UNIVERSIDAD CATOLICA
DE LA SANTISIMA CONCEPCION

FACULTAD DE INGENIERIA

Desarrollo de una Interfaz de usuario para análisis de sentimientos basada en Taverna Workbench

POR:
RICARDO ALFREDO CARTES VALENZUELA

TESIS PRESENTADA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL INFORMÁTICO

Profesor Guía:
Claudia Martínez Araneda

Comisión informante:
José Abreu Salas

Concepción, Chile
2018

Resumen

Existe un concepto llamado **Minería de opiniones o Análisis de sentimientos**, el cual consiste en aplicar diversas técnicas del procesamiento del lenguaje natural a la información generada por usuarios a través de distintas plataformas digitales y así obtener un valor tangible y directo, como puede ser la polaridad (positivo, negátivo), o bien realizar un análisis mas avanzado y saber que sentimiento se está expresando.

El aplicar estas técnicas mediante las distintas herramientas proporcionadas por R Project resulta súmamente difícil de implementar. Por lo cual, para dar apoyo a los interesados, se construyó una interfaz gráfica configurable que permita realizar las diversas tareas involucradas en el análisis de sentimientos, desde aquellas orientadas a la extracción como a la clasificación.

Agradecimientos

Quiero dar las gracias a muchas personas que han estado en este largo camino que pareciera estar llegando a su fin. Primero agradecer a mi abuela Q.E.P.D, quién siempre tuvo la ilusión puesta en que iba a llegar a ser el primer profesional de la familia y estoy a punto de cumplirlo. También, agradecer a mi madre, quién siempre ha estado apoyándome desde un comienzo, aconsejándome en momentos de complejidad, en donde muchas veces me sentí mal anímicamente debido a algunas malas notas que tuve en este largo camino. También agradecer al resto de mi familia, quienes han estado atentos, preguntando constantemente por mis avances.

Índice general

1. Introducción **12**

1.1. Presentación del Problema 12
1.2. Justificación del Problema 13
1.3. Delimitación del Problema 13
1.4. Objetivos 13
1.4.1. Objetivo General 13
1.4.2. Objetivos Específicos 14
1.4.3. Metodología 14

2. Marco Teórico **16**

2.1. Procesamiento del lenguaje natural "PLN" 16
2.2. Análisis de sentimientos o Minería de opiniones 16
2.3. Microblogging 18
2.4. Twitter 18
2.5. R Project 19
2.6. RStudio 19
2.7. Interfaz de Programación de Aplicaciones (API) 19
2.8. twitterR 20
2.8.1. La clase Status 20
2.8.2. Clase User 21
2.8.3. Funcionalidades 22
2.9. Taverna Workfow 27
2.10. Stemming 28
2.11. Manejo de negaciones 28

3. Estado del Arte **29**
3.1. Aplicación web SENTIMENT140 29
3.2. Paquete SENTIMET140 31
3.3. Indico 31
3.4. Paquete Indico 32
3.5. R temis 32
3.6. Bitext Sentiment Analysis 33

4. Especificación de requisitos **34**
4.1. Introducción 34
4.1.1. Propósito del sistema 34
4.1.2. Ámbito del sistema 34
4.2. Descripción general 35
4.2.1. Características de los usuarios 35
4.2.2. Restricciones 36
4.3. Requisitos específicos 37
4.3.1. Funcionalidades 37
4.3.2. Requisitos no funcionales 42

5. Diseño **44**
5.1. Diagrama de Actividad 44
5.1.1. Diagrama de actividad - Creación carpeta de trabajo 45
5.1.2. Diagrama de actividad - Cargar archivo de lexicón 45
5.1.3. Diagrama de actividad - Cambiar tamaño desplazamiento para manejo negaciones 46
5.1.4. Diagrama de actividad - Seleccionar tareas a realizar 47
5.1.5. Diagrama de actividad - Extracción de Tweets 47
5.1.6. Diagrama de actividad - Preprocesamiento 48
5.1.7. Diagrama de actividad - Clasificación 50
5.1.8. Diagrama de actividad - Creación archivo CSV final 51
5.2. Modelo de datos 52
5.3. Modelo del Workflow desarrollado 53

6. Implementación **57**
6.1. Estructura de la aplicación 57
6.1.1. Instalando paquetes necesarios 57
6.2. Antes de iniciar la aplicación 58
6.2.1. Iniciando el servidor 58
6.3. Creación de la carpeta de trabajo y configuración inicial 59
6.4. Menú de opciones 60
6.5. Módulo de configuración 62
6.5.1. Selección archivo lexicón 62
6.5.2. Cambiar tamaño desplazamiento para manejo de negaciones . 64
6.6. Primera etapa: Extracción 66
6.6.1. Como se extraen los Tweets? 68
6.7. Segunda etapa: Preprocesamiento 71
6.8. Tercera etapa: Clasificación 73
6.8.1. SubWorkflow Léxico 73
6.8.2. SubWorkflow APIs Aprendizaje automático 89
6.9. Cuarta etapa: Creación archivo CSV final 92

7. Pruebas **96**
7.1. Pruebas de usabilidad 96
7.2. Resultados prueba de usabilidad 97
7.3. Pruebas de requerimiento 98
7.4. Resultado Pruebas de requerimientos 98

8. Conclusión **99**
8.1. Cumplimiento de objetivos 99

Bibliografía **102**

Appendices **105**

A. **106**
A.1. Manual de usuario 106
A.1.1. **Instalando R Project:** 106

A.1.2. **Instalando Taverna Workflow** 107
A.1.3. Instalando paquetes necesarios 109
A.1.4. Iniciando el servidor 109
A.1.5. Iniciando la aplicación 110
A.1.6. Ejecutando la aplicación 112
A.1.7. Menú de opciones 112
A.1.8. Módulo de configuración 113
A.1.9. Extracción de Tweets 115
A.1.10. Preprocesando Tweets 118
A.1.11. Clasificación de Tweets 119
A.2. Diagramas modelo desarrollado 120
A.3. Pruebas de usabilidad 122

Índice de figuras

2.1. Workflow que hace una llamada a un servicio Python 27

3.1. Ejecutando aplicación SENTIMENT140 desde `http://www.sentiment140.com` 30

5.1. Diagrama de actividades para la funcionalidad creación carpeta de trabajo. 45
5.2. Diagrama de actividades para Carga de archivo lexicón. 45
5.3. Diagrama de actividades para Cambiar tamaño desplazamiento para manejo negaciones. 46
5.4. Diagrama de actividades para Selección tareas a realizar. 47
5.5. Diagrama de actividades para Extracción de tweets 1/2. 47
5.6. Diagrama de actividades para Extracción de tweets 2/2. 48
5.7. Diagrama de actividades para Preprocesamiento 1/2. 48
5.8. Diagrama de actividades para Preprocesamiento 2/2. 49
5.9. Diagrama de actividades para Clasificación. 50
5.10. Diagrama de actividad para Creación archivo CSV final. 51
5.11. Diagrama Vista del modelo parte 1/6. 53
5.12. Diagrama Vista del modelo parte 2/6. 54
5.13. Diagrama Vista del modelo parte 3/6. 55
5.14. Diagrama Vista del modelo parte 4/6. 55
5.15. Diagrama Vista del modelo parte 5/6. 56
5.16. Diagrama Vista del modelo parte 6/6. 56

