Tarea 1
IPLN
Grupo 9

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

September 29, 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 servico Freeling. 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 al requerimiento 4.b 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: $analyze\ -f\ ./config/es.cfg$

$--server\ --port$ 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 descrito.

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.

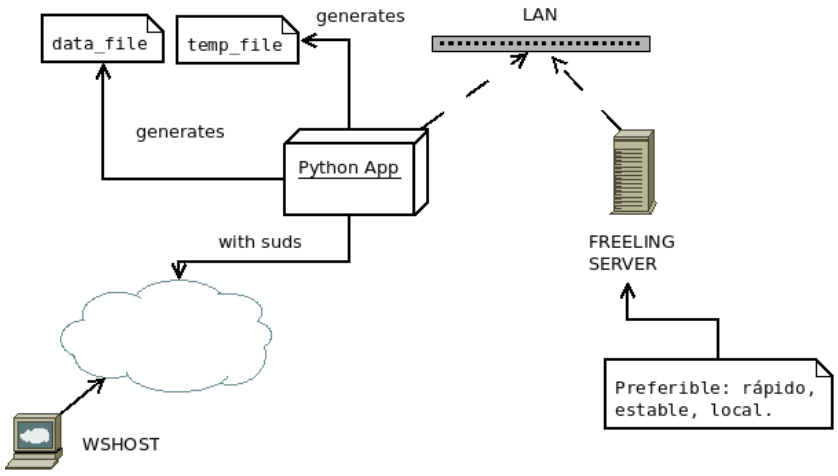

Figure 1: Interacción

5 Implementación

Se procede a la descripción de la implementación del programa. Para ello se utilizaron dos 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.

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

5.1 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.

5.2 loadStopwords

Encabezado: $def\ loadStopwords(self)$ 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.

5.3 loadSubjetiveElems

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

5.4 lematization_freeling_client

Encabezado: $def\ lematization\_freeling\_client(self, comment, stopwords)$.
Argumentos:

- *comment* : Comentario particular a ser procesado.
- *stopwords* : Se trata de un diccionario con las palabras vacías a ser excluídas en nuestro procesamiento.

Su funcionalidad principal es tomar un comentario específico mediante una string, guardarlo en un archivo particular llamado $temp_file$, para invocar la orden que se encargará de enviarle los datos al servidor local Freeling. Posterior a dicho procedimiento, se procede a la lectura del archivo de salida que genera el servidor, el mismo se encuentra en el archivo temporal $data_file$, de esta forma se lee completamente el archivo y se extra el tag, lema y probabilidad de todas las palabras presentes en el comentario. Se utiliza como función para garantizar la legibilidad de la función principal *main*.

5.5 plot

Encabezado: $def\ plot(self, listaPlot, n\_groups, filename)$
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.

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.

5.6 main

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.

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

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.

5.7 Eficiencia

La complejidad del un programa se reduce a las estructuras utilizadas, la correcta utilización de las mismas implica implica 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], 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.

6.4 Relación léxico-puntuación

En esta subsección se comentarán los puntos 4.a y 4.b presentes en la letra del Obligatorio.

6.4.1 Parte a

Como se describió anteriormente, se generaron diccionarios que contienen las palabras positivas y negativas[6.2], una vez se tienen estos diccionarios, se utiliza la función *sorted* brindada por Python, para ordenar los diccionarios según los valores indexados(no la llave), se utiliza la clase *operator* para dicho requerimiento. De esta forma se obtienen dos listas con elementos ordenados de forma decreciente, para las palabras positivas y negativas. Posteriormente se seleccionan los primeros cien elementos de cada lista, y con esto se satisface el punto a. Debido a que en el proceso de *lematization* ya se realiza el filtrado de las palabras excluyendo las palabras vacías, las listas con cien elementos queda totalmente definida y cumpliendo con los requerimientos.

