Tarea 2
IPLN
Grupo 9

Valentina Da Silva 5.113.011-5
Victor Díaz 4.295.936-0
Leonardo Clavijo 5.054.830-5

November 3, 2014

Abstract

En este presente documento se procede a la explicación, tanto de la implementación de la aplicación, así como los parámetros de configuración necesarios para la ejecución de la misma.

Contents

1 Introduction

En la presente aplicación se trabajó en base a dos metodologías de procesamiento de datos, mediante el uso de WebServices y una aplicación local(LAN) del servicio Freeling mediante sockets. Por lo tanto en la implementación contamos con dos modos de ejecución, esto puede ser útil para el caso de los usuarios que no cuenta con un servidor Freeling local, pero sí mediante una arquitectura orientada a servicios. Al pensar en ambos casos, tenemos la ventaja de que si no contamos con alguno de ellos, podemos utilizar el otro método.

2 Requerimientos

2.1 Librerías

Para la correcta ejecución de la aplicación es necesario contar con el siguiente conjunto de librerías:

- suds : Para utilizar la aplicación mediante Webservices, se trata de una librería que permite la creación de estructuras orientadas a webservices, utilizando el protocolo SOAP.
- mathplotlib: Se utiliza para realizar las gráficas pertientes a los requerimientos del laboratorio.
- openpyxl : Librería utilizada para la lectura de archivos xls, para el caso en particular el archivo que contiene los comentarios a ser procesados.

Los mismas pueden ser agregadas fácilmente utilizando pip o easy-install.

- suds : pip install suds-jurko — easy_install suds-jurko.
- mathplotlib : pip install matplotlib — easy_install -m matplotlib.
- openpyxl : pip install openpyxl — easy_install openpyxl.

2.2 Freeling

Para ejecutar la aplicación utilizando un servidor local, es necesario contar con la aplicación Freeling y ejecutarla en modo servidor, para ello se recurre

a la siguiente orden vía línea de comandos: $analyze -f\ ./config/es.cfg --flush --server --port$ 50005 & Donde es.cfg es la configuración general de Freeling para lograr el correcto procesamiento del léxico en cuestión, de esto se tratará en la sección 'Configuraciones Freeling'2.2.1; 50005 es el puerto en el que el servidor escucha los pedidos. Una vez que se realizó este paso correctamente, podemos ejecutar nuestra aplicación Python.

2.2.1 Configuración

Los parámetros que se utilizaron para configurar Freeling en modo servidor fueron los siguientes(./config/es.cfg):

- AffixAnalysis=no : No es necesario en este caso.
- MultiwordsDetection=yes : Se utiliza para detectar palabras múltiples en una 'string' ya que es habitual errores de tipeo como la falta de espacios o la introducción del caracter . como separador.
- NumbersDetection=yes : Opcional, se optó por introducirlo aunque aporte en menor proporción.
- PunctuationDetection=yes : Detecta la puntuación que posteriormente se excluye de las palabras finales.
- DatesDetection=no : No es necesario en este contexto, puesto que en esta categoría de comentarios no requiere tal precisión para procesar.
- QuantitiesDetection=no : No es necesario procesar este tipo de magnitudes, se toma como una palabra cualquiera.
- DictionarySearch=yes : Para definir el lema, es útil que se encuentre en el diccionario para generar un tag correcto.
- ProbabilityAssignment=yes : En algún caso podría ser necesario.
- OrthographicCorrection=no : No sería una idea negativa utilizarlo, aunque para el caso no es una prioridad a tener en cuenta.
- NERecognition=no : Identificar nombres por ejemplo no aporta calidad a la solución, además se observa comentarios como "Genial, Me Ha Atrapado la película", en realidad genera problemas cuando estamos

agrupando las palabras positivas y negativas ya que con esta opción agruparía en un conjunto "Me Ha Atrapado" y se pedería lemas que aportan datos a la solución.

Estos fueron las configuraciones más importantes a tener en cuenta en el análisis morfológico, se deja a consideración el archivo ubicado en $./config/es.cfg$ para más detalles, además el mismo debe procesarse cuando se inicia el servicio de Freeling.

3 Ejecución

Para ejecutar la aplicación, únicamente se le debe brindar permisos de ejecución al archivo $tarea_1.py$, y mediante línea de comandos la siguiente orden: $./tarea_1.py$ o $./tarea_1.py - w$ para ejecutarlo utilizando webservices anteriormente descripto.

3.1 Parámetros

Se brinda una serie de parámetros adicionales para la ejecución de la tarea.

- -d: Muestra las gráficas en tiempo de ejecución, con la posibilidad de redimensionar las ventanas, entre otras propiedades interesantes a tener en cuenta.
- -v: Se utiliza para ejecutar la tarea en versión 1, es decir el programa se ejecuta utilizando la lista de palabras vacías de la primera entrega, no se genera filtro de lemas con el dominio del problema, los comentarios positivos corresponden a los valores 3,4 y 5.
- -w: Modo webservice, se ejecuta la aplicación realizando requests a un servicio web.

Una aclaración importante, la aplicación se ha probado exclusivamente en ambiente UNIX, desde la instalación de Freeling hasta las librerías de Python, ante funcionamientos anómalos en plataformas Windows, se sugiere que el programa sea ejecutado en UNIX. Caso contrario de no disponer un entorno, igualmente puede ejecutarse el programa utilizando webservices. Aunque el funcionamiento no es óptimo los resultados son semejantes.

4 Visualización

Se muestra la interacción del sistema, según el modo de ejecución.

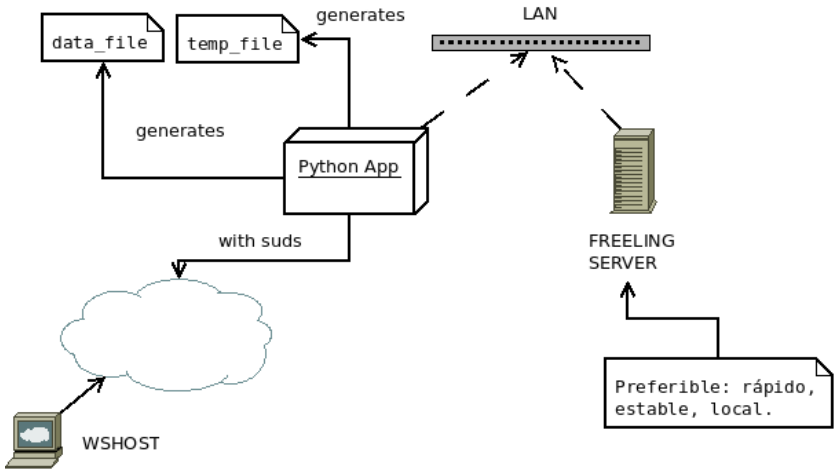

Figure 1: Interacción

5 Implementación

Se procede a la descripción de la implementación del programa. Para ello se utilizaron tres clases básicas para las configuraciones:

1. Utils : que se encuentra en el mismo archivo a ejecutar $tarea_1.py$, la misma se encarga de realizar operaciones de lectura y escritura en los archivos, así como el porcesamiento de los datos utilizando Freeling.

2. WebService : la misma se encuentra en el archivo webservices.py, se utiliza para cargar las especificaciones del servicio web en cuestión, además de brindar la implementación de solicitud y procesamiento de del léxico. Básicamente devuelve una lista con todas las palabras con su respectivo tag, lema y probabilidad.

3. Classification : Esta clase es la responsable de todo lo relacionado con el clasificador de comentarios, utiliza el paquete nltk para generar un clasificador NaiveBayes.

A continuación se describe las funciones definidas en el archivo $tarea_1.py$, específicamente de la clase utils.

5.1 tarea_1.py

5.1.1 loadXLSFile

Encabezado: def loadXLSFile(self).
Cuando se instancia la clase $Utils$, en sus atributos privados se encuentran ciertos parámetros que definen las condiciones a tener en cuenta para la lectura del archivo $.xlsx$, en particular el nombre del mismo $\_\_comment\_file$ y el rango de las celdas que deben ser procesadas($\_\_xls\_range$), básicamente utiliza la librería descrita en la sección anterior librería 2.1, y devuelve como resultado el rango de celdas especificado por el parámetro en cuestión, el cual será iterado en la función main.