6.1. Modelo representativo Tareas a realizar 60

6.2. Pantalla tareas a realizar 61
6.3. Modelo Módulo de configuración 62
6.4. Modelo Extracción 66
6.5. Solicitando cantidad de Tweets 66
6.6. Solicitando tipo de extracción de Tweets 67
6.7. Solicitando palabras claves para extraer Tweets 67
6.8. Solicitando Username para extraer Tweets 67
6.9. Solicitando Username para extraer Tweets 68
6.10. Dataframe que contiene titulares de noticias 69
6.11. Estructura del archivo CSV que contiene titulares de noticias 70
6.12. Estructura del archivo CSV que contiene titulares de noticias 71
6.13. Seleccionando archivo CSV para preprocesar 71
6.14. Modelo Clasificación 73
6.15. Modelo Clasificación por lexicón. 74
6.16. Seleccionando método de clasificación. 74
6.17. Diccionario de palabras, cada una con su polaridad asignada. Cabe destacar que este diccionario está adaptado para el uso de palabras chilenas. 75
6.18. Modelo APIS aprendizaje automático. 89
6.19. Seleccionando que API de aprendizaje utilizar 89
6.20. Proceso de creación archivo CSV final. Se crea al final de todo el proceso. 92

A.1. Iniciando R desde el menú inicio 107
A.2. Pantalla principal de R 107
A.3. Iniciando Taverna desde el menú inicio 108
A.4. Vista principal Taverna 108
A.5. Instalando los paquetes en R-Project 109
A.6. Servidor iniciado correctamente 110
A.7. Opción para abrir un Workflow 111
A.8. Buscando el Workflow 111
A.9. Vista de la aplicación 112
A.10. Vista del botón “Play” para ejecutar la aplicación 112
A.11. Vista del menú de opciones 113

A.12.Menú de configuración 114
A.13.Seleccionando tamaño de desplazamiento en caso de negaciones . . . 114
A.14.Ejemplo archivo lexicón 115
A.15.Seleccionando tipo extracción. Como ejemplo, se realizará una extracción por palabras claves 116
A.16.Ingresando cantidad de Tweets 116
A.17.Ingresando palabras claves 117
A.18.Vista archivo generado 117
A.19.Seleccionando archivo CSV a preprocesar 118
A.20.Seleccionando el enfoque de clasificación. Por ejemplo, léxicos 119
A.21.Seleccionando el método bolsa de palabras 119
A.22.Modelo del workflow desarrollado 120
A.23.Diagrama nodos del modelo y conexiones entre ellos 121

Índice de cuadros

2.1. Atributos de la clase Status 20
2.2. métodos de la clase Status 21
2.3. Métodos de la clase User 21
2.4. Atributos de la clase Status 22
2.5. Negadores y penalización a la polaridad 28

5.1. Diccionario de datos 52

7.1. Resultado promediado pruebas de usabilidad 97
7.2. Resultados pruebas de requerimientos 98
7.3. Resultados pruebas de requerimientos 98

A.1. Resultado pruebas de usabilidad individuo 1 122
A.2. Resultado pruebas de usabilidad individuo 2 123
A.3. Resultado pruebas de usabilidad individuo 3 123
A.4. Resultado pruebas de usabilidad individuo 4 124

Capítulo 1

Introducción

1.1. Presentación del Problema

El grupo de investigación SoMos (SOftware - MOdelling & Science) es un equipo de académicos de la Universidad del Bío-Bío y de la Universidad Católica de la Santísima Concepción, cuyo trabajo es apoyado por la dirección de Investigación y Facultad de Ciencias Empresariales de la Universidad del Bío-Bío. Este equipo se dedica a investigar sobre distintas áreas de las ciencias de la computación, como la lógica difusa, recuperación de información, evaluación de recursos digitales y procesamiento de lenguaje natural en donde se encuentra el análisis de sentimientos y afectos orientado a determinar polaridad y carga afectiva de textos escritos.

En el contexto del procesamiento del lenguaje natural (PLN), análisis de sentimientos o minería de opiniones se considera una técnica del PLN que permite determinar la polaridad de los textos escritos.

Una de las fuentes de opiniones escritas más utilizada hoy en día es Twitter y dentro de las librerías disponibles para su procesamiento se tiene a twitteR (Jeff Gentry, 2016), API que funciona como cliente de Twitter para R Project for Statistical Computing

La existencia de diversos paquetes y rutinas desarrolladas para R Project orientadas al análisis de subjetividad (TwitterR, SENTIMENT, CORPUS, RTextTools, entre otros) da origen a la necesidad de poder reutilizarlas e integrarlas en una única interfaz de usuario.

1.2. Justificación del Problema

A continación se presentan razones que justifican la realización del proyecto.

- Apoyar al trabajo realizado por el grupo de investigación SoMos sobre la minería de opiniones en textos escritos.
- Proporcionar una herramienta de apoyo a la investigación hacia personas con pocos conocimientos técnicos en programación y que estén interesadas en el tema minería de opiniones. De esta manera, dedicarían su esfuerzo netamente en realizar sus investigaciones, sin invertir tiempo y/o recursos en aprender algún lenguaje de programación específico.
- Aportar a la comunidad de desarrolladores con una interfaz reutilizable.

1.3. Delimitación del Problema

La realización de este proyecto se ve delimitada por los siguientes motivos:

- El software desarrollado será de carácter libre. Tanto documentación como código fuente estará a disposición de la comunidad, quien podrá distribuir, copiar, modificar y/o mejorar el producto realizado.
- La especificación de requisitos se debe basar en algunos aspectos de la norma IEEE830.
- Las herramientas utilizadas deben ser compatible con R Project.

1.4. Objetivos

En esta sección se definirá el objetivo general y los objetivos específicos que contribuyen en el cumplimiento de este proyecto.

1.4.1. Objetivo General

Desarrollar una interfaz de usuario para análisis de sentimientos basada en Taverna Workbench.

1.4.2. Objetivos Específicos

- Estudiar los conceptos de Procesamiento del Lenguaje Natural y Análisis de sentimientos.
- Especificar requisitos funcionales y no funcionales del software.
- Realizar diseño de la aplicación.
- Construir la aplicación.
- Realizar pruebas del sofware construido.
- Compartir el producto finalizado a la comunidad.

1.4.3. Metodología

El método de desarrollo utilizado es el *Modelo de desarrollo incremental*. Se realizaron cinco entregas del software, siendo la primera como base, la cual incluyó las funcionalidades más importantes. Cada entrega fue evaluada, y a partir de ella, se elaboraron las siguientes que contienen nuevas funcionalidades y mejoras con respecto a la anterior.

Actividades definidas por cada objetivo específico

1. Para el objetivo "Estudiar sobre Procesamiento del Lenguaje Natural y Análisis de sentimientos" se definieron las siguientes actividades.
    - Estudiar a nivel general la definición de PLN y Análisis de sentimientos desde la perspectiva de las tareas que se realizan.
    - Estudiar diversas herramientas que permitan la realización de las tareas involucradas en el análisis de sentimientos.
    - Realizar reuniones con profesor guía para resolver ciertas dudas que fueron apareciendo en la realización de este objetivo.
2. Para el objetivo "Especificar requisitos funcionales y no funcionales del software" se definieron las siguientes actividades.

- Realizar reuniones con las personas interesadas a fin de determinar aspectos requeridos del software.
- Redactar especificación de requisitos, basada en algunos items de la norma IEEE830.
- Entregar documento de requisitos a los stakeholders involucrados para verificar que ellos se encuentren satisfecho con lo redactado.

