FUTURO

El sistema propuesto analiza opiniones subjetivas recorriendo y descomponiendo cada una pero sin considerar el contexto al que esta sujeta cada opinión, se podría mejorar a futuro analizar la opinión teniendo en cuenta el tipo de contexto al que esta sometido cada opinión, además de mejorar el análisis en cuanto al rango de la polaridad positivo y negativo, y extendiéndolo a tipos de identificación de las opiniones tales como alegre, triste, ironico, amargo, etc.

El análisis de sentimientos es un tema que actualmente está siendo tratado y el cual abre muchos campos a analizar en lo que es cuestión de captura de opiniones, tales como puede ser:

- Sistema de mapas donde se indique los lugares seguros e inseguros, según las opiniones de los usuarios.
- Ver el estado de un producto en el mercado.
- Ver la aceptación de un político.
- Obtener información relevante de un país.
- Conformidad de alumnos frente a un curso en una universidad.
- Soluciones de negocio mediante la captura de opiniones en redes sociales.

REFERENCIAS

Margaret M. Bradley and Peter J. Lang (1999). Affective Norms for English Words (ANEW): Instruction Manual and Affective Ratings

Javi Fernandez, Ester Boldrini, Jose Manuel Gómez y Patricio Martinez-Barco (2011). Análisis de Sentimientos y Minería de Opiniones: el corpus EmotiBlog

Eladio Blanco López, Fernando Martínez Santiago, Antonio Pantoja Vallejo (2009). Análisis Automático de Emociones en la Red Internacional E-Culturas

Hernández M., Gómez J. (2014). Análisis de Sentimientos Aplicado a Referencias Bibliográficas

Nobuhiro Kaji y Masaru Kitsuregawa (2010). Building Lexicon for Sentiment Analysis from Massive Collection of HTML Documents

Felipe Bravo Marquez, Marcelo Mendoza, Barbara Poblete (2013). Combining Strengths, Emotions and Polarities for Boosting Twitter Sentiment Analysis

Robert C. Moore, Chris Quirk (2009). Improved Smoothing for N-gram Language Models Based on Ordinary Counts

Bin Li, Yuan Guoyong (2012). Improvement of TF-IDF Algorithm Based on Hadoop Framework

Cristina Cachero y Pedro J. Ponce de León (2011). Introducción al Paradigma Orientado a Objetos

Bing Liu (2007). Opinion Mining

Bo Pang y Lillian Lee (2008). Opinion Mining and Sentiment Analysis

Alexander Gelbukh (2010). Procesamiento de Lenguaje Natural y sus Aplicaciones

James W. Pennebaker, Matthias R. Mehl, y Kate G. Niederhoffer (2003). Psychological aspects of natural language use our words our selves

TheresaWilson, JanyceWiebe, Paul Hoffmann (2005). Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis

Daniel Jurafsky y James H. Martin (1999). Speech and Language Processing

Eugene Charniak (1997). Statistical Techniques for Natural Language Parsing

S. B. Kotsiantis (2007). Supervised Machine Learning: A Review of Classification Techniques

Peter D. Turney (2002). Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews

Hanna M. Wallach (2006). Topic Modeling: Beyond Bag-of-Words

Bal Krishna Bal (2009). Towards an Analysis of Opinions in News Editorials: How positive was the year

Alexander Pak, Patrick Paroubek (2010). Twitter as a Corpus for Sentiment Analysis and Opinion Mining

Alec Go, Richa Bhayani, Lei Huang (2009). Twitter Sentiment Classification using Distant Supervision

Yin Zhang, Rong Jin y Zhi-Hua Zhou (2010). Understanding Bag-of-Words Model: A Statistical Framework

Kunpeng Zhang, Yu Cheng, Yusheng Xie, Daniel Honbo, Ankit Agrawal, Diana Palsetia, Kathy Lee, Wei-keng Liao, y Alok Choudhary (2014). SES: Sentiment Elicitation System for Social Media Data

H. Cordobes, A. Fernandez, L. Nuñez, F. Perez, T. Redondo, A. Santos (2013). Técnicas basadas en grafos para la categorización de tweets por tema

Marcel Caraciolo (2010). Working on Sentiment Analysis on Twitter with Portuguese Language http://aimotion.blogspot.com/2010/07/working-on-sentiment-analysis-on.html

Danube (2010). SCRUM Methodology. http://scrummethodology.com

Jhon Pruitt (2011). Perspectives on software development. http://blog.jgpruitt.com/tag/personal-scrum/page/2/

Derek Davidson (2014). Can Personal Scrum be used for a team of one? http://webgate.ltd.uk/personal-scrum/

ANEXO A

CODIGO DE IMPLEMENTACION DEL SISTEMA DE ANALISIS DE SENTIMIENTOS

FORMULARIOS

Formulario.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using MySql.Data.MySqlClient;
using System.Text.RegularExpressions;

namespace sentiment
{
    public partial class Form1 : Form
    {
        bool[] palabradisponible, oracion;
        bool contar = false, conexiond, ppronombre, particulo, psustantivo,
        padjetivo, pverbo, padverbio, pconjuncion;
        string[] palabra, matr_aux,grama2, grama3, grama1, grama4,
        grama5;//gramas a comparar
        string[][] cgrama;
        int global_entero,cont_palabras, cArt, cPron, cPrep, cAdj, cSust, cVerb,
        cAdv, cConj;//para contar las palabras que tipos existen
        char[] delimiters;
        List<string> lpronombre, larticulo, lsustantivo, lverbo, ladjetivo,
        ladverbio, lconjuncion, lpreposicion;//variables donde se almacenaran las
        palabras
        List<string[]> oracioncompleta;//variable importante
        List<int> indicesoracioncompleta;//variable importante
        List<List<string[]>> consultaoraciones;
        List<cPalabra> lp;
        cConex cc;
        consultaOraciones variableConsulta;// esta variable conectara para hacer
        la consulta de todas las oraciones almacenadas en la BD
        cBayes bayes;
        Form2 k;
```

```
public Form1()
{
    cc = new cConex();
    InitializeComponent();
    conexiond = cc.conectar();
    oracioncompleta = new List<string[]>();//variable donde se
    almacenara las palabras detectadas para su posterior refinamiento
    indicesoracioncompleta= new List<int>();//variable donde se almacen
    los indices de las palabras detectaddas para su correcta ubicacion
}
private void captureButton_Click(object sender, EventArgs e)
{
    bayes = new cBayes();
    variableConsulta = new consultaOraciones();
    oracionTextBox.Select(0, oracionTextBox.Text.Length);//obtenemos la
    longitud de la oracion para su segmentacion de palabrras posterior
    oracionTextBox.SelectionColor = Color.Black;
    limpiarlabel();
    if (oracionTextBox.Text!="")
        gramas();//en esta funcion las las palabras seran almacenadas en
    arrays de gramas para su posterior clasificacion
    consultaoraciones=variableConsulta.consulta_oraciones();//esto es
    solo prueba
    bayes.oraciones(consultaoraciones.ElementAt(0),
    consultaoraciones.ElementAt(1), oracioncompleta);
    if(checkBox1.Checked){
        k = new Form2(consultaoraciones.ElementAt(0),
        consultaoraciones.ElementAt(1));
        k.Show();
    }
}
void false_funcion(int tamaOra)//funcion que ayudara a controlar las
                                palabras detectadas con las
{   //no detectadas
    oracion= new bool[tamaOra];
    for (int i = 0; i < oracion.Length;i++)
    {
        oracion[i] = false;//todas las palabras inicialmente estan en
                           false
    }
}
void poner_textBox(string[] fragmentosOracion)//tiene la finalidad de
limpiar de signos de puntuacion para la correcta deteccion en la oracion
{
    string or="";
    for (int i = 0; i < fragmentosOracion.Length;i++ )
    {
        or += fragmentosOracion[i]+" ";
    }
    oracionTextBox.Text = or.ToLower();
    oracionTextBox.Refresh();
}
void gramas()//funcion que servira para separar las gramas de varios
              tipos
{
    oracioncompleta.Clear();
    indicesoracioncompleta.Clear();//limpiamos siempre las variables
    string[][] gp, gp2, gp3;
    ngramas arr = new ngramas();
    delimiters = new char[] { '\r', '\n', ' ', ',', ':', '.', '?', '¿',
```