5.1.2 loadStopwords

Firma: def loadStopwords(self, Ver=2).
En este caso se carga en un diccionario las palabras vacías que se suministran en el archivo $stopwords.txt$, se devuelve un diccionario para el rápido procesamiento posterior llevado a cabo. El parámetro Ver con el valor en dos implica que se carga la lista de stopwords mejorada según el obligatorio 2, en caso contrario se procede a la carga con la lista brindada en la entrega anterior.

5.1.3 loadSubjetiveElems

Firma: def loadSubjetiveElems(self).
Esta función carga el léxico proporcionado en el archivo $listaElementosSubjetivos.pl$, devolviendo un diccionario como estructura para su posterior procesamiento.

5.1.4 loadDomain

Firma: def loadDomain(self).
Básicamente abre el archivo $resources/dominio.txt$, y carga en un diccionario las palabras consideradas dentro del dominio del cine. En caso de que el archivo no se encuentre en el directorio especificado, se procede a lanzar una excepción correspondiente a dicho error.

5.1.5 lematization_freeling_client

Firma: *def* lematization_freeling_client(self, comment).
Argumentos:

- *comment* : Comentario particular a ser procesado.

Su funcionalidad principal es tomar un comentario específico mediante una string, preprocesado anteriormente; se emplea sockets para la comunicación con el servidor Freeling, a diferencia de la primera entrega que utilizaba únicamente el servidor Freeling local y el uso de archivos auxiliares para procesar los pedidos. La ventaja de la utilización de sockets es que se puede ejecutar en ambientes remotos, desacomplando totalmente la aplicación del servidor. Básicamente devuelve una lista con lemas que cumplen ciertos criterios, como ser:

- No se encuentra en la lista de palabras vacías.
- No se encuentra en la lista del dominio del problema.

5.1.6 generateData

Firma: *def* generateData(self, allWords, filename).
Básicamente esta función se encarga de guardar en archivos con extensión *.csv* las palabras procesadas durante la ejecución del programa. El parámetro allWords consiste en una lista de lemas que serán persistidos, mientras que $filename$ es el archivo destino. En la función principal del programa llamada *process* se generan por medio de esta función los siguientes archivos:

- negativeWords.csv : Contiene las palabras negativas encontradas en los comentarios.
- positiveWords.csv : Análogo al primero, aplicado a palabras positivas.
- allWords.csv : Se persisten todos los lemas involucrados en los comentarios.

5.1.7 generateStatisticData

Firma: *def* generateStatisticData(self, allWords, posWords, negWords, t, n).
Genera datos estadísticos para posterior análisis. Los parámetros $allWords$

consiste en una lista con todas las palabras, análogo para $posWords$ que son lemas positivos y $negWords$ lemas negativos, mientras que t es la cantidad total de comentarios, n la cantidad de comentarios negativos. En la función principal *process* se genera un archivo que contiene datos para formular conclusiones estadísticas, el mismo se encuentra en el directorio *res* nombrado *Data.txt* .

5.1.8 plot

Firma: def plot(self, listaPlot, n_groups, filename, totalwords).
Argumentos:

- listaPlot : Contiene una lista ordenada con la siguiente tupla de datos $(Palabra, Comentarios)$, donde $Palabra$ es el lema a graficar, y $Comentario$ es la cantidad de veces comentarios en la que $Palabra$ está presente. Notar que se contabiliza únicamente una vez la palabra por comentario, para evitar casos extremos que degraden el comportamiento del programa en cuestión, ej. Comentario = "Hola, hola, hola, ...", si se evalúa globalmente, puede perjudicar el estudio en cuestión.

- n_groups : Cantidad de palabras que serán procesadas en la salida gráfica.

- filename : Nombre del archivo de salida del procesamiento gráfico, la imagen será guardada en el directorio $figures$ localizado en la ruta donde se procesa la aplicación.

- totalwords : cantidad de lemas a ser procesados.

Su funcionalidad lo describe el nombre de la función, retorna una salida gráfica mediante archivos *.png* con los resultados obtenidos. En la implementación en cuestión se invoca cinco veces la presente función, generando los archivos a continuación:

- AllWords.png : Se encuentran las n_groups palabras ordenadas de forma decreciente, en este caso no se distingue entre palabras positivas y negativas, básicamente es un conteo global de todas las palabras procesadas.

- NegativeSubjetive.png : En este caso se suministra el top n_groups palabras presentes en comentarios con un a puntuación 1 o 2 intersección palabras negativas suministradas en el archivo $listaElementosSubjetivos.pl$.

- PositiveSubjetive : Análogo al caso anterior pero aplicado a palabras positivas.
- NegativeWords.png : Ilustra las palabras presentes en los comentarios con evaluación negativa(1 y 2).
- PositiveWords.png : Análogo caso anterior pero teniendo en cuenta palabras con evaluación positiva.

5.1.9 process

Este bloque se encarga de tomar las funciones anteriormente mencionadas y procesar el algoritmo de forma adecuada. Se presenta a continuación un pseudo-código de la implementación.

$xls \leftarrow cargar\_comentarios()$
$stopwords \leftarrow cargar\_palabras\_vacias$
$subjetivas \leftarrow cargar\_palabras\_subjetivas$
$domain \leftarrow cargar\_dominio$
for *comentario* in xls **do**
    $lista \leftarrow lematizar(comentario)$
    for *palabra* in $lista$ **do**
        $(pos, neg) \leftarrow agregarDiccionario()$
    end for
end for
$imagenes \leftarrow graficarResultados(pos, neg)$

Las primeras cuatro líneas se compactan en la función de instanciación del objeto Utils, cargando entonces las palabras vacías, el dominio del problema, la lista de palabras subjetivas y los comentarios. La simplicidad que brinda Python para el procesamiento de datos, básicamente resume el pseudocódigo en la implementación en sí misma, salvo detalles menores que requieren un procesamiento refinado.

5.2 Classification.py

Implementa la clase $Classificator$ encargada de clasificar los comentarios. A continuación se detallan las funciones que brinda dicha clase.

5.2.1 loadComments

Firma : def loadComments(self).
Carga los comentarios en memoria a partir del archivo $.xlsx$ brindado, para realizar un procesamiento dinámico posteriormente, además de utilizar heurísticas que involucran el cuerpo del comentario y no el post procesamiento utilizando tokenización y análisis morfosintáctico.

5.2.2 generate_train_set

Firma : def generate_train_set(self, large, prop, train_name, test_name).
Genera a través de la función $get\_train\_test\_set$ los índices de los comentarios que formarán parte del conjunto de comentarios de entrenamiento y test. A dichos índices los persiste en los archivos con nombre pasados por parámetro $train\_name$ y $test_name$ respectivamente. Los parámetros $large$ y $prop$ son la cantidad de comentarios en total y la proporción para el conjunto de entrenamiento(70 %) respectivamente.

5.2.3 get_train_test_set

Firma : def get_train_test_set(self, large, Prop).
Devuelve una lista con índices de los comentarios que serán incluídos en los conjuntos de entrenamiento y testing, se utilizan los parámetros $large$ y $Prop$ para determinar el tope de los comentarios y la proporción para el conjunto de entrenamiento. La lista que se genera se realiza de forma aleatoria, utilizando la clase $random$ que brinda python.

5.2.4 generateHeuristic

Firma: def generateHeuristic(self, com).
Devuelve $True$ o $False$ en caso que el comentario com posea un trigrama que forme parte de la heurística.

5.2.5 feature

Firma : def feature(self, N, pcomment, comment, positive, negative, posSub, negSub).
Devuelve un diccionario con los atributos pertinentes al comentario $pcomment$, para generar los atributos se utiliza una lista de lemas generado anteriormente

por la clase $Utils$ con la función $process$, consiste en la lista de lemas obtenida luego del análisis morfosintáctico. Para agregar atributos al comentario se tienen en cuenta tres tipos de atributos.

- positive(lema) negative(lema) : Se generan a partir de los parámetros $positive$ y $negative$, que son diccionarios con los lemas positivos y negativos preprocesados, el atributo consiste básicamente si el comentario posee un lema de alguna de esos diccionarios, caso contrario también se agrega el atributo indicando que no lo posee. Por ejemplo el comentario $Bueno.$, supongamos que el lema "bueno" está incluído en el diccionario $positive$, por lo tanto dicho comentario obtendrá el atributo $positive(bueno) = True$.