3. Para el objetivo "Realizar diseño de la aplicación". Se definieron las siguientes actividades.
   - Realización de diagrama de actividades.
   - Construir el modelo relacional de la base de datos.
   - Diseñar el modelo final del Software.
4. Para el objetivo "Construir la aplicación se definieron las siguientes actividades:
   - Estudiar la sintaxis de R Project y Taverna Workbench.
   - Aplicar conocimientos adquiridos y construir la aplicación.
   - Entregar versiones de la aplicación para ser evaluada. Este proceso es iterativo para cada versión.
5. Para el objetivo "Realizar pruebas del sofware construido" se definieron las siguientes actividades.
   - Selección de test adecuados al contexto del software.
   - Aplicar los test seleccionados y evaluar funcionamiento.
6. Para el objetivo "Compartir el producto finalizado a la comunidad" se definieron las siguientes actividades.
   - Buscar comunidades interesadas en análisis de sentimiento a traves de la web.
   - Compartir ejecutable, documentación y código fuente a las comunidades interesadas.

Capítulo 2

Marco Teórico

En este capítulo se dará a conocer todos los conceptos y fundamentos teóricos estudiados.

2.1. Procesamiento del lenguaje natural "PLN"

Disciplina de la computación que se ocupa de la formulación e investigación de mecanismos computacionales para la comunicación entre personas y máquinas mediante el uso de Lenguajes Naturales (Carbonell, 2018).

2.2. Análisis de sentimientos o Minería de opiniones

Técnica de **Procesamiento del Lenguaje natural** permite clasificar los textos escritos de acuerdo a su polaridad (positivo, negativo, neutro). Esta es la clasificación más básica, ya que mediante métodos más avanzados se puede determinar la intensidad afectiva de los textos escritos.

Realizar análisis de sentimiento resulta ser una tarea compleja debido a las distintas características de los textos escritos. Esta técnica de la PLN posee diversos enfoques de trabajo:

1. Lexicones: Consiste en buscar cada palabra de un texto y asignar un valor de

polaridad. De este enfoque se desprenden dos métodos:

a) Bolsa de palabras: Consiste en el uso de un diccionario que posee palabras claves, en la que cada una tiene una ponderación de polaridad.

b) Método de orientación semántica basada en informacin mutua puntual: Método para inferir la orientación semantica de una palabra basado en estadística. Consiste en asociar una palabra a un conjunto "pivote" de palabras positivas y negativas para luego determinar la probabidad de que esta palabra se direccione a cierta polaridad. Este método se basa en la idea que una palabra con polaridad positiva tenga mas posibilidades de coexistir con otra positiva, lo mismo ocurre con una palabra con polaridad negativa (Turney y Peter D, 2002). Para calcular la fuerza de asociación, se evalua la concuerencia de estas asociaciones de palabras a través de un buscador web.

   Para evaluar la polaridad se utiliza un conjunto de palabras "pivotes", las cuales son las siguientes.

   - Positivas: bueno, agradable, excelente, positivo, afortunado, correcto, superior.
   - Negativas: malo, desagradable, pésicmo, negativo, desafortunado, incorrecto, inferior.

   La fórmula de Turney para determinar la polaridad es la siguiente:

$$\log_2 \frac{\prod_{hits(wNEARw+)} * \prod_{hits(w-)}}{\prod_{hits(wNEARW-)} * \prod_{hits(w+)}} \tag{2.1}$$

2. Aprendizaje de máquina: Consiste en la utilización de un conjunto de entrenamiento, el cual incluye textos ya polarizados. Se entrena el modelo de predicción y luego se pone a prueba.

Fases del análisis de sentimientos

1. **Extracción de datos:** Se extraen datos desde diversas fuentes de información. En este caso, se obtendrán Tweets de diversos usuarios desde Twitter.

2. **Preprocesamiento:** Una vez extraída la información, esta debe ser sometida a diversos procesos con el fin de obtener palabras claves para ser clasificadas. Algunos de estos procesos son la eliminación de caracteres especiales, URLs, stopwords, etc.

3. **Clasificación:** Una vez obtenidas las palabras claves del texto escrito y dependiendo del enfoque seleccionado, se clasifica el texto en positivo, negativo o neutro.

2.3. Microblogging

Servicio que permite a los usuarios enviar y publicar mensajes breves, generalmente de texto. Las opciones para el envío de mensajes varían de acuerdo a los distintos sitios web.

La principal y más popular característica de estos servicios es su sencillez y capacidad de sintetizar, ya que en pocos carácteres (140/280), se puede contar desde qué se está haciendo, anunciar cosas, promocionarse, etc (Roberto Gonzlez-Ibez et al., 2017).

2.4. Twitter

Servicio de **microblogging** que permite enviar mensajes de texto plano con un máximo de 280 carácteres llamados Tweets que se muestran en la página principal del usuario (Twitter, 2017). Estos pueden suscribirse a los Tweets de otros usuarios. Por defecto, los mensajes son públicos, pudiendo difundirse a la comunidad. El usuario tiene la posibilidad de establecerlos como privados, mostrándolos únicamente a ciertos seguidores determinados.

Los usuarios pueden "twittear" desde la web del servicio, utilizando apps oficiales o externas (como para teléfonos inteligentes), o mediante el Servicio de mensajes cortos (SMS) disponible en ciertos países.

Actualmente posee más de 500 millones de usuarios, generando un aproximado de 65 millones de Tweets por día, siendo este servicio una muy buena fuente de datos para realizar análisis de sentimientos.

2.5. R Project

Es un lenguaje y entorno para la informática estadística y gráficos. Proyecto GNU desarrollado en los laboratorios Bell por Johan Chambers y colegas.

R proporciona una gran cantidad de modelos estadísticos (lineales, no lineales, clasificación, agrupamiento, entre otros) (The R Foundation, 2017).

Representa un conjunto integrado de aplicaciones de software para la manipulación de datos, cálculo y visualización gráfica.

Está disponible como software libre bajo los términos de las licencias públicas GNU de la Free Software Fundation.

2.6. RStudio

Entorno de desarrollo integrado (IDE) para R Project. Incluye una consola, editor de texto que mediante el resaltado permite la ejecución directa del código seleccionado. Incluye herramientas para seguimiento de variables, historial, depuración y gestión del espacio de trabajo.

RStudio está disponible en ediciones comerciales y de código abierto, se ejecuta en distintas plataformas como Windows, Mac y Linux (RStudio, 2017).

2.7. Interfaz de Programación de Aplicaciones (API)

Conjunto de funciones y procedimientos que cumplen una o muchas funciones con el fin de ser utilizadas por otro software. La sigla API viene del inglés *Application Programming Interface*. En términos de programación se puede catalogar como una **capa de abstracción**. Para la realización de este proyecto, se utilizó el API de Twitter, la cual proporciona diversas funcionalidades en la etapa de extracción de Tweets.

2.8. twitterR

Paquete de R Project que proporciona comunicación con la **API de Twitter**. Permite un enlace a los servidores de la red, dando posibilidades al usuario de acceder a las funcionalidades que proporciona esta API, el uso de ellas depende que el usuario se encuentre autentificado, ya que hay muchos datos de caracter privado, negándole el permiso a los usuarios no autentificados (Jeff Gentry, 2016).

2.8.1. La clase Status

Clase que contiene toda la información obtenida de un estado (Tweet), incluyendo el texto e información relacionada con el emisor de este.