```
                                '!', '¡', '(', ')', ';','-','/' };
    string[] fragmentosOracion = oracionTextBox.Text.Split(delimiters,
                                 StringSplitOptions.RemoveEmptyEntries);
    poner_textBox(fragmentosOracion);
    int c = 1;
    cPron = 0; cArt = 0; cPrep = 0; cAdv = 0; cAdj = 0; cSust = 0;
    cVerb = 0; cConj = 0;
    grama1 = arr.ngr(1, fragmentosOracion);
    grama2 = arr.ngr(2, fragmentosOracion);
    grama3 = arr.ngr(3, fragmentosOracion);
    if (fragmentosOracion.Length == 2)
    {
    //    cgrama = arr.complemento(2, fragmentosOracion);
    }
    if (fragmentosOracion.Length == 3)
    {
     //  cgrama = arr.complemento(3, fragmentosOracion);
    }
    if (fragmentosOracion.Length == 4)
    {
      //  cgrama = arr.complemento(4, fragmentosOracion);
    }

    false_funcion(fragmentosOracion.Length);//se envia el tamaño de la
    oracion original para poder asignarles true en una etapa posterior
    detectar_palabras(grama3, grama1,3);
    detectar_palabras(grama2, grama1,2);
    detectar_palabras(grama1, grama1,1);
    especificacion_palabra();
}
void especificacion_palabra()//en esta funcion se ordenara las palabras
                             tal como se ingreso en la oracion
{//y no como estaban en el orden al haber sido detectadas por los gramas
    string[][] arrax = oracioncompleta.ToArray();//asignamos los valores
    almacenados en un array
    int[] ormtr = indicesoracioncompleta.ToArray();//tomamos en cuenta
    los indices que facilitaran el ordenamiento en la oracion
    arrax=MetodoBurbuja(arrax, ormtr);//como resultado lo sometemos a un
    ordenamiento burbuja  dando el ordenamiento esperado
    orden_correcto_palabras(arrax);
}
void orden_correcto_palabras(string[][] arx)//en esta funcion
procederemos a detectar correctasmente las palabras
{// de esta manera podremos tener en cuenta las reglas al momento que se
    escriben en una oracion tanto para
    // diferenciar sustantivos, adjetivos, adverbios, verbos, etc...
    for (int i = 0; i < arx.Length; i++)
    {
        //pronombre
        if (arx[i][1] == "Pronombre" )//reglas de identificacion de las
                                      palabras
        {
            if (arx.Length - 1 == i)
            {
                es_pronombre(arx[i][0], i);
                continue;
            }
            else if(arx[i + 1][1] == "Verbo" || arx[i + 1][1] ==
            "Conjuncion"|| arx[i + 1][1] == "Adverbio"){
                es_pronombre(arx[i][0], i);
            }

        }//articulo
```

```
else if (arx[i][1] == "Articulo")
{
    if (arx.Length - 1 == i)
    {
        es_articulo(arx[i][0], i);
        continue;
    }
    else if(arx[i + 1][1] == "Sustantivo" || arx[i + 1][1] ==
    "Adjetivo"){
        es_articulo(arx[i][0], i);
    }
}//sustantivo
else if (arx[i][1] == "Sustantivo" )
{
    if (arx.Length - 1 == i)
    {
        es_sustantivo(arx[i][0], i);
        continue;
    }
    else if(arx[i + 1][1] == "Verbo" || arx[i + 1][1] ==
    "Adjetivo"||arx[i + 1][1] == "Adverbio" || arx[i + 1][1] ==
    "Preposicion" || arx[i + 1][1] == "Conjuncion"){
        es_sustantivo(arx[i][0], i);
    }
}//verbo
else if (arx[i][1] == "Verbo" )
{
    if (arx.Length - 1 == i)
    {
        es_verbo(arx[i][0], i);
        continue;
    }
    else if(arx[i + 1][1] == "Sustantivo" || arx[i + 1][1] ==
    "Adjetivo"||arx[i + 1][1] == "Adverbio" || arx[i + 1][1] ==
    "Conjuncion" || arx[i + 1][1] == "Articulo"
    || arx[i + 1][1] == "Pronombre" || arx[i + 1][1] == "Verbo"
    || arx[i + 1][1] == "Preposicion")
    {
        es_verbo(arx[i][0], i);
    }
}//adjetivo
else if (arx[i][1] == "Adjetivo" )
{
    if (arx.Length - 1 == i)
    {
        es_adjetivo(arx[i][0], i);
        continue;
    }
    else if (arx[i + 1][1] == "Verbo" || arx[i + 1][1] ==
    "Preposicion" || arx[i + 1][1] == "Sustantivo"
    || arx[i + 1][1] == "Conjuncion" || arx[i + 1][1] ==
    "Adjetivo")
    {
        es_adjetivo(arx[i][0], i);
    }
}//adverbio
else if (arx[i][1] == "Adverbio" )
{
    if (arx.Length - 1 == i)
    {
        es_adverbio(arx[i][0], i);
        continue;
    }
```

```
                else if (arx[i + 1][1] == "Sustantivo" || arx[i + 1][1] ==
                "Articulo" || arx[i + 1][1] == "Adjetivo" ||
                arx[i + 1][1] == "Pronombre" || arx[i + 1][1] == "Conjuncion"
                || arx[i + 1][1] == "Verbo"){
                    es_adverbio(arx[i][0], i);
                }
            }//preposicion
            else if (arx[i][1] == "Preposicion" )
            {
                if (arx.Length - 1 == i)
                {
                    es_prep(arx[i][0], i);
                    continue;
                }
                else if (arx[i + 1][1] == "Verbo" || arx[i + 1][1] ==
                "Pronombre" || arx[i + 1][1] == "Sustantivo"
                || arx[i + 1][1] == "Adjetivo" || arx[i + 1][1] ==
                "Articulo")
                {
                    es_prep(arx[i][0], i);
                }
            }//conjuncion
            else if (arx[i][1] == "Conjuncion" )
            {
                if (arx.Length - 1 == i)
                {
                    continue;
                }
                else if(arx[i + 1][1] == "Verbo" || arx[i + 1][1] ==
                "Articulo"|| arx[i + 1][1] == "Sustantivo" || arx[i + 1][1]
                == "Pronombre" || arx[i + 1][1] == "Adjetivo"
                    || arx[i + 1][1] == "Preposicion")
                {
                    es_conj(arx[i][0], i);
                }
            }
        }
        oracioncompleta.Clear();
        for (int i = 0; i < arx.Length;i++)
        {
            oracioncompleta.Add(arx[i]);
        }
    }
    public string[][] MetodoBurbuja(string[][] mtr, int[] ordenmtr)//metodo
    burburja que ordena los elementos de la matriz dado con sus respectivos
    indices
    {
        int t;
        string[] aux1;
        int ax;
        matr_aux= new string[mtr.Length];
        for (int i = 0; i < mtr.Length - 1; i++)
        {
            for (int j = i+1; j < mtr.Length;j++ )
            {
                string pala = mtr[i][0];
                string[] pala1 = mtr[i][0].Split(' ');
                int tam = mtr[i][0].Split(' ').Length;
                if (ordenmtr[i] > ordenmtr[j])
                {
                    aux1 = mtr[i];
                    mtr[i] = mtr[j];
                    mtr[j] = aux1;
```