- subjetive_pos(lema) subjetive_negative(lema) : El comentario se adjudicará esta clase de atributos si el mismo posee un lema que se encuentra en el diccionario de elementos subjetivos, tanto positivo como negativo. Supongamos que en el análisis morfosintáctico un lema del comentario es "malo", y dicho lema se encuentra en uno de los diccionarios $posSub$ o $negSub$, entonces el comentario será marcado con el atributo correspondiente $subjetive\_pos(lema)$ o $subjetive\_neg(lema)$ correspondiente a los diccionarios anteriores.

- no_gusto : El comentario posee este atributo si se encuentra en $pcomentario$ alguna de las heurísticas planteadas, como por ejemplo "no me gusto", en caso de incluir una secuencia predeterminada que se define en $generateHeuristic$. Básicamente son trigramas que se estudiaron y se llegaron a la conclusión que aportan significancia estadística cuando se evalúan comentarios negativos.

5.2.6 load_train_test_set

Firma: def load_train_test_set(self, train="./resources/train.txt", test="./resources/test.txt").
Genera listas que contienen los índices de los comentarios incluídos en los conjuntos de entrenamiento y testing. Los parámetros $train$ y $test$ hacen referencia a la ruta de los archivos que contienen la información.

5.2.7 saveMetrics

Firma: def saveMetrics(self, toSave, file_name="./res/Metrics.txt").
Dadas las métricas de calidad, como ser $accuracy$, $Precision$ y $Recall$ además

de la matriz de confusión, son guardadas en el archivo $file\_name$. En la función principal *process* de la presente clase, se persiste los datos automáticamente en el directorio *res* con el nombre de archivo $Metrics.txt$.

5.2.8 process

Firma: def process(self, posWords, negWords, posI, negI, tokenizedComm, train_file="./resources/train.txt", test_file="./resources/test.txt").
Esta función se encarga básicamente de todo el procesamiento involucrando todas las funciones anteriormente mencionadas. Los pasos generados se detallan a continuación.

- Se cargan en memoria los comentarios del archivo $.xlsx$.
- Se cargan en listas los conjuntos de entrenamiento y testing previamente generados.
- Con todos los comentarios que pertenecen al conjunto de entrenamiento, se generan los atributos y se almacenan en una lista que posteriormente será utilizada para instanciar el clasificador.
- Creación del clasificador, utilizando la lista mencionada en el paso anterior.
- Dado el conjunto de testing se clasifican los comentarios utilizando el clasificador, y se almacenan los resultados en diccionarios para el posterior análisis de métricas de calidad.
- Se generan métricas de calidad, la matriz de confusión y se imprimen los 20 atributos más utilizados.

Los parámetros $posWords$ y $negWords$ son diccionarios que contienen los lemas positivos y negativos procesados anteriormente por la clase $Utils$, $posI$ y $negI$ son la intersección de los lemas positivos y negativos con los elementos subjetivos. Mientras que $tokenizedComm$ es un diccionario con la lista de lemas para cada comentario procesado, se utiliza para evitar repetir la tarea de análisis morfosintáctico y aumentar el tiempo de respuesta del programa. Finalmente $train\_file$ y $test\_file$ contienen las direcciones de los archivos que contienen los índices de los comentarios que pertenecen al conjunto de entrenamiento y testeo.

5.2.9 main

Genera los 30 casos de prueba que se tendrán en cuenta para el análisis estadístico generado por el módulo *Statistics.py*.

5.3 Statistics.py

5.3.1 doMetrics

Firma: def doMetrics(Display=False, WS=False, Ver=2).
Básicamente realiza el proceso de análisis morfosintáctico utilizando la clase $Utils$ y posteriormente clasifica los comentarios con los 30 conjuntos de entrenamiento y test, persisitiendo los datos obtenidos en el directorio $res/statistics$

5.3.2 generateFile

Firma: def generateFile(metrics, outfile).
Dadas las métricas de calidad para los 30 casos de prueba *metrics* se persisten las mismas en el archivo $outfile$, generando un archivo de salida *.csv* que contienen las métricas *accuracy*, *precision* y *recall*, además como los valores promedio para todos los casos.

5.3.3 main

Utiliza las dos funciones anteriormente mencionadas para generar las métricas de los 30 casos de prueba y persistirlos en los archivos $tabla1.csv$ y $tabla2.csv$.
Archivos generados:

- $tabla1.csv$: Contiene las métricas para los 30 casos de prueba procesados, se consideran los comentarios positivos con puntuación 4 y 5, y se utiliza el dominio y la lista mejorada de stopwords para un refinamiento de los lemas a procesar.

- $tabla2.csv$: Análogo al caso anterior pero los comentarios positivos poseen puntuación 3,4 y 5, no se utiliza dominio para filtrar lemas, ni la lista mejorada de stopwords. Básicamente se utiliza la versión 1 de la tarea para el procesamiento morfosintáctico.

5.4 Eficiencia

La complejidad del un programa se reduce a las estructuras utilizadas, la correcta utilización de las mismas implica un impacto directo sobre la eficiencia del programa. Para ello se utilizaron diccionarios para almacenar las palabras contenidas en los comentarios, de esta forma poder acceder en orden 1. La búsqueda se reduce a investigar si la llave(key) de indexado se encuentra presente en el diccionario. Por lo contrario, utilizando listas, la performance al procesar una lista mayor de comentarios tiene un impacto negativo en tiempos de ejecución. Se recomienda fuertemente utilizar el modo de ejecución con servidor local con Freeling, de esta forma se reducen costos de tráfico de red y posibles "errores de ejecución" ajenas al programa diseñado. La simplicidad de utilizar expresiones regulares para realizar reemplazos y búsquedas impacta positivamente en el desempeño del programa, es por ello que se utilizaron las mismas para sintetizar la lista de elementos subjetivos. Además agrega elegancia a la solución. La captura de excepciones permite brindarle al usuario un detalle sobre los requerimientos que debe satisfacer para ejecutar correctamente el programa, para ello se implementó un manejo de las mismas en los casos de lectura y escritura de archivos así como también en el caso del WebService, que en ocasiones adversas no responde de forma adecuada.

6 Análisis

En las secciones anteriores se ha detallado aspectos de inicialización del ambiente para ejecutar la aplicación, así como las configuraciones pertinentes de Freeling, detalles sobre la implementación y algunas consideraciones sobre eficiencia. En este apartado se analizarán los resultados obtenidos luego de la ejecución del programa. Para ello en las siguientes subsecciones se detallarán los puntos requeridos en la letra del laboratorio.

6.1 Información CORPUS

Para esta consigna se utilizó la librería *openpyxl* ya citada en la sección Librerías [2.1]. Básicamente se utilizó la función $loadXLSFile$ descrita en la sección Implementación [5.1.1], particularmente no se realizó un mapeo a memoria de los comentarios y sus respectivos puntajes, debido a que almacenar los mismos en estructuras internas como ser listas o diccionarios implica

contar con memoria extra para ejecutar la aplicación. En lugar de ello se recorre el archivo *.xls* brindado, haciendo lecturas del archivo en sí mismo, y no realizando el almacenamiento para su posterior lectura. Por lo tanto la información no se extrae completamente, pero sí parcialmente, con el fin de mejorar el rendimiento tanto en consumo de memoria como en la velocidad de ejecución del programa. La lectura se realiza por medio de la función $loadXLSFile$ y posteriormente se recorre en la función *main*, 'lematizando' y realizando un análisis morfosintáctico, pero ello hace parte de la siguiente subsección.

6.2 Procesamiento Información

Como se mencionó anteriormente, las tareas de extracción de la información del CORPUS y el procesamiento de la información mediante Freeling se realizaron en paralelo con la finalidad de optimizar recursos y potenciar la eficiencia del programa. Para este punto se puede utilizar procesamiento mediante WebServices o un servidor Freeling local(recomendado), las configuraciones del servidor local se encuentran detalladas exhaustivamente en el archivo $es.cfg$ que se encuentra en el directorio $/config$. Los detalles particulares sobre la configuración de Freeling ya fueron mencionados[2.2.1]. En la función *main* se utiliza la función $lematizacion\_freeling\_client$ para procesar el comentario perteneciente al CORPUS, la cual tiene como retorno una lista de las palabras únicamente con el lema, puesto que el tag se utiliza para filtrar algunos elementos que no serán necesarios para el post procesamiento, en este caso se filtraron los siguientes tags:

- F : Denotan signos de puntuación, los cuales no consideraremos en nuestro análisis posterior.
- Z : Los determinantes numerales tampoco son considerados en el análisis.