En la tabla 2.1 se describen los atributos de esta clase:

Atributo	Descripción
text	El texto del estado.
screenName	Nombre de pantalla del usuario que escribió el estado.
id	ID del estado.
replyToSN	Nombre del usuario a quien respondió, esto en caso de ser respuesta.
replyToUID	ID del usuario a quien respondió, esto en caso de ser respuesta.
StatusSource	Origen del Tweet.
created	Fecha de creación del Tweet.
truncated	Si el estado fue truncado.
favorited	Si el estado es favorito.
reTweeted	TRUE si el estado ha sido un reTweet.
reTweetCount	El número de veces que el estado ha sido reTweeteado.

Cuadro 2.1: Atributos de la clase Status

En la tabla 2.2 se describen los métodos de esta clase:

Método	Descripción
toDataFrame	Convierte este objeto en un data.frame de una fila, cada campo se representa por una columna.

Cuadro 2.2: métodos de la clase Status

2.8.2. Clase User

Clase que representa a un usuario de Twitter. Contiene diversos datos asociades a un individuo.

En la tabla 2.3 se describen los atributos de esta clase :

Atributo	Descripción
name	Nombre del usuario.
screenName	Nombre mostrado por el usuario.
id	ID del usuario.
lastStatus	último estado que escribio el usuario.
description	Descripcón del usuario.
StatusesCount	Número de veces que el usuario ha actualizado su estado.
followersCount	Cantidad de seguidores.
favoritesCount	Cantidad de seguidos.
friendsCount	Cantidad de amigos.
url	URL del usuario.
created	Cuando fue creada la cuenta del usuario.
protected	Si la cuenta de usuario es protegida.
verified	Si la cuenta de usuario es verificada.
location	Ubicación del usuario.
listedCount	Cantidad de veces que el usuario aparece en listas públicas.
followRequestSent	Será verdadero si el usuario ha enviado una solicitud de amistad al usuario autentíficado.
profileImageUrl	URL de la imagen de perfil del usuario si es que existe.

Cuadro 2.3: Métodos de la clase User

Los métodos de esta clase están descritos en la tabla 2.4:

Método	Descripción
getFollowerIDs	Retorna una lista que contendrá los IDs de las personas seguidas por el usuario.
getFollowers	Retorna una lista que contendrá objetos User, los cuales representa a las personas seguidas por el usuario.
getFriendIDs	Retorna una lista que contendrá los IDs de los amigos del usuario.
getFriends:	CRetorna una lista que contendrá objetos User, los cuales representa a las personas amigas por el usuario.
toDataFrame	Convierte este objeto en un data.frame de una fila, cada campo se representa por una columna.

Cuadro 2.4: Atributos de la clase Status

2.8.3. Funcionalidades

A continuación se describen las funcionalidades mas relevantes de este paquete:

- decode_short_url: Función que decodifica dirección web acortada.
  - Entrada: url (acortada, cadena de caracteres).
  - Salida: URL original (cadena de caracteres).
- favorites: Función que obtiene los Tweets favoritos de un usuario específico.
  - Entrada:
    - User: Usuario de Twitter (cadena de caracteres y objeto User).
    - n: Número máximo de Tweets a obtener (número entero).
    - max_id: ID máximo de Tweets a buscar.
    - sice_id:ID mínimo de Tweets a buscar.
  - Salida: Lista de n objetos Status.

- friendships: Función que detalla la relación entre el usuario autentificado con otros.
  - Entrada:
    - screen_name: Vector de uno o mas nombre de usuario de Twitter.
    - User_ids: Vector de uno o mas id de usuario de Twitter.
  - Salida:data.frame que contiene el id y nombre del usuario, siguiendo, seguido. Las dos ultimas columnas son BOOLEANAS, siendo valor TRUE en caso que el usuario autentficado siga o es seguido por el resto de los usuarios, caso contrario será FALSE.
- getTrends: Función que obtiene las tendencias en una ubicación en específico.
  - Entrada:
    - woeid: Número que identifica la localización en donde se obtendrá la tendencia actual.
    - lat: Numérico que representa latitud, valor entre -180 y 180. Oeste es negativo, Este es positivo.
    - long: Numérico que representa longitud, valor entre -180 y 180. Sur es negativo, Norte es positivo.
    - exclude: Si se establece en null, excluirá hashtags.
  - Salida: data.frame que contendrá detalles de las tendencias obtenidas para la ubicación dada en entrada.
- getUser: Función que permite obtener información sobre uno o varios usuarios de Twitter.
  - Entrada:
    - User: Puede ser una cadena de caracteres que especifica el nombre del usuario o también un objeto User.
    - Users: Un vector que incluya el nombre o ID de usuarios a obtener información, nombre e ID pueden ser combinados.

    - includeNA: Si es TRUE, por cada usuario que no exista, se mostrará como NA.
  - Salida: Lista de objetos User.

- get_latest_Tweet_id: Función que devuelve el último Tweet almacenado en una base de datos.
  - Entrada: table_name (nombre de la tabla que contiene los Tweets).
  - Salida: ID del último Tweet encontrado en la tabla.

- import_Statuses: Función que permite importar datos de Tweets desde fuentes externas y los empaqueta en un objeto Status. Función experimental que soporta fuentes externas con formato JSON.
  - Entrada: raw_data (Datos a ser importados).
  - Salida: Lista de objetos Status.

- reTweets: Función que permite obtener los reTweets para un Tweet en específico.
  - Entrada:
    - id: ID del Tweet a obtener sus reTweets.
    - n: Numérico, cantidad de reTweets a obtener. El máximo de este valor es 100.
  - Salida: Lista de reTweets para el Tweet dado en la entrada (Lista de objetos Status).

- searchTwitter: Función que permite hace una búsqueda de Tweets basada en una cadena de caracteres proporcionada.
  - Entrada:
    - searchString: Cadena de caracteres que indica la palabra que los Tweets a buscar deben contener. Usar + para separar términos de consulta.

- n: Numérico. Indica el máximo de Tweets a buscar.
- lang: Si no es NULL, restringe los Tweets al idioma dado. Se debe ingresar un código ISO 639-1.
- since: Si no es NULL, restringe los Tweets a buscar desde la fecha dada. Se debe ingresar en formato YYYY-MM-DD.
- until: Si no es NULL, restringe los Tweets a buscar hasta la fecha dada. Se debe ingresar en formato YYYY-MM-DD.
- locale: Si no es NULL, configurará la congifuración regional para la búsqueda.
- geocode: Si no es NULL, devuelve Tweets de los usarios ubicados dentro de un radio dado en longitud y latitud. La forma de ingresar este parámetro es latitud, longitud, radio (millas o km). Por ejemplo geocode='37.781157,-122.39720,1mi'.
- sinceID: Si no es NULL, restringe los Tweets a buscar desde el ID especificado.
- maxID: Si no es NULL, restringe los Tweets a buscar hasta el ID especificado.
- resultType: Si no es NULL, restringe los Tweets a buscar según el valor dado. Los valores permitidos son:
  - recent: Retorna el máximo de resultados recientes.
  - popular: Retorna sólo los resultados mas populares.
- retryOnRateLimit: Si no es 0, se realizarán X búsquedas. Esto puede relentizar el proceso, pero asegurará una búsqueda completa.

- Salida: Lista de objetos tipo Status.

- setup_twitter_oauth: Función que configura las credenciales para una sesión de twitterR.

  - Entrada:
    - consumer_key: Llave de cunsumidor suministrada por Twitter.
    - consumer_secret: Clave de consumidor suministrada por Twitter.
    - access_token: Token de acceso suministrada por Twitter.