```
                ax = ordenmtr[i];
                ordenmtr[i]=ordenmtr[j];
                ordenmtr[j] = ax;
            }
        }
    }
    for (int i = 0; i < mtr.Length;i++)
    {
        matr_aux[i]=mtr[i][0];
    }
    return mtr;
}
void detectar_palabras(string[] grama, string[] gramaoriginal, int
tamGram)
{
    lpronombre = new List<string>();
    larticulo = new List<string>();
    lsustantivo = new List<string>();
    lverbo = new List<string>();
    ladjetivo = new List<string>();
    ladverbio = new List<string>();
    lconjuncion = new List<string>();
    lpreposicion = new List<string>();

    string[] aux;
    int cn=0;
    string[] palabra_aux=null;
    string error = null;
    try
    {
        ppronombre = particulo = psustantivo = padjetivo = pverbo = padverbio
        = pconjuncion = false;
        if (grama != null)
        {
            for (int i = 0; i < grama.Length; i++)
            {
                cn = 0;
                palabra = cc.recup_palabras(grama[i]);//palabra a
                comparar
                if (palabra.Length > 1)//aca se ve si existe la palabra
                                        segun el grama
                {
                    error = grama[i];//para almacenar por si ocurre error
                    con palabra comentarla y mostrar en pantalla
                    aux = palabra[0].Split();
                    for (int j = 0; j < aux.Length; j++)
                    {
                        if (oracion[j + i] == false)
                        {
                            oracion[j + i] = true;
                            if (!oracioncompleta.Contains(palabra))
                            {
                                oracioncompleta.Add(palabra);
                                indicesoracioncompleta.Add(j + i);
                            }
                            cn++;
                        }
                    }
                }
            }
        }
    }
    catch (Exception e)
```

```
    {
        MessageBox.Show("Error con Palabra: " + error);//palabra error
        agregarPalabra o = new agregarPalabra(error);
    }
}
void limpiarlabel()
{
    pronLabel.Text = "0 Pronombres";
    artLabel.Text="0 Articulos";
    sustLabel.Text="0 Sustantivos";
    verbLabel.Text="0 Verbos";
    adjLabel.Text="0 Adjetivos";
    advLabel.Text = "0 Adverbios";
    conjLabel.Text = "0 Conjunciones";
    prepLabel.Text = "0 Preposiciones";
}
void es_pronombre(string palabra, int i)
{
    cPron++;
    pronLabel.ForeColor = Color.Blue;
    pronLabel.Text = cPron.ToString()+" Pronombre";
    funcion_pasar(palabra, Color.Blue, i);
    lpronombre.Add(palabra);//con esto tenemos las palabras separadas en
    tipos
}

void es_articulo(string palabra, int i)
{
    cArt++;
    artLabel.ForeColor = Color.Red;
    artLabel.Text = cArt.ToString()+" Articulo";
    larticulo.Add(palabra);
    funcion_pasar(palabra, Color.Red, i);
}
void es_sustantivo(string palabra, int i)
{
    cSust++;
    sustLabel.ForeColor = Color.Green;
    sustLabel.Text = cSust.ToString()+" Sustantivo";
    lsustantivo.Add(palabra);
    funcion_pasar(palabra, Color.Green, i);
}
void es_verbo(string palabra, int i)
{
    cVerb++;
    verbLabel.ForeColor = Color.Purple;
    verbLabel.Text = cVerb.ToString()+ " Verbo";
    lverbo.Add(palabra);
    funcion_pasar(palabra, Color.Purple, i);
}
void es_adjetivo(string palabra, int i)
{
    cAdj++;
    adjLabel.ForeColor = Color.Cyan;
    adjLabel.Text = cAdj.ToString()+" Adjetivo";
    ladjetivo.Add(palabra);
    funcion_pasar(palabra, Color.Cyan, i);
}
void es_adverbio(string palabra, int i)
{
    cAdv++;
    advLabel.ForeColor = Color.Pink;
    advLabel.Text = cAdv.ToString()+" Adverbio";
```

```
        ladverbio.Add(palabra);
        funcion_pasar(palabra, Color.Pink, i);
    }
    void es_conj(string palabra, int i)
    {
        cConj++;
        conjLabel.ForeColor = Color.Orange;
        conjLabel.Text = cConj.ToString()+" Conjuncion";
        lconjuncion.Add(palabra);
        funcion_pasar(palabra, Color.Orange, i);
    }
    void es_prep(string palabra, int i)
    {
        cPrep++;
        prepLabel.ForeColor = Color.Gold;
        prepLabel.Text = cPrep.ToString() + " Preposicion";
        lpreposicion.Add(palabra);
        funcion_pasar(palabra, Color.Gold, i);
    }
    void funcion_pasar(string palabras, Color k, int i)
    {
        global_entero+= busca(this.oracionTextBox, palabras, k, i);
    }
    int busca(RichTextBox rtb, string palabra, Color color, int pospalabra)
    {
        string[] palabras = matr_aux;
        int tam = palabra.Length;
        int tamañoparcial = 0;
        Regex regExp = new Regex(palabra);
        foreach (Match match in regExp.Matches(rtb.Text))
        {
            rtb.Select(match.Index, match.Length);
            rtb.SelectionColor = color;
        }
        cont_palabras = tamañoparcial;
        return  tam;
    }
    //recup_palabras consulta con la base de datos las palabras de la oracion
    void recup_palabras(string palabrac)
    {
        string pal;
        string tpal;
        string ppal;
        try
        {
            MySqlDataReader rd ;
            MySqlCommand inst =new MySqlCommand("SELECT * FROM palabras where
            palabra = '" + palabrac + "'", cc.conn);
            if(inst!=null){
                rd = inst.ExecuteReader();
                rd.Read();
                pal = rd["palabra"].ToString();
                tpal = rd["tipoPalabra"].ToString();
                ppal = rd["polaridad"].ToString();
                rd.Dispose();
            }

        }catch(MySqlException e){
        }
    }
    private void button1_Click(object sender, EventArgs e)
    {
        Palabras o = new Palabras(ladjetivo, ladverbio, lsustantivo, lverbo);
```