Una vez que se cuenta con la lista de lemas de las palabras de un comentario, se almacenan en un diccionario usando como llave dicho lema, y como valor un dato del tipo Integer, con la finalidad de contabilizar la cantidad de veces que ocurre dicho lema en todos los comentarios a estudiar. Dicho conteo se realiza una vez por comentario, no la cantidad de ocurrencias global, con el fin de evitar comentarios con repetición de palabras que puedan interferir en el análisis de la muestra. Se agregan los lemas en tres diccionarios, uno

contiene todos los lemas procesados hasta el momento, y realiza el conteo global de todos los lemas independientemente del puntaje de un comentario. El segundo diccionario contiene únicamente los lemas que ocurren en comentarios evaluados con notas 1 y 2, mientras que el tercer diccionario almacena los lemas que ocurren en comentarios cuyo puntaje es 3,4 o 5.

6.3 Incorporar Recursos

Este punto se realiza ejecutando las funciones descritas en la sección implementación[5], ellas son *loadSubjetiveElems*, la cual se encarga de incorporar el léxico suministrado en el archivo *listasElementosSubjetivos.pl* y la función *loadStopwords* que incorpora el archivo *stopwords.txt* que contiene una lista con las palabras vacías, a modo de poder filtrar una cantidad mayor de palabras. Ambos archivos se encuentran en el directorio */resources* junto con el .xls a procesar. Las funciones retornan un diccionario con las palabras que se encuentran en los respectivos archivos, en el caso de los elementos subjetivos retorna dos diccionarios, uno con las palabras positivas y el otro las negativas, a modo de trabajar posteriormente con conjuntos y realizar las intersecciones pertinentes.

Se consideró oportuno mantener la información del obligatorio anterior, como son las secciones anteriores, ya que la tarea es incremental.

6.4 Parte 1 :

Para este requerimiento fue sencillo mejorar la lista de palabras vacías(stopwords), básicamente se generalizó la función 5.1.2 agregando un parámetro que indique si se debe cargar una lista en particular o la intersección de varias listas encontradas en la web. En cuanto al requerimiento del dominio del problema, se generó una lista que se encuentra en un archivo llamado *dominio.txt*, ubicado en el directorio */resources*. Para cargar esta lista de palabras, nuevamente se utilizó un diccionario para mejorar el proceso de búsqueda. Para más referencias observar la descripción de la función 5.1.4. En el requerimiento que considera los comentarios positivos únicamente con la puntuación 4 y 5, se agregó el parámetro Ver, si el mismo posee el valor 2, se excluyen los comentarios con puntuaciones 3, caso contrario se mantiene como en el obligatorio 1. Este argumento se agrega al instanciar la clase $Utils$, y depende del pasaje de parámetros cuando se ejecuta el programa, en caso de existir la opción $-v$, se considera la versión de la tarea 1, y por tanto los comentarios

positivos contienen los valores 3,4 y 5. También vale la pena destacar que la presencia de este parámetro también condiciona los filtros sobre los lemas, utilizando listas de stopwords mejorada, así como el dominio del problema.

6.5 Parte 2 :

Los atributos seleccionados fueron los siguientes:

- *positive*(*string*): Donde *string* pertenece a los N primeros lemas de la lista cuyos comentarios tuvieron una mención positiva, indicando si el lema está está presente o no en dicho comentario. Para ello se utiliza un atributo binario(True, False).

- *negative*(*string*): Análogo al anterior, aplicado a los N elementos de la lista con lemas donde los comentarios fueron evaluados con puntuación negativa.

- *subjetive_pos*(*lema*) : *lema* pertenece a la lista de elementos subjetivos brindada, si en el comentario dicho lema pertenece, entonces se agrega el atributo mencionado.

- *subjetive_neg*(*lema*) : Análogo al anterior aplicado a la lista de elementos subjetivos negativos.

- *no_gusto* : Una heurística detectada es que en una gran proporción de comentarios negativos a procesar, poseen términos afectivos positivos, como ser 'gustó', sin embargo están contenidos en un contexto semejante a 'no me gustó' o semejante. Para ello se procedió a evaluar 2 palabras anteriores a los términos afectivos positivos, y en caso de encontrarse con una negación, se agrega el atributo mencionado. Nuevamente un atributo binario, demostrando la presencia o no de dicha heurística en el comentario.

Los atributos se analizaron con una selección aleatoria de comentarios, con la proporción requerida por la letra del obligatorio. Se realizaron pruebas con conjuntos distintos y se realizó un análisis estadístico de los datos, principalmente si el comportamiento del conjunto es normal, para ello se utilizaron test de normalidad de DAgostino y ejecutaron sobre 30 conjuntos distintos, ya que con dicho valor agrega un significado estadístico relevante. Si bien la letra del problema no requiere este procedimiento, se consideró oportuno observar

si el conjunto seleccionado tiene impacto positivo o negativo en el momento de evaluar la calidad del clasificador. Para eso consideraremos la desviación estándar en los datos, así como la media de los valores *precision* y *recall*, así como también *accuracy*. En el caso que los conjuntos seleccionados de forma aleatoria se comportan siguiendo una distribución normal, podemos establecer una cota en los atributos de calidad a evaluar, de esta forma también podemos asegurar que en promedio la calidad del clasificador es semejante, cualquiera sea el conjunto de datos tomado como referencia, con un margen de error acotado. Además para evaluar la calidad de las soluciones, no es suficiente tratar con un conjunto particular, puede ocurrir que al seleccionar un conjunto cualquiera, con la solución propuesta los atributos de calidad sean óptimos, pero no para un caso global. Entonces evaluar la solución sobre una cantidad significativa de conjuntos si brinda datos significativos sobre el comportamiento del clasificador de forma global y no particular.

6.6 Parte 3 :

Para esta sección utilizamos el conjunto de librerías *nltk* para generar un clasificador. Para ello se creó una clase llamada $Classification.py$ que cuenta con las funciones mencionadas en el apartado de la arquitectura del sistema. Básicamente las funciones principales a tratar en esta clase son las siguientes:

- $generate\_train\_set$
- $feature$

La primera se utiliza para generar los conjuntos de entrenamiento y testing, utilizando una proporción 0.7 para el conjunto de entrenamiento. La segunda función agrega los atributos a los comentarios en cuestión, descriptos en la sección anterior[6.5]. Como se mencionó anteriormente se evaluó el comportamiento a partir de 30 conjuntos de entrenamientos distintos, tanto para los comentarios positivos con puntuación 3,4 y 5 así como los que excluyen el puntaje 3 del conjunto. Los resultados obtenidos en esta parte se presentan en la sección Estadístico[6.7], con dichos valores se procederá a tomar conclusiones adecuadas a la calidad del clasificador generado, dependiendo también de qué se considera como comentario positivo. La generación del conjunto de entrenamiento se realizó de forma aleatoria, para evitar linealidad en la selección de los comentarios, influyendo de forma negativa en el momento de

evaluar la calidad del clasificador. Para ello se toma la cantidad de comentarios en el Corpus y se toma aleatoriamente 0.7 de los mismos, los restantes formarán parte del conjunto de testing. Cabe mencionar que por facilidad en la ejecución del programa, se guarda en un archivo *test.txt* y *train.txt* los números relativos a los comentarios seleccionados.