- access_secret: Clave suministrada por Twitter.

- Salida: Sin Valor de retorno.

- showStatus: Función que permite buscar Tweets específicos.
  - Entrada:
    - id: ID del Tweet a buscar. El valor puede ser un String o numérico.
    - id's: Un vectoe que contenga los IDs a buscar. El vector puede estar combinado con valores numéricos o cadenas de caracteres.
  - Salida: Un objeto Status o una lista de objetos de tipo Status, dependiendo del parámetro de entrada.

- strip_retweets: Dada una lista de Tweets. La función eliminará los reTweets de la lista para proporcionar un conjunto puro de Tweets.
  - Entrada: Tweets (Lista de objetos Status).
  - Salida: Lista de objetos Status con los reTweets removidos.

- timelines: Función que permite recuperar varias lineas de tiempo desde el universo Twitter.
  - Entrada:
    - User: El usuario de Twitter a obtener su linea de tiempo. Puede ser un String o un objeto User.
    - n: El tamaño máximo de Tweets a obtener. Máximo 3200.
    - maxID: ID máximo para buscar Tweets.
    - sinceID: ID mínimo para buscar Tweets.
    - includeRts: Si es false, no incluirá reTweets.
    - excludeReplies: Si es true, cualquier respuesta se eliminará de los resultados.
  - Salida: Lista de objetos Status.

- twListToDF: Función que transforma una lista de Tweets a un data.frame.

- Entrada: twList (Lista de objetos Status).
- Salida: data.frame corresponiente a la lista de Tweets.

2.9. Taverna Workfow

Sistema de gestión de flujos de trabajos científicos de código abierto. Se cataloga como un conjunto de herramientas utilizadas para diseñar y ejecutar estos flujos de trabajo (Katherine Wolstencroft et al., 2013).

Esta herramienta es compatible con diversas plataformas, carácteristica importante, ya que a través de un navegador de escritorio se pueden ejecutar estos flujos de trabajo, sin requerir programa alguno.

Además, esta herramienta promueve el uso de la reutilización, ya que, si se desea usar un componente de un proceso en otro, sólo basta integrar este componente al nuevo proceso, procurando que las entradas y salidas sean convincentes con el el resto de los componentes del proceso.

Taverna permite a los usuarios integrar muchos componentes de software diferentes, por lo cual es utilizado en muchos dominios, como medicina, astronomía, bioinformática (Robert D. Stevens et al., 2003). Además, los flujos de trabajos realizados en Taverna se puede compartir con otras personas a través del sitio MyExperiment (The University of Manchester y University of Southampton, 2018).

En la figura 2.1 se aprecia un ejemplo de worfkflow básico realizado en Taverna.

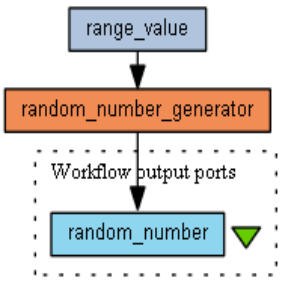


Figura 2.1: Workflow que hace una llamada a un servicio Python

2.10. Stemming

Es un método para reducir una palabra a su raiz. Permite aumentar la cantidad de elementos que se puedan encontrar en una consulta.

Cuando queremos buscar una palabra, tal vez nos interese utilizar este método para recuperar más elementos relacionados con la palabra. Por ejemplo si buscamos "biblioteca" en forma pura, nos devolverá los registros que contengan esta palabra. pero si aplicamos stemming, obtendremos aquellas palabras con igual raíz léxica como "biblioteconomía" o "bibliotecario".

2.11. Manejo de negaciones

Uno de los enfoques consiste en invertir la polaridad del elemento léxico encontrado al lado de un negador, cambiando bueno (+3) a no bueno (-3) (Maite Taboada et al., 2011). Este enfoque mejorado considera un tamaño de desplazamiento, el cual indica la cantidad máxima de palabras a desplazar en caso de encontrar un negador. Si se encuentra una palabra positiva dentro de este límite, se le invierte la polaridad. Caso contrario, se busca otro posible negador.

Otro enfoque, consiste en restar una cantidad fija a la polaridad en caso de existir un negador previo a una palabra polarizada. Esta cantidad fija a restar depende del negador encontrado. En la tabla 2.5 observamos la cantidad a descontar a la polaridad.

Negador	Cantidad a descontar
No, nunca	4
Sin	3.5

Cuadro 2.5: Negadores y penalización a la polaridad

Capítulo 3

Estado del Arte

En este capítulo se darán a conocer distintas herramientas disponibles para al análisis de sentimiento, desde las orientadas a la extracción como también a las de preprocesamiento y clasificación.

3.1. Aplicación web SENTIMENT140

Aplicación que permite descubrir el sentimiento de un texto dado como parámetro de entrada. (Alec Go Lei Huang et al., 2017). Ver figura 3.1.

Esta aplicación posee las siguientes características:

- Basado en aprendizaje automático: Se obtienen mejores resultados vs un clasificador basado en diccionarios, ya que no está limitado a realizar la búsqueda en un conjunto finito de elementos, evitando una mal clasificación por la no existencia de algunas palabras claves.

- Uso de interfaz gráfica: No requiere la necesidad de saber programar.

- Transparencia para los resultados de clasificación de clasificación de Tweets individuales: Esto se puede considerar una ventaja o una limitación, ya que permite mostrar un resultado en forma directa. Es decir, al momento de ingresar un tweer, se mostrará que polaridad se ha encontrado, esto en parte es una limitación, ya que no se puede obtener mas información del texto.

- No es de código abierto, pero proporciona un API de desarrollo que permite la integración sin complicaciones con otro software.

Figura 3.1: Ejecutando aplicación SENTIMENT140 desde `http://www.sentiment140.com`

SENTIMENT140 utiliza un efoque basado en el aprendizaje supervisado por distancia. Como cualquier otro clasificador, necesita un conjunto de entrenamiento. Segun su página oficial, este conjunto se basa en un arhivo CSV que se compone de los siguientes elementos:

- Polaridad de Tweet: (0=negativo, 2=neutral, 4=positivo)
- El id del Tweet (2087)
- Fecha del Tweet (sábado 16 de mayo 23: 58:44 UTC 2009)
- El usuario que hizo el Tweet.
- El texto del Tweet.

El autor proporciona el conjunto de entrenamiento para textos escritos en inglés. Lamentablemente, no nos da acceso al conjunto de entrenamiento de Tweets en español. Por lo que se debe hacer uso del API proporcionada para trabajar con textos escritos en este idioma.

3.2. Paquete SENTIMET140

Paquete para R Project que permite realizar análisis de sentimiento en textos escritos a través de la aplicación SENTIMENT140 (Yanchang Zhao, 2016).

Este paquete posee las siguientes carácteristicas:

- Clasificación basada en aprendizaje automático: Se obtienen mejores resultados vs un clasificador basado en diccionarios, ya que no está limitado a realizar la búsqueda en un conjunto finito de elementos, evitando una mal clasificación por la no existencia de algunas palabras claves.
- Entrenamiento rápido sin uso de PLN: Basta con sólo ingresar el texto del Tweet a analizar, sin la necesidad de realizar tareas de preprocesamiento. Este paquete soporta textos en inglés y en español.
- Fácil implementación: Sólo se debe hacer un llamado a la función que permita obtener la polaridad de los textos escritos. Esta nos retorna el conjunto de textos junto a la polaridad detectada.