```
            o.Show();
        }
        private void agregarPalabrasToolStripMenuItem_Click(object sender,
        EventArgs e)
        {
            agregarPalabra o = new agregarPalabra();
            o.Show();
        }
    }
}
```

AgregarPalabras.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace sentiment
{
    public partial class agregarPalabra : Form
    {
        public agregarPalabra()
        {
            InitializeComponent();
            cc = new cConex();
        }
        public agregarPalabra(string palabra)
        {
            InitializeComponent();
            textBox1.Text = palabra;
            cc= new cConex();
        }
        cConex cc;
        void existe_palabra()
        {
            cc.conectar();
            string pala = textBox1.Text;
            string tpala = comboBox2.SelectedItem.ToString();
            string ppala = comboBox1.SelectedItem.ToString();
            int c = cc.sql("INSERT INTO palabras(palabra, tipoPalabra, polaridad)
                           VALUES ( '" +
                           pala + "', '" +
                           tpala + "', '" +
                           ppala + "')");
            MessageBox.Show("Palabra ingresada");
            this.Close();

        }
        private void addButton_Click(object sender, EventArgs e)
        {
            existe_palabra();
        }
        private void button1_Click(object sender, EventArgs e)
        {
            existe_palabra();
        }
    }
}
```

Palabras.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace sentiment
{
    public partial class Palabras : Form
    {
        public Palabras()
        {
            InitializeComponent();
        }
        public Palabras(List<string> adj, List<string> adv, List<string> sust,
                        List<string> verb)
        {
            InitializeComponent();
            adjListBox.DataSource = adj;
            advListBox.DataSource = adv;
            sustListBox.DataSource = sust;
            verbListBox.DataSource = verb;
        }
    }
}
```

CLASES

cConex.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using MySql.Data.MySqlClient;
using System.IO;
using System.Runtime.Serialization.Formatters.Binary;
using System.Windows.Forms;
namespace sentiment
{
    class cConex
    {
        public MySqlConnection conn;
        conteoPalabras cp;
        string conexionString, conec;
        public List<string[]> orsql, oracion_completa;
        public List<string> auxpron, lpronombre, auxar, larticulo, auxs,
        lsustantivo, auxv, lverbo, auxadj, ladjetivo, auxadv, ladverbio,
        auxcon, lconjuncion, auxprep, lpreposicion;//variables donde se
        almacenaran las palabras
        public bool conectar()
        {
            try
            {
                conn= new MySqlConnection();
```

```
            oracion_completa = new List<string[]>();
            conexionString = "Server=127.0.0.1; Database=mydb; Uid=root;
                              password=root;";
            conn.ConnectionString = conexionString;
            conn.Open();
            return true;
            //la conexion se aperturo
        }catch(MySqlException e){
            return false;
        }
    }
    public string[] recup_palabras(string palabrac)
    {
        string pal;
        string tpal;
        string ppal;
        string[] palabra = new string[3];
        string[] palabranoExiste = new string[1];
        MySqlDataReader rd=null;
        MySqlCommand inst;
        try
        {
            inst = new MySqlCommand();
            inst= conn.CreateCommand();
            inst.CommandText = "SELECT * FROM palabras where palabra = '" +
                               palabrac + "'";
            //aca se hace el filtro
            rd = inst.ExecuteReader();
            int i = 0;
            while (rd.Read())
            {
                string auv = rd["palabra"].ToString();
                if (palabrac.ToLower() == rd["palabra"].ToString())
                {
                    palabra[0] = rd["palabra"].ToString();
                    palabra[1] = rd["tipoPalabra"].ToString();
                    palabra[2] = rd["polaridad"].ToString();
                    break;
                }
            }

            if (!rd.HasRows)
            {
                palabranoExiste[0] = palabrac;
                rd.Dispose();
                rd.Close();
                return palabranoExiste;
            }
            rd.Dispose();
            rd.Close();
            return palabra;

        }
        catch (MySqlException e){return null;}}
    public int sql(string sql1)
    {
        int count=0;
        try
        {
            MySqlCommand command = conn.CreateCommand();
            command.CommandText = sql1;
            command.ExecuteNonQuery();
            count =Convert.ToInt32(command.ExecuteScalar());
```

```
        }catch(MySqlException e){
        }
        return count;
    }
    public List<string[]> recup_oraciones_positivas()
    {
        string pal;
        string tpal;
        string ppal;
        string[] oracion = new string[2];
        string[] palabranoExiste = new string[1];
        MySqlDataReader rd = null;
        MySqlCommand inst;
        orsql = new List<string[]>();
        try
        {
            inst = new MySqlCommand();
            inst = conn.CreateCommand();
            inst.CommandText = "SELECT * FROM oraciones where polaridad =
                                \"Positivo\" ";
            rd = inst.ExecuteReader();
            int i = 0;
            while (rd.Read())
            {
                oracion[0] = rd["oraciones"].ToString();
                oracion[1] = rd["polaridad"].ToString();
                cp = new conteoPalabras(oracion[0]);//esto se queda por que
                 hay un nuevo analisis cada oracion
                //recuperamos todos las palabras de todas las oraciones
                orsql = cp.d_oracioncompleta();
                oracion_completa.AddRange(orsql);
                cp = null;//y todo debe comenzar en cero
            }
            rd.Dispose();
            rd.Close();
            return oracion_completa;
        }
        catch (MySqlException e)
        {
            return null;
        }
    }
    public List<string[]> recup_oraciones_negativas()
    {
        string pal;
        string tpal;
        string ppal;
        string[] oracion = new string[2];
        string[] palabranoExiste = new string[1];
        MySqlDataReader rd = null;
        MySqlCommand inst;
        orsql = new List<string[]>();
        try
        {
            inst = new MySqlCommand();
            inst = conn.CreateCommand();
            inst.CommandText = "SELECT * FROM oraciones where polaridad =
                                \"Negativo\" ";
            rd = inst.ExecuteReader();
            int i = 0;
            while (rd.Read())
            {
                oracion[0] = rd["oraciones"].ToString();
```

```
                    oracion[1] = rd["polaridad"].ToString();
                    cp = new conteoPalabras(oracion[0]);//esto se queda por que
                    hay un nuevo analisis cada oracion
                    //recuperamos todos las palabras de todas las oraciones
                    orsql = cp.d_oracioncompleta();
                    oracion_completa.AddRange(orsql);
                    cp = null;//y todo debe comenzar en cero
                }
                rd.Dispose();
                rd.Close();
                return oracion_completa;
            }
            catch (MySqlException e)
            {
                return null;
            }
        }
        public List<string[]> recup_oraciones()
        {
            string pal;
            string tpal;
            string ppal;
            string[] oracion = new string[2];
            string[] palabranoExiste = new string[1];
            MySqlDataReader rd = null;
            MySqlCommand inst;
            orsql = new List<string[]>();
            try
            {
                inst = new MySqlCommand();
                inst = conn.CreateCommand();
                inst.CommandText = "SELECT * FROM oraciones";
                rd = inst.ExecuteReader();
                int i = 0;
                while (rd.Read())
                {
                    oracion[0] = rd["oraciones"].ToString();
                    oracion[1] = rd["polaridad"].ToString();
                    cp = new conteoPalabras(oracion[0]);//esto se queda por que
                    hay un nuevo analisis cada oracion
                    //recuperamos todos las palabras de todas las oraciones
                    orsql = cp.d_oracioncompleta();
                    cp=null;//y todo debe comenzar en cero
                }
                rd.Dispose();
                rd.Close();
                return orsql;
            }
            catch (MySqlException e)
            {
                return null;
            }
        }
    }
}
```