6.7 Estadístico

Para la obtención de los resultados, se implementó el módulo *Statistics.py*, el mismo utiliza los módulos *tarea_1.py* y $classification.py$, toma los 30 conjuntos de entrenamiento anteriormente generados y clasifica los comentarios según los mismos. Además posee la funcionalidad de generar las tablas presentes en esta sección. Se obtuvieron los siguientes resultados[2] generando 30 conjuntos aleatorios de entrenamiento y test, los resultados obtenidos implican que el *accuracy* del clasificador es de aproximadamente 80%, el mejor caso se obtuvo en el conjunto de número 23, donde el *accuracy* se aproximó a un 85 %. Este estudio demuestra en un caso global, ya que con una cantidad apreciable de conjuntos distintos podemos obtener significancia estadística en el momento de evaluar el clasificador en forma genérica. En la tabla 3 tenemos los resultados incluyendo las puntuaciones 3 como comentarios positivos. Intuitivamente se podría decir que eliminando los comentarios con puntuación 3 tanto para entrenar el clasificador, como para evaluar los comentarios con ese puntaje, se obtendrían mejores resultados, pero en cambio las tablas demuestran esa hipótesis. Analizando la segunda tabla, podemos decir que los comentarios con puntuación 3 tiene un impacto impacto nulo cuando se clasifican los comentarios positivos, sin embargo en el momento de evaluar los negativos el desempeño es inferior al clasificador entrenado con un conjunto de comentarios excluyendo puntuaciones 3. Esto es un dato interesante para considerar, si tomamos el caso número 10 tenemos un *accuracy* de 84.7% apenas inferior al de la primera tabla. Cuando contrastamos con las métricas *Recall* y *Precision* concluimos que para el caso 1, existe una leve mejora respecto al caso 2. Algunos detalles del estudio estadístico como ser la desviación estándar, y el test de normalidad de D Agostino se brindan en la tabla 1. Para éste último como utilizamos un tamaño de muestra de 30 conjuntos, evaluaremos con un $\alpha = 0.02$ y a partir de esto tenemos las siguientes cotas $[0.2622, 0.2871]$, en el caso que el resultado del test de normalidad sea un valor que pertenece a dicho conjunto, la muestra es normal, por lo cual podemos utilizar como referencia para el planteo de conclusiones

en cuanto a la calidad del clasificador se refiere. Los casos propuestos se resumen:

- Caso 1: Evaluación excluyendo comentarios con puntuación igual a 3.
- Caso 2: Evaluación incluye comentarios con puntuación igual a 3, dentro de la categoría positivos.

Se tomó como referencia todos los atributos de calidad planteados, la nomenclatura se detalla en el siguiente numerado:

- ACC: Accuracy.
- PP: Precision Positive, métrica Precision aplicada a los comentarios positivos.
- RP: Recall Positive, métrica Recall aplicada a los comentarios positivos.
- PN: Precision Negative, métrica Precision aplicada a los comentarios negativos.
- RN: Recall Negative, métrica Recall aplicada a los comentarios negativos.

Las conclusiones que se pueden observar a partir de la tabla es que se puede tomar cualquier métrica como referencia, puesto que al realizar el test de normalidad efectivamente estamos comprobamos que para los datos, se comportan siguiendo una distribución normal. A partir de estos datos podemos comparar los dos casos propuestos de forma independiente según la métrica deseada, tomando la media como referencia y estableciendo qué atributo es mejor para los dos casos planteados. Podemos decir entonces que para el Caso 1, se obtiene una mejor evaluación de los comentarios negativos, mientras que para los positivos en ambos casos las métricas son semejantes. Por tanto a nivel global con la métrica *accuracy* el clasificador con distintos dominios de comentarios se comporta de forma similar, teniendo ventaja el primer clasificador excluyendo los comentarios con puntuaciones 3 de la clasificación positiva.

	Caso 1		Caso 2	
Métrica	σ	DA	σ	DA
ACC	0.029982	0.27883	0.027009	0.27709
PP	0.029982	0.27346	0.022482	0.26285
RP	0.031108	0.28232	0.030635	0.27720
PN	0.070022	0.28525	0.068971	0.27332
RN	0.049914	0.27910	0.065903	0.27842

Table 1: Datos Estadísticos

6.8 Parte 4

Para esta subsección posteriormente a un análisis estadístico que involucra varios conjuntos de estudio, se selecciona un conjunto independiente a los del caso de prueba, si bien las métricas obtenidas están acotadas por lo cual se centrará particularmente en la generación de la matriz de confusión, así como las métricas *Precision* y *Recall*.

Se establece la separación de conjuntos en comentarios negativos y positivos, evaluado *Recall* y *Precision*.

- Accuracy: 0.8078
- Precision Positive: 0.8514
- Recall Positive: 0.8922
- Precision Negative: 0.6666
- Recall Negative: 0.5806

La matriz de confusión pertinente queda representada en la tabla 4, los valores en la diagonal indican los casos Verdadero-Positivo, mientras que cualquier otra celda que no corresponde a la diagonal indica que la clasificación del comentario es errónea. Como podemos observar, la correcta clasificación de los comentarios existe en mayor proporción cuando los comentarios poseen una valoración positiva. La proporción de comentarios negativos correctamente clasificados apenas supera el 50%, lo cual implica que el rendimiento del clasificador es estrechamente superior en el momento

Caso	Accuracy	PP	RP	PN	RN
1	0.773	0.840	0.850	0.586	0.567
2	0.834	0.860	0.919	0.745	0.613
3	0.755	0.776	0.900	0.674	0.443
4	0.774	0.821	0.868	0.638	0.552
5	0.791	0.836	0.879	0.655	0.571
6	0.790	0.837	0.889	0.617	0.509
7	0.805	0.879	0.864	0.586	0.618
8	0.825	0.854	0.913	0.731	0.603
9	0.836	0.888	0.898	0.660	0.636
10	0.831	0.834	0.950	0.818	0.545
11	0.817	0.839	0.916	0.740	0.578
12	0.833	0.880	0.896	0.685	0.649
13	0.762	0.811	0.858	0.627	0.544
14	0.817	0.836	0.939	0.714	0.455
15	0.828	0.867	0.908	0.692	0.600
16	0.827	0.864	0.917	0.674	0.544
17	0.790	0.840	0.878	0.627	0.552
18	0.846	0.866	0.942	0.762	0.561
19	0.830	0.866	0.912	0.700	0.593
20	0.835	0.861	0.916	0.755	0.635
21	0.865	0.880	0.951	0.800	0.604
22	0.789	0.855	0.855	0.619	0.619
23	0.853	0.882	0.935	0.725	0.580
24	0.823	0.849	0.913	0.741	0.606
25	0.812	0.868	0.893	0.596	0.538
26	0.809	0.822	0.931	0.756	0.515
27	0.826	0.866	0.903	0.692	0.610
28	0.806	0.855	0.900	0.605	0.500
29	0.739	0.800	0.837	0.569	0.508
30	0.762	0.845	0.835	0.525	0.544
AVG	0.809	0.849	0.899	0.677	0.566

Table 2: Métricas conjuntos

de evaluar comentarios positivos. *Precision* es una métrica que se evalúa

Caso	Accuracy	PP	RP	PN	RN
1	0.771	0.874	0.822	0.500	0.600
2	0.794	0.872	0.855	0.561	0.597
3	0.763	0.804	0.896	0.583	0.400
4	0.775	0.821	0.892	0.580	0.433
5	0.771	0.849	0.849	0.524	0.524
6	0.794	0.854	0.888	0.531	0.456
7	0.817	0.877	0.894	0.569	0.527
8	0.840	0.883	0.910	0.684	0.619
9	0.813	0.873	0.894	0.560	0.509
10	0.847	0.864	0.944	0.771	0.561
11	0.809	0.866	0.884	0.617	0.578
12	0.813	0.871	0.893	0.577	0.526
13	0.790	0.829	0.902	0.627	0.471
14	0.813	0.869	0.899	0.563	0.491
15	0.771	0.859	0.842	0.500	0.533
16	0.840	0.869	0.937	0.683	0.491
17	0.767	0.867	0.828	0.478	0.552
18	0.878	0.922	0.922	0.719	0.719
19	0.824	0.879	0.897	0.618	0.576
20	0.802	0.859	0.884	0.596	0.540
21	0.805	0.895	0.856	0.516	0.604
22	0.794	0.861	0.869	0.574	0.556
23	0.828	0.896	0.892	0.549	0.560
24	0.794	0.866	0.857	0.588	0.606
25	0.802	0.876	0.876	0.500	0.500
26	0.802	0.833	0.918	0.652	0.455
27	0.824	0.894	0.877	0.603	0.644
28	0.805	0.870	0.890	0.511	0.462
29	0.752	0.851	0.812	0.500	0.569
30	0.782	0.863	0.859	0.500	0.509
AVG	0.803	0.866	0.881	0.578	0.539

Table 3: Métricas conjuntos

según la siguiente ecuación:

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

Entonces esta métrica nos reporta el porcentaje de acierto en la clasificación, observar que $Precision$ se maximiza cuando $FP = 0$, es decir que no existen Falsos-Positivos es decir, el clasificador evaluó correctamente todos los comentarios. En cambio tenemos la ecuación de $Recall$:

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

Análogo al anterior pero define la métrica en proporción a los Falsos-Negativos obtenidos por el clasificador. Con los datos obtenidos, podemos decir que el clasificador no posee un comportamiento óptimo, si evaluamos el clasificador en un CORPUS distinto, pero aplicado al mismo dominio del cine donde predominen comentarios negativos, posiblemente el clasificador sea correctamente entrenado, mejorando los resultados en gran proporción. Vale la pena destacar que existe anomalía en ciertos comentarios, como no corresponder la puntuación con el cuerpo del comentario, un ejemplo claro es la contradicción de cierto usuario cuyo comentarios es "Muy bueno!", y el puntaje asignado corresponde a un 2. Estas anomalías no están al alcance del clasificador si no se tiene un dominio suficientemente extenso como para llegar a la conclusión que el lema "bueno", condiciona que el comentario sea negativo. En otras palabras, es más probable que dicho lema corresponda a un comentario cuya puntuación sea positiva. Sin embargo, existen comentarios de la forma "No me gusta", en los cuales si bien contiene un lema que pondera a comentarios positivos, aplicado al contexto con una negación implica lo contrario y por ende esos casos se consideraron como heurística factible para mejorar las métricas del clasificador en el momento de evaluar el conjunto test.