Este paquete proporciona un acceso al API de SENTIMENT140 web

3.3. Indico

Conjunto de aplicaciones que emplean diversas técnicas de aprendizaje automático en el ambiente empresarial. Utiliza un efoque conocido como aprendizaje por transferencia. Según su página oficial, permite entrenar modelos de aprendizaje de una menor magnitud de datos en comparación a otros modelos. (Indico, 2017)

Es una técnica que permite tomar una red neuronal entrenada para resolver una tarea y ajustarla pa poder resolver otra tarea.

- Clasificación basada en aprendizaje automático: Se obtienen mejores resultados vs un clasificador basado en diccionarios, ya que no está limitado a realizar la búsqueda en un conjunto finito de elementos, evitando una mal clasificación por la no existencia de algunas palabras claves.
- Predecir emociones: Permite predecir que emoción expresa el autor al momento de escribir un texto.

3.4. Paquete Indico

Paquete de R Project que permite hacer uso de la API Indico. A través de ella podemos acceder a funciones de clasificación de textos escritos.

Esta API fue utilizada en el desarrollo del Software.

- Fácil implementación en R: Sólo requiere su importación y llamado a la función de clasificación
- Requiere un API Key: Para utilizar este paquete, debemos registrarnos en la página oficial y obtener una llave de acceso.

3.5. R temis

Paquete de R Project que permite analizar, manipular y crear corpus de textos (Jos Pino-Daz, 2017).

Este paquete posee las siguientes características:

- Paquete con interfaz gráfica: Facilita ciertas acciones, pero no olvidar que es un paquete de R Project, por lo cual, se requiere un manejo en programación.
- Permite realizar preprocesamiento: Mediante su interfaz gráfica, permite ingresar textos y preprocesarlos, para luego ingresarlos a un clasificador de polaridad.
- Permite comparar distintos documentos y evaluar similitud: Permite detectar patrones de oraciones repetidas en los distintos textos, como también, búsqueda de palabras que aparezcan en los diversos textos a comparar.

- Herramienta gratis: De acuerdo a la GNU Free Software Fundation.

3.6. Bitext Sentiment Analysis

Aplicación orientada al análisis de textos escritos. Está enfocada a empresas que desean obtener información sobre las opiniones vertidas por los usuarios de sus productos (Bitext- Official Page, 2017).

Sus características son las siguientes:

- Análisis de sentimientos basados en temas: Además de obtener la polaridad del texto analizado, permite determinar sobre que tema se está hablando, ya que esta aplicación posee un diccionario en constante crecimiento, el cual de acuerdo a las palabras encontradas, detecta un patrón que indica sobre que tema se está hablando.

- Dada una oración, detecta tantas opiniones como contenga esta: Por ejemplo, *Este teléfono es increíble, pero era demasiado caro y la pantalla no es lo suficientemente grande*

  A partir de esto, extraerán tres opiniones:

  - "Teléfono" + "impresionante"
  - "Teléfono" + "demasiado caro"
  - "Pantalla" + "no es lo suficientemente grande"

- Herramienta de pago: Solo de puede descargar una versión de prueba con funcionalidades limitadas. Una vez realizada la prueba, el usuario tiene la opción de comprar una versión completa.

Capítulo 4

Especificación de requisitos

En este capítulo se definirán los requisitos que posee la aplicación. Se incluirá una lista de requisitos funcionales, no funcionales y atributos de calidad.

4.1. Introducción

En el siguiente capítulo, se dará a conocer la especificación de requisitos para la interfaz gráfica a desarrollar. El desarrollo de esta especificación está basado en algunos aspectos de la norma IEEE-830

4.1.1. Propósito del sistema

El propósito de este capítulo es indicar que funcionalides realiza el sistema en cuestión. Estas son detalladas indicando entradas, proceso, salidas y restricciones para cada una de ellas. Esto permite al usuario conocer las capacidades y límites del sistema desarrollado. Como tambíen, en caso que otras personas deseen realizar mejoras, estas podrán recurrir a esta guía que detalla el funcionamiento de la aplicación.

4.1.2. Ámbito del sistema

El sistema desarrollado se llamó WorkFlowSentiment, el cual realizará 3 tareas principales. Siendo la de extracción de Tweets dado ciertos parámetros definidos

por el usuario, preprocesamiento de estos Tweets extraídos, permitiendo al usuario indicar que tipo de preprocesamiento se deberá realizar, y por último, la tarea de clasificación de los Tweets ya preprocesados mediante dos enfoques, de los cuáles, el usuario deberá especificar cuál o cuales utilizará. Por ejemplo, si en la actualidad se desea realizar una tarea de extracción con TwitterR se debe realizar un llamado a determinadas funciones, como lo ilustrado a continuación.

```
1  library(twitteR)
2  consumer_key<-"E6IR6qhsJXXzT8FlS0U6Su19s"
3  consumer_secret<-"EBweU7FSXqU5kEeK6T4uhHoJ4neIeTSLa0g6jYXtX5M74GhOif"
4  access_token<-"123433869-31ekSJJ52iSolwP7nHjwGdqFnO8A80mF17FgPAzZ"
5  access_secret<-"spEuOieJPeOmJAgbUddxbg62tXLHkSmDSKFGf8Cp3OAYT"
6  setup_twitter_oauth(consumer_key, consumer_secret, access_token,
       access_secret) #autentificacion en API Twitter
7  tweets<-searchTwitter("cadena de busqueda",lang = "es",n=10)
       #obtener tweets
```

Como se aprecia, es imprescindible poseer conocimientos en programación, resultando la realización de las tareas del análisis de sentimientos algo complejo para ciertos tipos de usuarios. La solución propone desarrollar una interfaz gráfica configurable usando workflows a través de Taverna, de modo que el usuario final no requiera conocimientos de programación. Se espera que esto contribuya a los investigadores en el campo de minería de opiniones.

Finalmente, es importante mencionar que mucho de los algoritmos utilizados en las 3 tareas ya se encuentran implementados en distintos paquetes de R Project.

4.2. Descripción general

4.2.1. Características de los usuarios

Para la realización de este proyecto se considera sólo un tipo de usuario:

- Interesado en minería de opiniones o análisis de sentimiento: Usuario que utiliza el sistema. Esta deberá indicar si desea recolectar Tweets por tema o usuarios,

seleccionar que enfoque de clasificación desea emplear para realizar el procesamiento de estos Tweets extraidos. Para luego, obtener la información procesada en un archivo de salida.

4.2.2. Restricciones

Las restricciones para el sistema son las siguientes:

- Plataforma compatible con Taverna Workflow (Windows, Linux, Mac).
- Conexión a internet, ya que las tareas de extracción la requieren.

4.3. Requisitos específicos

4.3.1. Funcionalidades

- **R1.-Configurar aplicación**
  Descripción: Conjunto de funcionalidades que permiten configurar distintos aspectos en el funcionamiento de la aplicación.

  - **R1.1.-Creación ruta principal de trabajo**
    Descripción: Funcionalidad que permite crear la carpeta principal de trabajo. Esta es indispensable para el funcionamiento de la aplicación, ya que en ella se almacenan los resultados obetenidos.
    Entrada:
    - Ruta absoluta del usuario.