cBayes.cs

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.ComponentModel;
using System.Data;

namespace sentiment
{
    class cBayes
    {
        List<string[]> orNe, orPo, orPrue, orAux,orAux2, orAux3;
        List<double> frecuenciasPositivas, frecuenciasNegativas;
        int cop, con;//variable donde se alamcena las cantidades de oraciones
        public cBayes()
        {
            frecuenciasNegativas = new List<double>();
            frecuenciasPositivas = new List<double>();
        }
        public void oraciones(List<string[]> oracionNegativa, List<string[]>
                              oracionPositiva, List<string[]> oracionPrueba)
        {
            List<double> lbayesiano = new List<double>();
            orPo = oracionPositiva;
            orNe = oracionNegativa;
            orPrue = oracionPrueba;
            ordenar_nlista_palabras_cantidad();
            calcular_frecuencias();
            bayes_aplicado();
            lbayesiano.Add(bayesN);
            lbayesiano.Add(bayesP);
        }
        double pOrPo, pOrNe, condP = 0, condN = 0, probTotal, bayesP, bayesN;
        void bayes_aplicado()
        {   //necesitamos las frecuencias, cantidad oraciones, probabilidad
            oraciones negativas, probabilidad oraciones positivas
            //probabilidad total oraciones
            //o->Oraciones y p->Polaridad
            //formula bayes P(o|p)=P(o)*P(p|o)/P(oracionestotales)
            //P(o)-> probabilidad de oraciones ya sea positiva o negativa
            //P(p|o)->probabilidad de las frecuencias de las palabras
              multiplicados
            //P(oracionestotales)->probabilidad total
            double suma_oraciones = con + cop;
            //P(o) probabilidad apriori
            pOrPo = cop / suma_oraciones;
            pOrNe = con / suma_oraciones;
            //P(p|o) probabilidad condicional
            for (int i = 0; i < frecuenciasPositivas.Count;i++ )
            {
                if (!Double.IsNaN(frecuenciasPositivas.ElementAt(i)))//ver esto
                    condP = frecuenciasPositivas.ElementAt(i) + condP;
            }
            for (int i = 0; i < frecuenciasNegativas.Count; i++)
            {
                if (!Double.IsNaN(frecuenciasNegativas.ElementAt(i)))
                    condN = frecuenciasNegativas.ElementAt(i) + condN;
            }
            //P(oracionestotales) probabilidad total
            probTotal = (pOrPo * condP) + (pOrNe * condN);
            //probabilidad a posteriori o bayes P(o|p)
            bayesN = (pOrNe * condN) / probTotal;
            bayesP = (pOrPo * condP) / probTotal;
```

```
        if (bayesN > bayesP){
            MessageBox.Show("La oracion es Negativa\nPN: "+bayesN+"\nPP:
                            "+bayesP);
        }else{
            MessageBox.Show("La oracion es Positiva\nPN: " + bayesN + "\nPP:
                            " + bayesP);
        }
    }
     void ordenar_nlista_palabras_cantidad()
     {
         orAux = new List<string[]>();
         orAux2 = new List<string[]>();
         orAux3 = new List<string[]>();
         orPrue.Sort((s, t) => String.Compare(s[0], t[0]));
         for (int i = 0; i <orPrue.Count;i++ )
         {
             orAux = orNe.FindAll(delegate(string[] s) { return s[0] ==
                                                 orPrue.ElementAt(i)[0]; });
             orAux2.AddRange(orAux);
             orAux = orPo.FindAll(delegate(string[] s) { return s[0] ==
                                                 orPrue.ElementAt(i)[0]; });
             orAux3.AddRange(orAux);
         }
         orNe = orAux2;
         orPo = orAux3;
     }
     List<string[]> listas(List<string[]> mayor, List<string[]> menor)
     {
         List<string[]> may = new List<string[]>(mayor);
         List<string[]> men = new List<string[]>(menor);
         string[] var_aux = null; ;
         for (int i = 0; i<mayor.Count;i++ )
         {
             if (men.Find(delegate(string[] s) { return s[0] ==
                 may.ElementAt(i)[0]; })==null)
             {
                 var_aux = may.ElementAt(i).ToArray();//es necesario clonar
                 los elementos para poder evitar que se haga
                 var_aux[3] = "0";//referencia al momento de modificar los
                 elementos
                 men.Add(var_aux);
             }
         }
         men.Sort((s, t) => String.Compare(s[0], t[0]));
         return men;
     }
     void calcular_frecuencias()
     {
         double suma, frecuenciaP,frecuenciaN;
         int cli1 = orNe.Count;
         int cli2 = orPo.Count;
         if (cli1 > cli2)
         {
             orPo=listas(orNe, orPo);
             if (orPo.Count > orNe.Count)
             {
                 orNe = listas(orPo, orNe);
             }
         }
         orNe = ((from s in orNe select s).Distinct()).ToList();
         orPo = ((from s in orPo select s).Distinct()).ToList();
         if(orPo.Count > orNe.Count)
         {
```

```
                orNe = listas(orPo, orNe);
                if (orNe.Count > orPo.Count)
                {
                    orPo = listas(orNe, orPo);
                }
            }
            orNe = ((from s in orNe select s).Distinct()).ToList();
            orPo = ((from s in orPo select s).Distinct()).ToList();

            for (int i=0; i <orPo.Count;i++ )
            {
                suma = Convert.ToDouble(orPo.ElementAt(i)[3]) +
                       Convert.ToDouble(orNe.ElementAt(i)[3]);
                frecuenciaP = Convert.ToDouble(orPo.ElementAt(i)[3]) / suma;
                frecuenciaN = Convert.ToDouble(orNe.ElementAt(i)[3]) / suma;

                frecuenciasNegativas.Add(frecuenciaN);
                frecuenciasPositivas.Add(frecuenciaP);
            }
            cantidad_oraciones();
        }
        void cantidad_oraciones()
        {
            cConex co = new cConex();
            co.conectar();
            con = co.sql("SELECT count(*) FROM oraciones where
                         polaridad='Negativo'");//cantidad oraciones negativas
            cop = co.sql("SELECT count(*) FROM oraciones where
                         polaridad='Positivo'");//cantidad oraciones positivas
        }
        void quitar_palabras_oracionprueba()//ver luego
        {
            orAux = new List<string[]>();
            for (int i = 0; i < orPrue.Count; i++)
            {
                if (orPrue.ElementAt(0)[1] == "Articulo" ||
                    orPrue.ElementAt(0)[1] == "Pronombre"
                    || orPrue.ElementAt(0)[1] == "Conjuncion" ||
                     orPrue.ElementAt(0)[1] == "Preposicion")
                {
                    orPrue.RemoveAt(i);
                }
            }
        }
        public DataTable ConvertToDataTable<T>(IList<T> data)
        {
            PropertyDescriptorCollection properties =
            TypeDescriptor.GetProperties(typeof(T));
            DataTable table = new DataTable();
            foreach (PropertyDescriptor prop in properties)
            table.Columns.Add(prop.Name,
            Nullable.GetUnderlyingType(prop.PropertyType) ?? prop.PropertyType);
            foreach (T item in data)
            {
                DataRow row = table.NewRow();
                foreach (PropertyDescriptor prop in properties)
                    row[prop.Name] = prop.GetValue(item) ?? DBNull.Value;
                table.Rows.Add(row);
            }
            return table;
        }
    }
}
```

cConversionListas.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Runtime.Serialization.Formatters.Binary;
using System.IO;
namespace sentiment
{
    [Serializable()]
    class cConversionListas
    {
        public List<List<string[]>> lista_dat { get; set; }
        public cConversionListas(List<List<string[]>> laux)
        {
            this.lista_dat = laux;
        }
    }
}
```