	POS	NEG
POS	149	18
NEG	26	36

Table 4: Matriz Confusión

6.9 Parte 5

Se seleccionaron los primeros 20 atributos en la tabla 5 como podemos apreciar, los atributos corresponden en su gran mayoría a los que recurren con

mayor frecuencia, los lemas graficados en la Parte 1 del obligatorio. Si realizamos un análisis según el tipo de atributo implementado, tenemos la siguiente información:

6.9.1 Atributos Positive-Negative

Los atributos que se utilizan con mayor frecuencia para este caso:

- negative(malo)
- positive(excelente)
- positive(recomendable)
- negative(aburrir)
- positive(divertir)
- positive(recomer)
- positive(encantar)
- positive(lindo)
- negative(terminar)

6.9.2 Atributos Subjetive

- subjetive_neg(malo) = 1
- subjetive_neg(malo) = 2
- subjetive_neg(aburrir) = 1
- subjetive_neg(absurdo) = 1
- subjetive_neg(lento) = 1
- subjetive_neg(falta) = 1
- subjetive_neg(desear) = 1
- subjetive_pos(vida) = 2

- subjetive_pos(familia) = 2
- subjetive_pos(niño) = 1

Atributo	POS-NEG	%
subjetive_neg(malo) = 1	neg : pos	26.4 : 1.0
negative(malo) = True	neg : pos	24.0 : 1.0
positive(excelente) = True	pos : neg	22.5 : 1.0
no_gusto = True	neg : pos	20.6 : 1.0
positive(recomendable) = True	pos : neg	13.7 : 1.0
subjetive_neg(malo) = 2	neg : pos	10.8 : 1.0
negative(aburrir) = True	neg : pos	8.8 : 1.0
subjetive_neg(aburrir) = 1	neg : pos	8.8 : 1.0
subjetive_neg(absurdo) = 1	neg : pos	8.8 : 1.0
positive(divertir) = True	pos : neg	7.1 : 1.0
positive(recomer) = True	pos : neg	6.4 : 1.0
positive(encantar) = True	pos : neg	5.4 : 1.0
subjetive_neg(lento) = 1	neg : pos	5.3 : 1.0
subjetive_neg(falta) = 1	neg : pos	5.3 : 1.0
subjetive_neg(desear) = 1	neg : pos	4.9 : 1.0
subjetive_pos(vida) = 2	neg : pos	4.9 : 1.0
positive(lindo) = True	pos : neg	3.3 : 1.0
negative(terminar) = True	neg : pos	2.9 : 1.0
subjetive_pos(familia) = 2	neg : pos	2.9 : 1.0
subjetive_pos(niño) = 1	neg : pos	2.9 : 1.0

Table 5: Atributos

Los términos más utilizados por los comentaristas en comentarios negativos sin lugar a dudas son "malo","aburrir" y "absurdo", a tal punto que "malo" aplica en dos atributos, los subjetivos y positive-negative, además en los primeros aparecen con gran frecuencia una y dos veces en los comentarios, sin embargo no tienen relevancia los lemas "gustar", "quedar" y "año", ubicados en 2,3 y 4 lugar en la lista de palabras negativas con mayor frecuencia. Esto tal vez implica que dichos lemas no tengan un aporte significativo cuando se trata de comentarios negativos, el primero(gustar) probablemente aparece que frecuencia en comentarios cuyo antecesor sea una negación y por

ende pondera en comentarios negativos. Si se excluye el lema "año", puede mejorar incluso el clasificador. Para los comentarios positivos tenemos "excelente", "recomendable" y "divertir", aparecen en las posiciones 1, 3 y 6 de los lemas con mayor frecuencia en el análisis morfológico, sin embargo los lemas de la posición 2 y 5 no pueden ser descartados, ya que en el presente conjunto no aportan significancia estadística pero en cualquier otro conjunto el aporte puede ser sustancial. Los datos en la tabla son coherentes con los lemas más utilizados tanto para comentarios positivos como negativos. La función $show\_most\_informative\_features$ brinda la información necesaria a considerar en el momento de evaluar comentarios, a partir de estos atributos podemos llegar a la conclusión que el aporte del atributo $subjetive\_neg(malo)$ es de aproximadamente un cuarto, esto implica que si un comentario contiene este lema, es muy probable que al clasificarlo se lo valore como negativo, lo cual es coherente con el tipo de comentarios en general, excepto si se trata de comentarios que actúan como respuesta a usuarios, por ejemplo "@usuario, no me pareció una película mala.", en este caso el comentario puede poseer una valoración positiva, pero irremediablemente el clasificador por ponderar el lema malo seguramente clasifique dicho comentario como negativo. Las métricas de calidad indican que el clasificador tiene problemas evaluando comentarios negativos. Existen diversos factores para esto que se enumerarán a continuación.

- En su totalidad tenemos 192 comentarios negativos, de los cuales evaluamos 62. Entonces estamos ante un problema de que no contamos con los comentarios suficientes para entrenar un clasificador de forma adecuada. Si observamos proporciones estamos evaluando un 32 % de los comentarios, es decir que el método de selección aleatorio es prácticamente uniforme(utilizamos 70% de los comentarios para el conjunto de entrenamiento). Descartando el método de selección como fuente de problema, concluimos que efectivamente el problema a considerar sería la cantidad de comentarios negativos dado el CORPUS no es suficiente para obtener buenas métricas para el clasificador.

- Lemas por sí solos no aportan lo suficiente, lo que no permite una evaluación precisa del clasificador. Fundamentalmente la coherencia entre el comentario y la puntuación depende de la relación entre los lemas presentes, involucra un contexto en muchas ocasiones poco predecible. Esto se debe a que existen atributos los cuales consideramos positivos

que forman parte de un comentario negativo y viceversa. Esto implica que la evaluación depende del contexto y la relación de los lemas presentes, factores externos como ser frases con alto contenido irónico puede inducir al clasificador a evaluar de formar errónea un comentario. Lo último es complejo, implica el uso de puntuaciones que pueden cambiar totalmente el significado de un comentario, y esto no se evalúan únicamente utilizando lemas por separado.

Ambos puntos, tanto la carencia de comentarios negativos para entrenar correctamente al clasificador, como un contexto incierto conllevan a evaluar correctamente 56 % de los comentarios negativos, una cifra baja aunque en contextos similares con CORPUS distintos esta métrica suele ser peor. Sin embargo cuando tratamos de evaluar comentarios positivos, la métrica es claramente favorable, esto es *Recall* aproximado a 90%, sin lugar a dudas para este tipo de comentarios tiene una efectividad relativamente alta. Claro está un *Precision* menor, debido a que la cantidad de comentarios negativos evaluados como positivos es superior a la cantidad de comentarios positivos evaluados incorrectamente. En cambio *Precision* para comentarios negativo es superior a *Recall*, puesto que el clasificador evalúa incorrectamente comentarios positivos con un margen muy pequeño(10%).