    Proceso: Se comprueba la no existencia de la carpeta de trabajo. Si se cumple esta condición, el sistema obtiene la ruta absoluta del usuario (dada por el sistema operativo). Una vez obtenida, se crea en ella un directorio llamado sentiment_workflow, en el cual se almacenarán todos los archivos de salida generados en los procesos posteriores.
    Salidas:
    - **Salida 1:** Carpeta de trabajo creada.
    - **Salida 2:** Mensaje de error, en caso de algún imprevisto.
  - **R1.2.-Cargar archivo de lexicón**
    Descripción: Funcionalidad que permite cargar un archivo de lexicón. Este archivo es indispensable para la clasificación mediante bolsa de palabras, siendo este el diccionario que contendrá las palabras junto a su polaridad.
    Entrada:
    - Archivo xlsx correspondiente al lexicón.

    Proceso: Se desplegará un cuadro de selección de archivos en el cual deberá seleccionar el archivo de lexicoón provisto para el usuario. El sistema

comprobará internamente que el archivo ingresado sea el correspondiente. En caso de serlo, este se copiará a la ruta principal de trabajo. Caso contrario, se volverá a solicitar hasta haber ingresado un archivo correcto.
Salidas:

- **Salida 1:** Mensaje de confirmación en caso de ingreso correcto.
- **Salida 2:** Despliegue ventana selección archivo Lexicón en caso de un ingreso incorrecto.

- **R1.3.-Configurar desplazamiento para manejo de negaciones**
Descripción: Funcionalidad que permite configurar el máximo de palabras a avanzar en busca de una palabra polarizada.Esto en caso de detectar una negación.
Entrada:

  - Desplazamiento: Número entero.

  Proceso: El usuario deberá entrar la opción cambiar desplazamiento. Se desplegará una pantalla en la cual se deberá seleccionar la cantidad de desplazamiento. En caso de seleccionar la opción 0, se considerará el no utilizar manejo de negadores.
  Salidas:

  - **Salida 1:** Mensaje de confirmación al cambiar la cantidad de desplazamiento.

- **R2.-Selecciónar tareas a realizar**
Descripción: Funcionalidad que permite al usuario seleccionar que tarea o tareas va a realizar el sistema.

  Entrada:

  - Opción: Se desprenden 5 opciones.
    - Sólo extracción de Tweets.
    - Sólo preprocesamiento.
    - Sólo clasificación.
    - Extracción y preprocesamiento.

- Extracción, preprocesamiento y clasificación.

Proceso: Una vez iniciada la aplicación, el sistema desplegará una pantalla en la cual el usuario deberá seleccionar que tareas se realizarán. Una vez seleccionada la opción, se consultarán distintos parámetros dependiendo de la opción elegida. En caso de haber seleccionado la opción salida, se mostrará un mensaje de finalización.
Salidas:

- **Salida 1:** Pantalla del/los procesos elegidos y selección de parámetros para cada uno de ellos.
- **Salida 2:** Mensaje de indicando el cierre de la aplicación.

- **R3.-Extracción de Tweets**
  Descripción: Funcionalidad que permite al usuario extraer Tweets, los cuales posteriormente podrán ser preprocesados y/o clasificados.

  Entradas:

  - Cantidad de Tweets a extraer: Número entero, desde 1 a 100
  - Opción tipo de extracción: Se desprenden dos opciones.
    - Por palabras claves
    - Por usuario
  - Palabras claves: Cadena de caracteres.
  - Nombre de usuario: Cadena de caracteres.

  Proceso: Al comenzar el proceso de extracción, el sistema desplegará una pantalla en donde el usuario deberá ingresar la cantidad de Tweets a extraer (hasta un máximo de 100). Una vez ingresado este valor, se deberá indicar que tipo de extracción desea realizar. Si se selecciona la extracción por palabras claves, el usuario deberá escribir el la o las palabras claves a buscar a través de un cuadro de texto desplegado por pantalla. En caso de seleccionar extracción por nombre de usuario, se deberá ingresar mediante otro cuadro de texto el nombre del usuario emisor de Tweets. Ya ingresado estos datos, el sistema se

conecta al API de Twitter y comienza a realizar la extracción de acuerdo a los criterios ingresados anteriormente.
Salidas:

- **Salida 1:** Archivo CSV temporal que almacena los Tweets y metadatos asociados.
- **Salida 2:** Mensaje de error en caso de ingresar una cantidad inválida de Tweets para extraer.
- **Salida 3:** Mensaje de error en caso de no encontrar Tweets que cumplan con los requisitos dados por el usuario.

- **R4.-Preprocesamiento**
  Descripción: Funcionalidad que permite realizar una serie de tareas de tal manera de obtener un conjunto de Tweets preparados para ser clasificados.
  Entrada:

  - Archivo CSV que contenga Tweets extraídos.
  - Serie de Tweets obtenidos en **R3**.

  Proceso: En caso que el usuario haya seleccionado en **R2** una opción que incluya sólo preprocesamiento, se le solicitará que ingrese un archivo CSV, el cual debe contener los Tweets extraídos para realizar el proceso. Caso contrario, el sistema no requerirá del ingreso de un archivo CSV con los Tweets a procesar, ya que se considererán los Tweets extraídos en la etapa anterior (extracción). Para obtener un conjunto de datos listos para ser clasificados, el sistema realizará tareas de eliminación de caracteres especiales, reTweets, links, puntuaciones, espacios en blanco y stopwords y stemming, todo esto sobre los textos de los Tweets.
  Salidas:

  - **Salida 1:** Archivo CSV temporal que almacena los Tweets y metadatos asociados, agregando como otro campo, los Tweets recientemente preprocesados
  - **Salida 2:** Mensaje de error en caso de ingresar un archivo CSV inválido.

- **R5.-Clasificación**
  Descripción: Funcionalidad que determina la polaridad de los Tweets previamente preprocesados. Los valores de esta pueden variar dependiendo del tipo de clasificación y de la calidad del preprocesamiento.

  Entradas:

  - Enfoque de clasificación: Opción que selecciona el enfoque de clasificación a utilizar.
    - Enfoque por léxicos: De este se desprenden tres métodos posibles
      - Bolsa de palabras.
      - Método SO-PMI.
      - Ambos.
    - Enfoque por APIs aprendizaje automático: De este se desprenden tres métodos posibles.
      - API Indico.
      - API SENTIMENT140.
    - Ambos.
  - Archivo CSV que contiene Tweets preprocesados.
  - Conjunto de Tweets preprocesados obtenidos en **R4**

  Proceso: En caso que el usuario haya seleccionado una opción que sólo incluya este proceso, deberá ingresar un archivo CSV que contiene los Tweets previamente preprocesados. Caso contrario, se considerará el archivo temporal que incluye los Tweets preprocesados en **R4** para su clasificación. Una vez definidos los elementos a clasificar, el usuario deberá ingresar que enfoque utilizar, y que método para el enfoque seleccionado. Una vez seleccionado esto, el sistema realizará las tareas de clasificación.

  - **Salida 1:** Archivo CSV temporal que incluye los Tweets, junto a ello sus metadatos. Además se incluirán filas que contendrán la lista de Tweets

preprocesados y clasificados de acuerdo al método o métodos seleccionados.

- **Salida 2:** Mensaje de error en caso de algún imprevisto en el proceso.

- **R8.-Creación de archivo CSV final**
  Descripción: Funcionalidad que almacena en la carpeta del usuario un archivo CSV final que contendrá la lista de Tweets extraídos, preprocesados o clasificados. Archivo de salida que permitirá su visualización y manipulación.

  - **Entrada:** Archivo CSV temporal que contiene lista de Tweets extraídos, preprocesados y/o clasificados.