6.4.2 Parte b

Una vez se obtienen las listas ordenadas con los cien elementos positivos y negativos es fácil realizar la intersección, para ello se transforma en un conjunto las listas recién mencionadas y se utiliza *intersection* propiedad de conjuntos definida en Python, con las llaves de las palabras negativas y positivas obtenidas en el momento de incorporar la lista de elementos subjetivos(genera diccionarios $(positiveWords, negativeWords)$). De esa forma obtenemos una lista con la intersección de elementos positivos y negativos. Posteriormente por eficiencia se transforma dicha lista en un diccionario con los valores de los diccionarios generados en el proceso *lematization*, finalmente se ordena el diccionario y se procesan los datos con la función *plot* en diversos órdenes[5.5]. En las gráficas se describen los valores porcentuales de las palabras en cuestión, con el fin de generar una idea sobre la utilización de las mismas en la totalidad de los comentarios.

6.5 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', 'película', 'ver', 'haber', 'historia'. 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 'ver', 'película', 'haber', 'bueno', 'malo', 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/*.

6.5.1 Lemas

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

6.5.2 Comentarios

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

6.5.3 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.

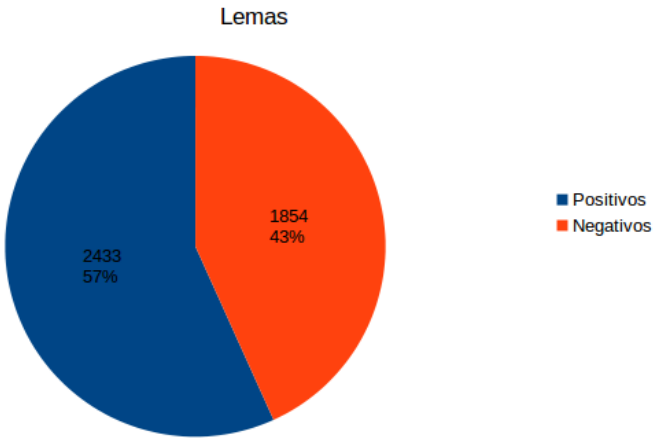

Figure 2: Cantidad de lemas

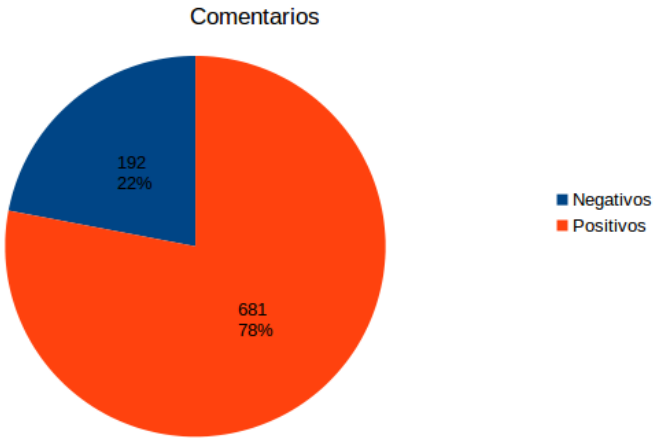

Figure 3: Cantidad de comentarios

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. Se observa la presencia de las palabras "ver", "película"', "haber","pelicula","bueno" ,"gustar" entre las primeras con mayor frecuencia, como también sucede en el caso de la gráfica de todas las palabras.