6.10 Resultados

Finalmente contamos con los resultados requeridos, ya sea ejecutando la aplicación que brinda una ventana con los gráficos o las imágenes guardadas en el directorio figures una vez terminada la ejecución. Primeramente analizaremos los resultados globales, para ello observamos gráfica de la figura [4], en la cual las palabras más utilizadas son 'bueno', 'excelente', 'gustar', 'recomendable', 'encantar'. Posteriormente analizaremos a qué conjunto pertenecen dichas palabras, es decir, si se tratan de palabras positivas o negativas y si se incluyen en el léxico suministrado *listasElementosSubjetivos.pl*. Los primeros cuatro elementos suman aproximadamente un 30% de las palabras sintetizadas, puede interpretarse como un resultado específico del contexto, al tratarse de comentarios sobre películas el uso de las mismas es excesivo, ya sea por la calidad al escribir los comentarios por parte de los usuarios o porque el contexto en sí mismo amerita este tipo de palabras. Quiere decir estadísticamente que un tercio de las palabras encontradas en comentarios sobre películas se resume a las primeras cuatro. Para el siguiente caso se

analiza las palabras presentes con mayor frecuencia en los comentarios evaluados con puntaje negativo, ellas son 'bueno', 'malo', 'gustar', 'quedar', 'año', paradójicamente el lema bueno es superior al lema malo, por lo que puede interpretarse que los usuarios comentan utilizando la negación, como ser un ejemplo "no es buena la película", en este caso involucra un comentario negativo pero utilizando un lema valorado positivo. Para más detalles en el momento de realizar el análisis se obtuvo una salida extra para procesar datos con mayor precisión e incluir en el presente informe un análisis con 'peso' estadístico. El mismo se puede encontrar en el directorio *res/*.

Lo anterior corresponde al análisis morfosintáctico de los comentarios. En cuanto a los resultados del clasificador implementado la mayoría de las conclusiones se han extraído de la parte 5[6.9] en cuanto a los atributos de calidad se refieren. Cabe mencionar que la implementación del clasificador se realiza de forma sencilla mediante la utilización del conjunto de librerías *nltk*, lo cual permitió incluso realizar un análisis estadístico de los datos de forma parcial. Además de los atributos de calidad requeridos por la tarea, se pudieron acotar dichos valores realizando a través del estudio estadístico, se utilizaron como referencia la desviación estándar(σ) como parámetro para acotar a través del promedio(X), formando un intervalo de confianza $[X - \sigma, X + \sigma]$ y de esta forma estimar correctamente los parámetros de calidad utilizando una cantidad suficiente de conjuntos para llegar a estos resultados. Si bien no se llegaron a resultados esperados, puesto que se pretendía llegar a un clasificador con un *accuracy* promedio de 90 %, se pudo lograr un clasificador promedio. Esto demuestra la complejidad del problema, sin dudas es una temática que requeriría años de estudio, primeramente para analizar los conjuntos con otras herramientas que nos permitan seleccionar atributos que aporten de forma más eficiente a la solución del problema. La segunda problemática es lograr comprender adecuadamente la relación existente el contexto y los lemas utilizados, existe un dominio extenso, dependientes del contexto y por lo tanto no es suficiente utilizar únicamente atributos representativos a través de los lemas.

6.10.1 Lemas

Se grafica la cantidad de lemas obtenidos luego del procesamiento de datos[2].

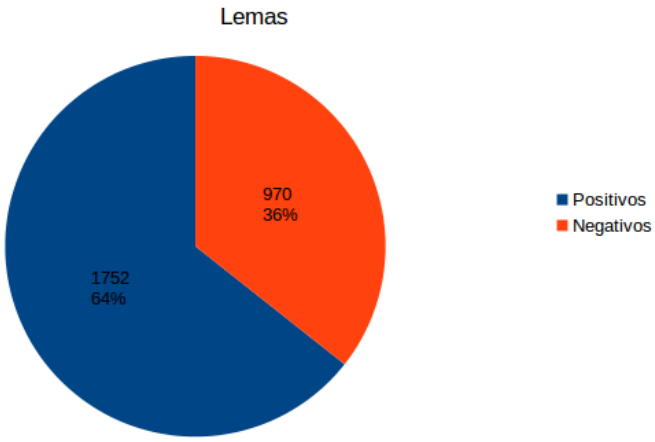

Figure 2: Cantidad de lemas

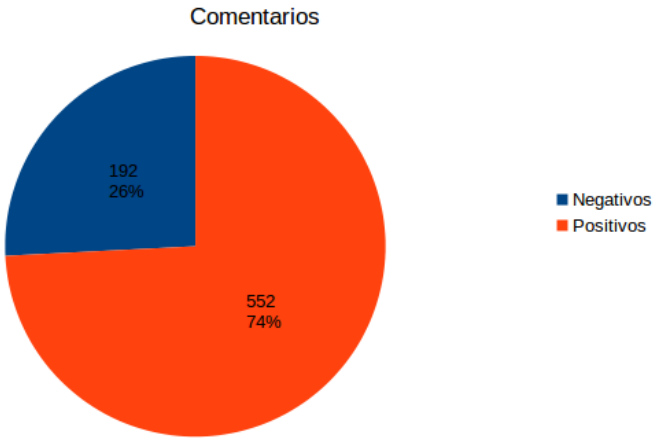

Figure 3: Cantidad de comentarios

6.10.2 Comentarios

En este caso, se contabilizan la cantidad de comentarios positivos y negativos[3].

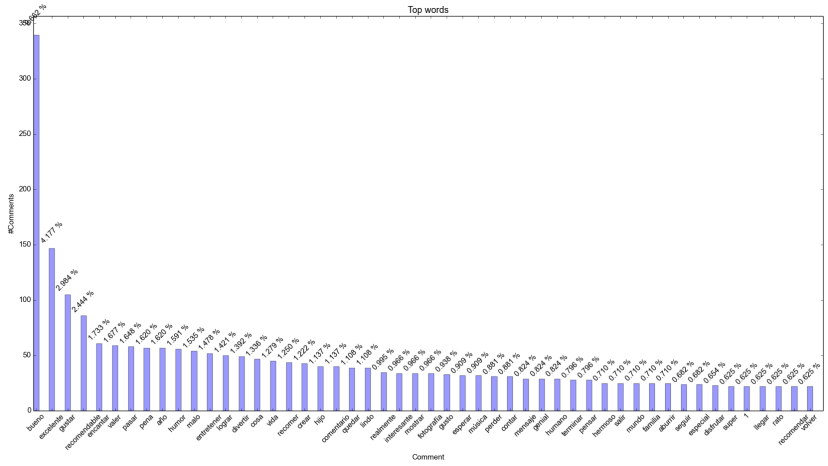

Figure 4: Todas las palabras

6.10.3 Gráficas

La gráfica[4] muestra las palabras más encontradas en los comentarios punteados de manera positiva como negativa. Se aprecia en la misma una mayor cantidad de palabras de semántica positiva con respecto a las de semántica negativa.

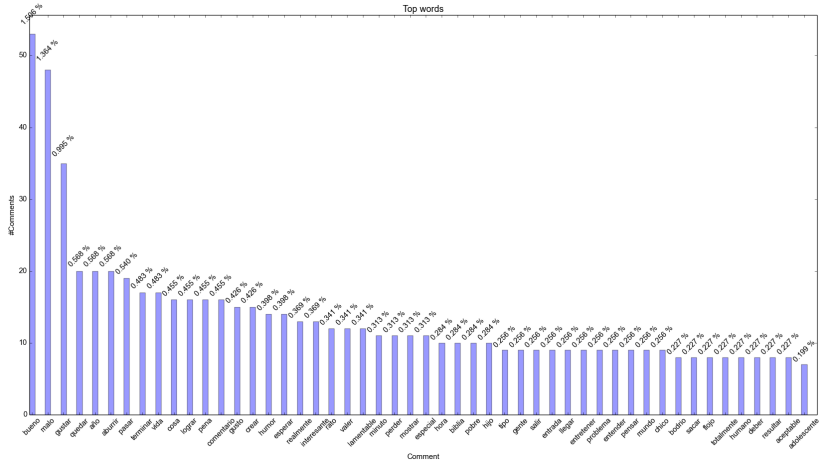

Figure 5: Todas las palabras negativas

La gráfica[5] muestra las palabras más encontradas en los comentarios

de puntuación negativa. El bajo porcentaje obtenido de frecuencia de las palabras, se cree que es debido al ruido que poseen los datos, que lleva a que se obtengan resultados no del todo fiables.