consultaOraciones.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
using System.Runtime.Serialization.Formatters.Binary;
namespace sentiment
{
    class consultaOraciones
    {
        cConex conexion;
        List<string[]> oraciones, oraciones1, oraciones2;
        List<List<string[]>> laux,lauxp,lauxn;
        public List<List<string[]>> consulta_oraciones()
        {
            laux = new List<List<string[]>>();
            lauxn = new List<List<string[]>>();
            lauxp = new List<List<string[]>>();
            conexion = new cConex();//tienes que haber un nuevo objeto por cada
                                    consulta positiva o negativa
            conexion.conectar();
            //en esta parte se guarda los datos en un archivo serializable para
              poder mayor facilidad de leerlos al momento
            //de haber analizado los datos
            //lista negativa si no existe el archivo se genera pero si existe
              simplemente se lee
            if (!File.Exists(@"listaN.bin"))
            {
                oraciones1 = conexion.recup_oraciones_negativas();
                lauxn.Add(vantidad_palabras(oraciones1));
                clista = new cConversionListas(lauxn);
                leer_datos(@"listaN.bin");
                grabar_datos(@"listaN.bin");
            }
            else
            {
                clista = new cConversionListas(lauxn);
```

```
            leer_datos(@"listaN.bin");
            lauxn.AddRange(clista.lista_dat);
        }
        laux.AddRange(lauxn);
        conexion = new cConex();
        conexion.conectar();
        if (!File.Exists(@"listaP.bin"))
        {
            oraciones2 = conexion.recup_oraciones_positivas();
            lauxp.Add(vantidad_palabras(oraciones2));//positivas oraciones
            clista = new cConversionListas(lauxp);
            leer_datos(@"listaP.bin");
            grabar_datos(@"listaP.bin");
        }
        else
        {
            clista = new cConversionListas(lauxp);
            leer_datos(@"listaP.bin");
            lauxp.AddRange(clista.lista_dat);
        }
        laux.AddRange(lauxp);
        return laux;
    }
    cConversionListas clista;
    void leer_datos(string FileName)
    {
        if (File.Exists(FileName))
        {
            Stream TestFileStream = File.OpenRead(FileName);
            BinaryFormatter deserializer = new BinaryFormatter();
            clista =
           (cConversionListas)deserializer.Deserialize(TestFileStream);
            TestFileStream.Close();
        }
    }
    void grabar_datos(string FileName)
    {
        Stream TestFileStream = File.Create(FileName);
        BinaryFormatter serializer = new BinaryFormatter();
        serializer.Serialize(TestFileStream, clista);
        TestFileStream.Close();

    }
    public List<string[]> vantidad_palabras(List<string[]> or)
    {
        List<string[]> todo;
        List<string[]> palabPo;
        or.Sort((s, t) => String.Compare(s[0], t[0]));//ordena
        alfanumericamente la lista de elementos
        int cant;
        palabPo = new List<string[]>();
        string[] k;
        for (int i = 0; i < or.Count;i++ )
        {
                k = new string[4];
                todo = or.FindAll(delegate(string[] s) { return s[0] ==
                or.ElementAt(i)[0]; });//encontramos todos los parecidos
                cant = todo.Count;
                i = i + cant - 1;//este contador esta para evitar las
                palabras repetidas una cant(variable) de veces
                k[0]=or.ElementAt(i)[0];//palabra
                k[1] = or.ElementAt(i)[1];//Tipo palabra
                k[2] = or.ElementAt(i)[2];//polaridad
```