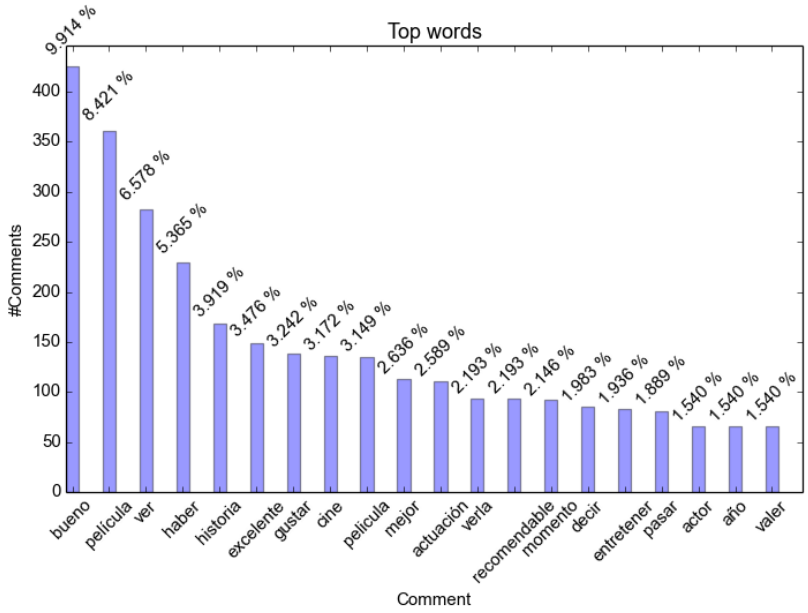

Figure 4: Todas las palabras

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 obtenegan resultados no del todo fiables. Se observa la presencia de las palabras "ver", "película"', "haber","pelicula","bueno" ,"gustar" 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 "película" 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 terminos "bueno" y "película" con respecto a los otros terminos en su

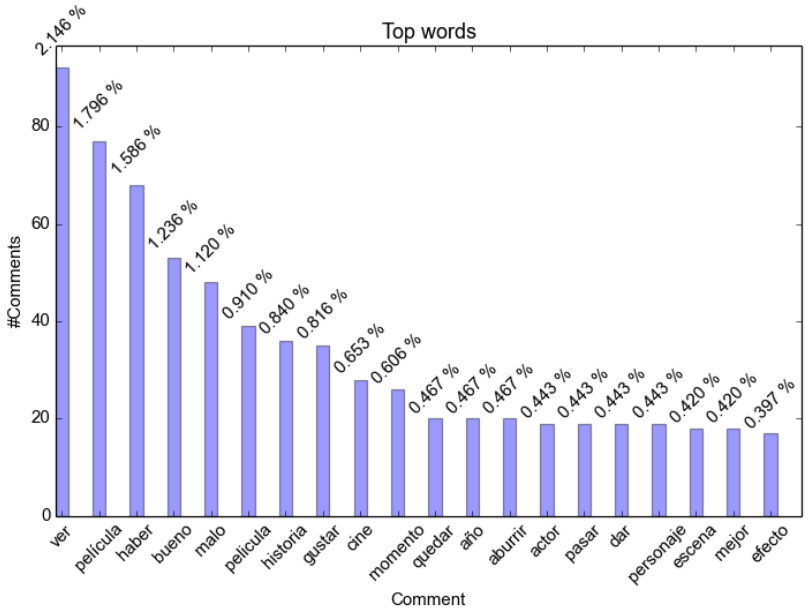

Figure 5: Todas las palabras negativas

frecuencia. Estos terminos también son los términos de mayor frecuencia en la gráfica Top palabras positivas.