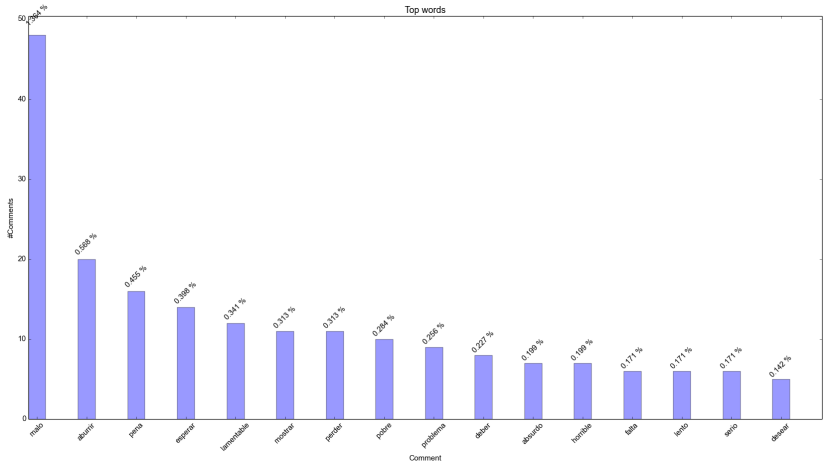

Figure 6: Intersección palabras negativas

La gráfica[6] también se observa los bajos valores de los porcentajes de las palabras encontradas, se cree que esto es debido al ruido que poseen los datos como se explico en la gráfica anterior. En la gráfica se observa una diferencia notoria del termino “malo” con respecto a los otros términos en su frecuencia.

La gráfica[7] muestra las palabras más encontradas en los comentarios de puntuación positiva. El bajo porcentaje obtenido de frecuencia de las palabras, se cree que es debido al ruido que poseen los datos, que lleva a que se obtengan resultados no del todo fiables. Se observa la presencia de las palabras “bueno”, “excelente”", “gustar”,”recomendable”,”encantar” entre las primeras con mayor frecuencia, como también sucede en el caso de la gráfica de todas las palabras y la gráfica correspondiente al Top de las palabras negativas. Se observa una mayor frecuencia de los términos “bueno” y “excelente” con respecto a los otros.

La gráfica[8] también se observa los bajos valores de los porcentajes de las palabras encontradas, se cree que esto es debido al ruido que poseen los datos como se explico anteriormente. En la gráfica se observa una diferencia notoria de los términos “bueno” y “humor” con respecto a los otros términos en su frecuencia. Estos terminos también son los términos de mayor frecuencia en

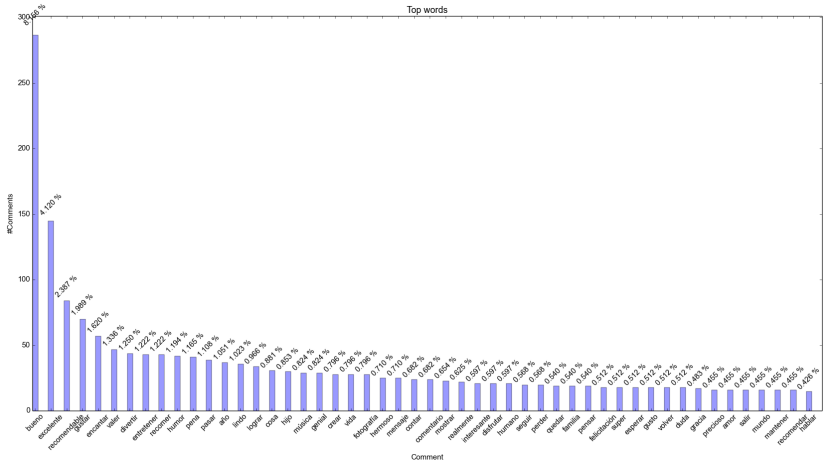

Figure 7: Todas las palabras positivas

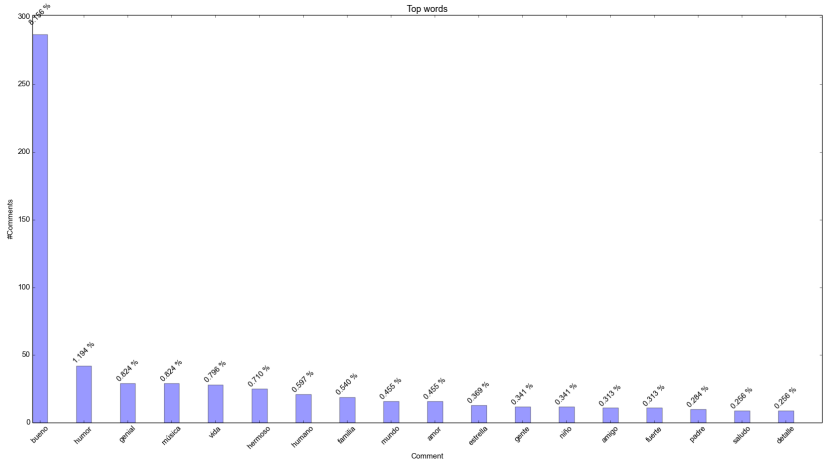

Figure 8: Intersección palabras positivas

la gráfica Top palabras positivas.

7 Conclusiones

Se detallan algunas puntos a tener en cuenta luego de finalizada la presente tarea, se realizan algunos comentarios sobre la implementación. Posteriormente detalles sobre los resultados obtenidos relacionados con la implementación.

7.1 Implementación

Lenguaje de programación alto nivel, presenta gran diversidad de librerías internas como externas que facilitan la tarea en gran parte. Su alto nivel permite no definir tipos específicos y de esta forma se trabaja de manera más fluida, aunque esto pueda llegar a confusiones si no se declaran correctamente las variables con un nombre adecuado. Escaso tiempo de implementación, dada la versatilidad, el tiempo de implementación se reduce considerablemente considerando con otros lenguajes de programación. Al ser lenguaje interpretado, es sencillo de utilizarlo para realizar tareas de menor porte que requieran una cantidad de datos moderado. Se desconoce funcionamiento con gran cantidad de comentarios(potencialmente superior a el caso propuesto). Codificación legible, si se utilizan estándares como el propuesto en la implementación(PEP8), brinda a terceros la posibilidad de tener un código legible y simple a la lectura. Portabilidad con sistemas operativos no es 100% satisfactorio, para desarrollar existieron problemas específicamente de codificación de las palabras, en entornos UNIX no existe dicho problema, mientras que en sistemas Windows existieron los problemas mencionados, además de todo el proceso de instalación del lenguaje(tedioso en Windows) así como también las herramientas(Freeling). Salvo detalles menores es un lenguaje adecuado para utilizarlo para el procesamiento de datos a pequeña y mediana escala. Al brindar características de un paradigma funcional(map, zip, etc), brinda simplicidad y 'ahorro' en cantidad de líneas utilizadas en total. La unificación derivada de la programación lógica, combinada con paradigma funcional hace un lenguaje poderoso con lo que refiere manipulación de datos en general, una característica deseable también como ser la posibilidad de generar APIs con swig y combinar la eficiencia de c++ en la ejecución de algoritmos que requieran uso excesivo del recurso CPU.

7.2 Resultados

Se lograron resultados satisfactorios a lo que refiere la implementación del clasificador, la utilización de la librería *nltk* permite un manejo sencillo y una representación trivial del problema a resolver. Los conjuntos utilizados para las pruebas se crearon dinámicamente sin mayores problemas, gracias a la versatilidad del lenguaje de programación. El análisis estadístico fue posible implementarlo de forma sencilla a lo que refiere la ejecución de las mismas. Se utilizaron herramientas para el cálculo de desviación estándar y estimadores de DAgostino como ser GNU-Octave, que al igual que Python brinda un nivel de abstracción alto para el procesamiento de gran cantidad de datos. Si bien como se mencionó en la sección Resultados correspondiente al Análisis, no se llegó a la meta principal de lograr un *accuracy* del 90%, en gran parte por la problemática de clasificar los comentarios negativos, así como no contar con un CORPUS lo suficientemente grande, como para obtener mejores datos en el momento de implementar atributos que eleven la calidad tanto el *Precision* como *Recall*. Se estima que dados los resultados del laboratorio y todo el procesamiento de datos logrado, es posible mejorar el clasificador en un 5% o 10% si se cuenta con mayor tiempo para la realización de la tarea, de esta forma se lograría realizar un estudio más exhaustivo utilizando diferentes corpus para recolectar información más precisa, de esta forma generar un clasificador universal para el dominio de comentarios relativos a carteleras de cine. De todas formas conociendo las dificultades existentes en el problema, se arriba a que el estudio realizado ha aportado una herramienta muy poderosa en el momento de procesar y clasificar elementos, en este caso comentarios, pero el mismo puede ser aplicable en diversas áreas, ya sea para detectar errores o predecir comportamientos en determinado contexto.