```
                    k[3] = cant.ToString();//cantidad palabra
                    if (k[1] != "Articulo" && k[1] != "Pronombre" && k[1] !=
                    "Conjuncion" && k[1] != "Preposicion")
                    {
                        palabPo.Add(k);
                    }
                }
                return palabPo;
            }
        }
    }
```

conteoPalabras.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace sentiment
{
    class conteoPalabras
    {
        bool[] palabradisponible, oracion;
        bool contar = false, conexiond,ppronombre, particulo, psustantivo,
        padjetivo, pverbo, padverbio, pconjuncion, ppreposicion;
        string[] matr_aux,palabra, grama2, grama3, grama1, grama4,
        grama5;//gramas a comparar
        string[][] cgrama;
        public string oracion_global;//oracion que se detecytara al hacer el sql
        int cArt, cPron, cPrep, cAdj, cSust, cVerb, cAdv, cConj;//para contar las
        palabras que tipos existen
        int cont_palabras, global_entero;
        char[] delimiters;
        List<string> lpronombre, larticulo, lsustantivo, lverbo, ladjetivo,
        ladverbio, lconjuncion, lpreposicion;//variables donde se almacenaran las
                                             palabras
        List<string[]> oracioncompleta;//variable importante donde se almacenan
        todas las palabras con sus respectivos categorías y polaridades
        List<int> indicesoracioncompleta;//variable importante
        cConex cc;
        List<cPalabra> lp;

       public conteoPalabras(string oracionSQL)//constructor que recibira de n
       iteraciones las oraciones para
       //filtrarlas y extraer las palabras y poder tener las palabras positivas y
         negativas asi como la frecuencia
       //de estas y  luego poder asnañizar en otra clase con el teorema de bayes
        {
            cc = new cConex();
            conexiond = cc.conectar();
            oracioncompleta = new List<string[]>();//variable donde se
            almacenara las palabras detectadas para su posterior refinamiento
            indicesoracioncompleta = new List<int>();
            oracion_global = oracionSQL;
            gramas();
        }

        void false_funcion(int tamaOra)//funcion que ayudara a controlar las
                                       palabras detectadas con las
        {//no detectadas
```

```
        oracion = new bool[tamaOra];
        for (int i = 0; i < oracion.Length; i++)
        {
            oracion[i] = false;//todas las palabras inicialmente estan en
                               false
        }

    }
    void limpiar_sigp_orsql(string[] fragmentosOracion)//tiene la finalidad
    de limpiar de signos de puntuacion para la correcta deteccion en la
    oracion
    {
        string or = "";
        for (int i = 0; i < fragmentosOracion.Length; i++)
        {
            or += fragmentosOracion[i] + " ";
        }
        oracion_global = or.ToLower();
    }

    void gramas()//funcion que servira para separar las gramas de varios
                 tipos
    {

        oracioncompleta.Clear();
        indicesoracioncompleta.Clear();//limpiamos siempre las varaibles
        string[][] gp, gp2, gp3;
        ngramas arr = new ngramas();
        delimiters = new char[] { '\r', '\n', ' ', ',', ':', '.', '?', '¿',
                                  '!', '¡' };
        string[] fragmentosOracion = oracion_global.Split(delimiters,
                                     StringSplitOptions.RemoveEmptyEntries);
        limpiar_sigp_orsql(fragmentosOracion);
        int c = 1;
        //gramas para comparar con base de datos la comparacion es hasta 2
          gramas
        cPron = 0; cArt = 0; cPrep = 0; cAdv = 0; cAdj = 0; cSust = 0;
        cVerb = 0; cConj = 0;
        grama1 = arr.ngr(1, fragmentosOracion);
        grama2 = arr.ngr(2, fragmentosOracion);
        grama3 = arr.ngr(3, fragmentosOracion);
        if (fragmentosOracion.Length == 2)
        {
            //    cgrama = arr.complemento(2, fragmentosOracion);
        }
        if (fragmentosOracion.Length == 3)
        {
            //  cgrama = arr.complemento(3, fragmentosOracion);
        }
        if (fragmentosOracion.Length == 4)
        {
            //  cgrama = arr.complemento(4, fragmentosOracion);
        }
        false_funcion(fragmentosOracion.Length);//se envia el tamaño de la
        oracion original para poder asignarles true en una etapa posterior
        detectar_palabras(grama3, grama1, 3);
        detectar_palabras(grama2, grama1, 2);
        detectar_palabras(grama1, grama1, 1);
        especificacion_palabra();
    }
    void detectar_palabras(string[] grama, string[] gramaoriginal, int
                           tamGram)
    {
```

```
        lpronombre = new List<string>();
        larticulo = new List<string>();
        lsustantivo = new List<string>();
        lverbo = new List<string>();
        ladjetivo = new List<string>();
        ladverbio = new List<string>();
        lconjuncion = new List<string>();
        lpreposicion = new List<string>();

        string[] aux;
        int cn = 0;
        string[] palabra_aux = null;
        string error = null;
        try
        {
            ppronombre = particulo = psustantivo = ppreposicion = padjetivo =
            pverbo = padverbio = pconjuncion = false;
            if (grama != null)
            {
                for (int i = 0; i < grama.Length; i++)
                {
                    cn = 0;
                    palabra = cc.recup_palabras(grama[i]);//palabra a
                                                          comparar
                    if (palabra.Length > 1)//aca se ve si existe la palabra
                                           segun el grama
                    {
                        error = grama[i];//para almacenar por si ocurre error
                        con palabra comentarla y mostrar en pantalla
                        aux = palabra[0].Split();
                        for (int j = 0; j < aux.Length; j++)
                        {
                            if (oracion[j + i] == false)
                            {
                                oracion[j + i] = true;
                                if (!oracioncompleta.Contains(palabra))
                                {
                                    oracioncompleta.Add(palabra);
                                    indicesoracioncompleta.Add(j + i);
                                }
                                cn++;
                            }
                        }
                    }
                }
            }
        }
        catch (Exception e)
        {
        }
    }
    void especificacion_palabra()//en esta funcion se ordenara las palabras
                                 tal como se ingreso en la oracion
    {//y no como estaban en el ordenal haber sido detectadas por los gramas
        string[][] arrax = oracioncompleta.ToArray();//asignamos los valores
                                                     almacenados en un array
        int[] ormtr = indicesoracioncompleta.ToArray();//tomamos en cuenta
        los indices que facilitaran el ordenamiento en la oracion
        arrax = MetodoBurbuja(arrax, ormtr);//como resultado lo sometemos a
        un ordenamiento burbuja  dando el ordenamiento esperado
        orden_correcto_palabras(arrax);
    }
    public string[][] MetodoBurbuja(string[][] mtr, int[] ordenmtr)//metodo
```

```
burburja que ordena los elementos de la matriz dado con sus respectivos
indices
{
    int t;
    string[] aux1;
    int ax;
    matr_aux = new string[mtr.Length];
    for (int i = 0; i < mtr.Length - 1; i++)
    {
        for (int j = i + 1; j < mtr.Length; j++)
        {
            string pala = mtr[i][0];
            string[] pala1 = mtr[i][0].Split(' ');
            int tam = mtr[i][0].Split(' ').Length;
            if (ordenmtr[i] > ordenmtr[j])
            {
                aux1 = mtr[i];
                mtr[i] = mtr[j];
                mtr[j] = aux1;

                ax = ordenmtr[i];
                ordenmtr[i] = ordenmtr[j];
                ordenmtr[j] = ax;
            }
        }
    }
    for (int i = 0; i < mtr.Length; i++)
    {
        matr_aux[i] = mtr[i][0];
    }
    return mtr;
}
void orden_correcto_palabras(string[][] arx)//en esta funcion
procederemos a detectar correctamente las palabras
{// de esta manera podremos tener en cuenta las reglas al momento que se
    escriben en una oracion tanto para
    //diferenciar sustantivos, adjetivos, adverbios, verbos, etc...
    for (int i = 0; i < arx.Length; i++)
    {
        //pronombre
        if (arx[i][1] == "Pronombre")//reglas de identificacion de las
                                     palabras
        {//aca es el problema porque no restringe si es verdadero o
           falso
            if (arx.Length - 1 == i)
            {
                es_pronombre(arx[i][0], i);
                continue;
            }
            else if (arx[i + 1][1] == "Verbo" || arx[i + 1][1] ==
            "Conjuncion"|| arx[i + 1][1] == "Adverbio")
            {
                es_pronombre(arx[i][0], i);
            }

        }
        //articulo
        else if (arx[i][1] == "Articulo")
        {
            if (arx.Length - 1 == i)
            {
                es_articulo(arx[i][0], i);
                continue;
```

```
        }
        else if (arx[i + 1][1] == "Sustantivo" || arx[i + 1][1] ==
                 "Adjetivo")
        {
            es_articulo(arx[i][0], i);
        }
    }
    //sustantivo
    else if (arx[i][1] == "Sustantivo")
    {
        if (arx.Length - 1 == i)
        {
            es_sustantivo(arx[i][0], i);
            continue;
        }
        else if (arx[i + 1][1] == "Verbo" || arx[i + 1][1] ==
       "Adjetivo" ||arx[i + 1][1] == "Adverbio" || arx[i + 1][1] ==
       "Preposicion" || arx[i + 1][1] == "Conjuncion")
        {
            es_sustantivo(arx[i][0], i);
        }
    }
    //verbo
    else if (arx[i][1] == "Verbo")
    {
        if (arx.Length - 1 == i)
        {
            es_verbo(arx[i][0], i);
            continue;
        }
        else if (arx[i + 1][1] == "Sustantivo" || arx[i + 1][1] ==
        "Adjetivo" ||arx[i + 1][1] == "Adverbio" || arx[i + 1][1] ==
        "Conjuncion" || arx[i + 1][1] == "Articulo"
        || arx[i + 1][1] == "Pronombre" || arx[i + 1][1] == "Verbo"
        || arx[i + 1][1] == "Preposicion")
        {
            es_verbo(arx[i][0], i);
        }
    }
    //adjetivo
    else if (arx[i][1] == "Adjetivo")
    {
        if (arx.Length - 1 == i)
        {
            es_adjetivo(arx[i][0], i);
            continue;
        }
        else if (arx[i + 1][1] == "Verbo" || arx[i + 1][1] ==
        "Preposicion" || arx[i + 1][1] == "Sustantivo"
        || arx[i + 1][1] == "Conjuncion" || arx[i + 1][1] ==
        "Adjetivo")
        {
            es_adjetivo(arx[i][0], i);
        }
    }
    //adverbio
    else if (arx[i][1] == "Adverbio")
    {
        if (arx.Length - 1 == i)
        {
            es_adverbio(arx[i][0], i);
            continue;
        }
```

```
                else if (arx[i + 1][1] == "Sustantivo" || arx[i + 1][1] ==
                "Articulo" || arx[i + 1][1] == "Adjetivo" ||
                arx[i + 1][1] == "Pronombre" || arx[i + 1][1] == "Conjuncion"
                || arx[i + 1][1] == "Verbo")
                {
                    es_adverbio(arx[i][0], i);
                }
            }
            //preposicion
            else if (arx[i][1] == "Preposicion")
            {
                if (arx.Length - 1 == i)
                {
                    es_prep(arx[i][0], i);
                    continue;
                }
                else if (arx[i + 1][1] == "Verbo" || arx[i + 1][1] ==
                "Pronombre" || arx[i + 1][1] == "Sustantivo"
                || arx[i + 1][1] == "Adjetivo" || arx[i + 1][1] ==
                "Articulo")
                {
                    es_prep(arx[i][0], i);
                }
            }
            //conjuncion
            else if (arx[i][1] == "Conjuncion")
            {
                if (arx.Length - 1 == i)
                {
                    continue;
                }
                else if (arx[i + 1][1] == "Verbo" || arx[i + 1][1] ==
                "Articulo"|| arx[i + 1][1] == "Sustantivo" || arx[i + 1][1]
                == "Pronombre" || arx[i + 1][1] == "Adjetivo"
                    || arx[i + 1][1] == "Preposicion")
                {
                    es_conj(arx[i][0], i);
                }
            }
        }
        oracioncompleta.Clear();
        for (int i = 0; i < arx.Length; i++)
        {
            oracioncompleta.Add(arx[i]);
        }
    }
    void es_pronombre(string palabra, int i)
    {
        cPron++;
        lpronombre.Add(palabra);//con esto tenemos las palabras separadas en
                                tipos
    }
    void es_articulo(string palabra, int i)
    {
        cArt++;
        larticulo.Add(palabra);
    }
    void es_sustantivo(string palabra, int i)
    {
        cSust++;
        lsustantivo.Add(palabra);
    }
    void es_verbo(string palabra, int i)
```

```
        {
            cVerb++;
            lverbo.Add(palabra);
        }
        void es_adjetivo(string palabra, int i)
        {
            cAdj++;
            ladjetivo.Add(palabra);
        }
        void es_adverbio(string palabra, int i)
        {
            cAdv++;
            ladverbio.Add(palabra);
        }
        void es_conj(string palabra, int i)
        {
            cConj++;
            lconjuncion.Add(palabra);
        }
        void es_prep(string palabra, int i)
        {
            cPrep++;
            lpreposicion.Add(palabra);
        }
        public List<string> d_pronom()
        {
            return lpronombre;
        }
        public List<string> d_arti()
        {
            return larticulo;
        }
        public List<string> d_sust()
        {
            return lsustantivo;
        }
        public List<string> d_adjeti()
        {
            return ladjetivo;
        }
        public List<string> d_verbo()
        {
            return lverbo;
        }
        public List<string> d_adver()
        {
            return ladverbio;
        }
        public List<string> d_conj()
        {
            return lconjuncion;
        }
        public List<string> d_prep()
        {
            return lpreposicion;
        }
        public List<string[]> d_oracioncompleta()
        {
            return oracioncompleta;
        }
    }
}
```