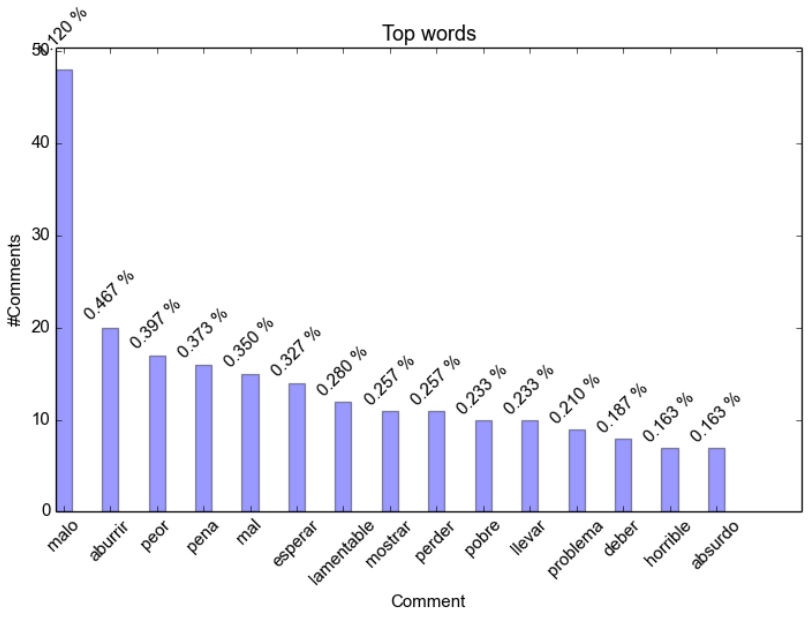

Figure 6: Intersección palabras negativas

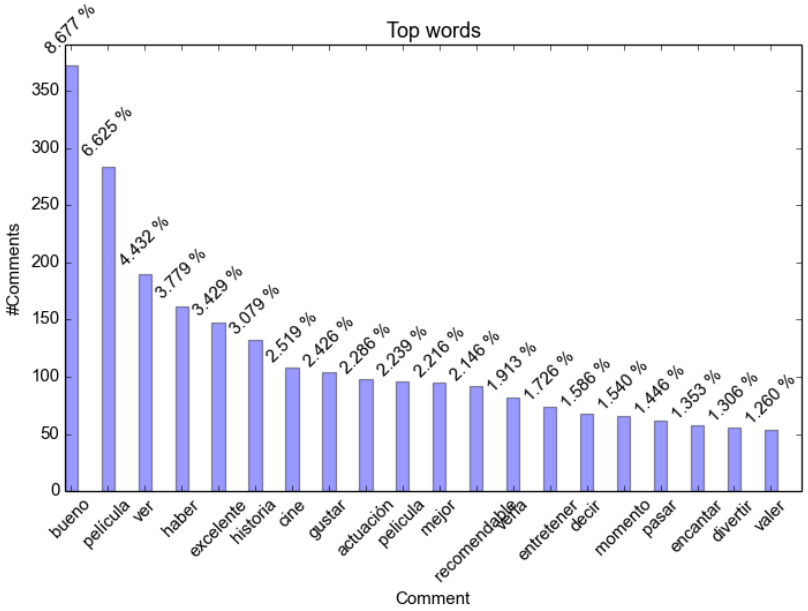

Figure 7: Todas las palabras positivas

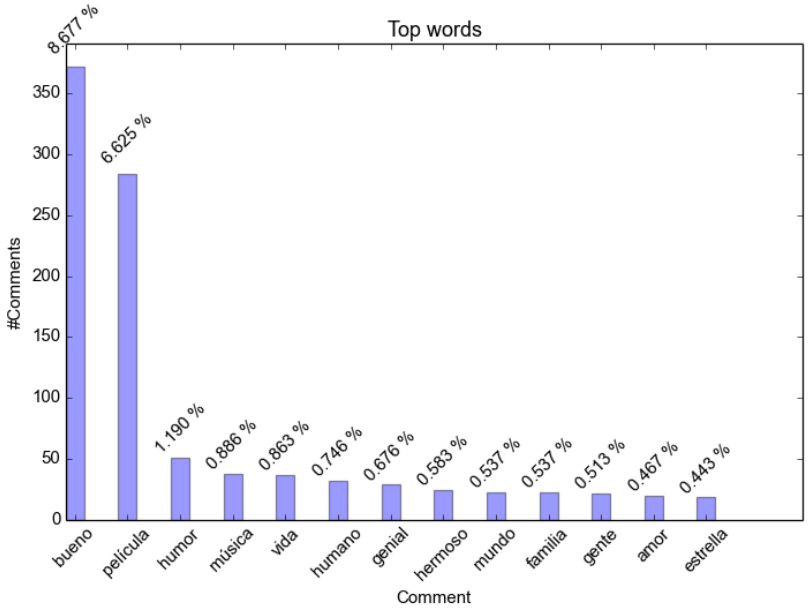

Figure 8: Intersección palabras positivas

7 Conclusiones

Se detallan algunas puntos a tener en cuenta luego de finalizada la presente tarea, se realizan algunos comentarios sobre la implementación.

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.