ngramas.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace sentiment
{
    class ngramas
    {   //tipo de derivacion de los ngramas pero este se llamaria narreglosgramas
        string[] k,arre;
        int c, tamañodelgrama;
        int count;
        public string[] ngr(int tamañoGrama, string[] arreglo)
        {
            if (arreglo.Length < tamañoGrama)
            {
                return null;
            }
            int tamañoriginal = arreglo.Length;
            int tam=tamañoGrama-1;
            k= new string[tamañoriginal-tam];//tamaño de arreglo por gramas
            c = 0;
            count = 0;
            arre=arreglo;
            tamañodelgrama = tamañoGrama;
            for (int i = 0; i < arreglo.Length-tam;i++ )
            {
                recur(c,i);
                count++;
                c = 0;
            }
            return k;
        }
        void recur(int cuenta,int indice)
        {
            if(c<tamañodelgrama-1){
                k[count] += arre[indice+c]+" ";
                c++;
                recur(c+count,indice);
            }
            else if(c<tamañodelgrama){
                k[count] += arre[indice+c];
            }
        }
        public string[][] complemento(int numgramas,string[] oracion)
        {
            ngramas gra = new ngramas();
            string[][] gramas = new string[numgramas][];
            int tamañomatrizoraciones = 0;
            //esto para extraer la derivada de gramas desde 1 hasta n donde el n
              maximo recomendado es 4 debido a que
            //no existen palabras compuestas de mas de 4 palabras simples
            for (int i = 0; i < numgramas; i++)
            {
                gramas[i] = gra.ngr(i + 1, oracion);
            }
            int n = numgramas - 1;
            while (n > 0)
            {
                tamañomatrizoraciones += gramas[n].Length;//se determina el
```

```
            tamaño de nuemro de filas de la matriz
            n--;
        }
        int num=numgramas-1,tamñoinicil = gramas[1].Length;
        string[][] matriz = new string[tamañomatrizoraciones][];
        int controlador = 0, tama = 0;
        for (int i = 0; i < tamañomatrizoraciones;i++ )
        {
            if (controlador < tamñoinicil)
            {
                matriz[i] = new string[tamñoinicil];
                controlador++;
            }
            else
            {
                tamñoinicil--;
                controlador = 0;
                i--;
            }
        }
        int c=0,co=0,con=0,cont=0,tamnito=0,porsi=1;
        int tamaño= gramas[1].Length;//necesario y esencial el tamaño del 2
                                     arreglo en la matriz
        string[][]aux = gramas;
        //no tocar esta funcion de esta funcion depende toda slas oraciones
          que salgan en adelante
        while(c<tamañomatrizoraciones){
            if (c < tamaño)
            {
                matriz[c][con] = gramas[co + 1][con];//para colocar las
                                                     palabras compuestas
                //no borrar matriz[][]
                cont = 0;
                for (int i = 0; i < gramas[co + 1].Length;i++)
                {
                    if (matriz[c][i] == null)
                    {
                        matriz[c][i] = gramas[0][cont];
                        cont += porsi;
                    }
                    else
                    {
                        cont += co+2;
                    }
                }
                con++;
            }
            else if (c >= gramas[1].Length)
            {
                co++;
                con = 0;
                tamaño += gramas[co+1].Length;
                c--;
            }
            c++;
        }
        return matriz;
    }
}
}
```