  Proceso Una vez que el sistema haya finalizado las tareas seleccionadas en **R3**. Se tomará el archivo temporal generado en **R3**, **R4** o **R5** para posteriormente crear una carpeta, que como nombre contendrá el nombre del o las palabras a buscar en **R3** (en caso de haber seleccionado extracción por palabras claves) o el nombre del usuario (En caso de haber seleccionado extracción por usuario), esto concatenado con la fecha y hora actual del sistema. Luego, se creará un archivo CSV que contendrá el mismo nombre dado en la carpeta anteriormente creada, Este fichero contendrá todos los datos almacenados en el CSV temporal. Una vez replicados, se procederá a la eliminación del archivo temporal y finalmente, se abrirá en el explorador de archivos la carpeta generada como tambien el CSV de salida.
  Salidas:

  - **Salida 1:** Archivo CSV final que contendrá los Tweets extraídos, preprocesados y/o clasificados.

  - **Salida 2:** Mensaje de error en caso de algún imprevisto en el proceso.

4.3.2. Requisitos no funcionales

- Portabilidad: El sistema deberá ejecutarse en plataformas Windows, Linux y Mac, desde un navegador web de escritorio (no en dispositivos móviles).

Esto se logrará ya que Taverna WorkFlow, es un gestor compatible con estas plataformas. Además, posee un servicio de ejecución de flujos de trabajo desde la web.

- Escalabilidad: Se requiere que el sistema eventualmente pueda ser mejorado para soportar cargas mayores de procesamiento y extracción de datos dependiendo de las necesidades de usuarios. Además, debe estar preparado para la implementación de otras funcionalidades.

Capítulo 5

Diseño

En este capítulo se mostrarán distintos modelos para representar el comportamiento del sistema. Se incluirán diagramas de actividades, modelo de datos y el modelo del sistema realizado.

5.1. Diagrama de Actividad

En el diagrama de actividad, se describirá en forma gráfica todas las actividades realizadas en el sistema, detallando de mejor manera los procesos involucrados. A partr de la siguiente página, se observan los distintos diagramas de actividades al detalle.

5.1.1. Diagrama de actividad - Creación carpeta de trabajo

En la figura 5.1 se muestran las actividades realizadas en el requisito funcional *Creación carpeta de trabajo.*

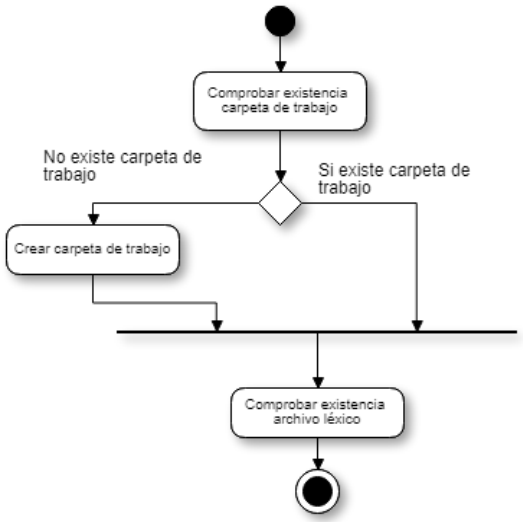


Figura 5.1: Diagrama de actividades para la funcionalidad creación carpeta de trabajo.

5.1.2. Diagrama de actividad - Cargar archivo de lexicón

En la figura 5.2 se muestran las actividades realizadas en el requisito funcional *Cargar archivo de léxicos.*

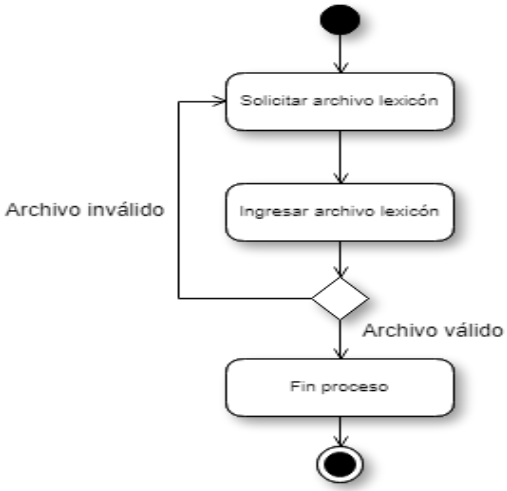


Figura 5.2: Diagrama de actividades para Carga de archivo lexicón.

5.1.3. Diagrama de actividad - Cambiar tamaño desplazamiento para manejo negaciones

En la figura 5.3 se muestran las actividades realizadas en el requisito funcional *Cambiar tamaño desplazamiento para manejo negaciones*.

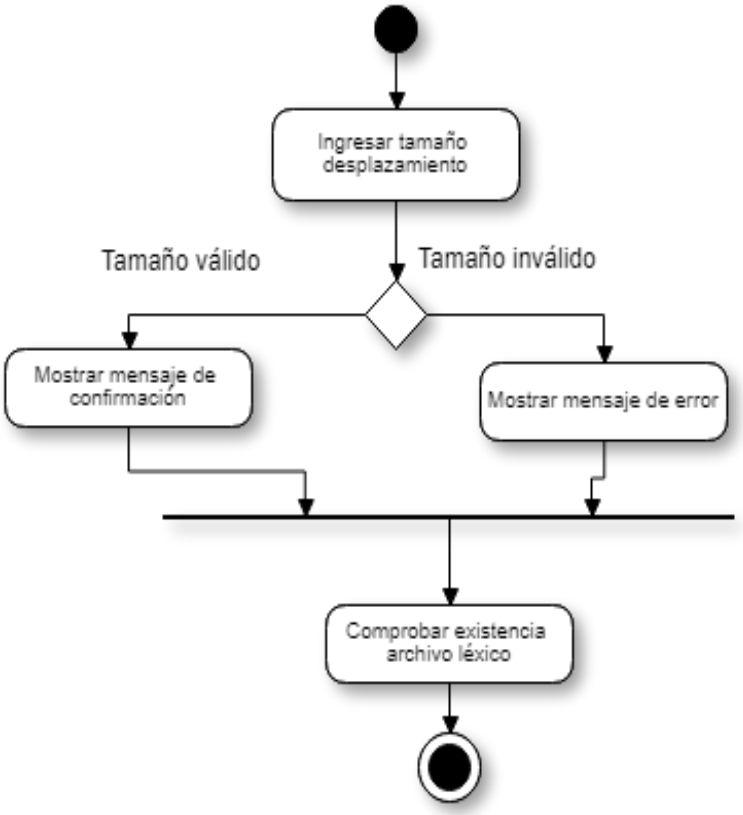


Figura 5.3: Diagrama de actividades para Cambiar tamaño desplazamiento para manejo negaciones.

5.1.4. Diagrama de actividad - Seleccionar tareas a realizar

En la figura 5.4 se muestran las actividades realizadas en el requisito funcional *Selecciónar tareas a realizar*.

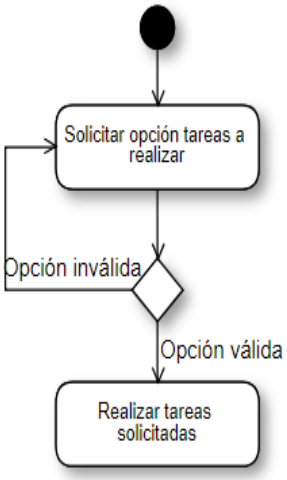


Figura 5.4: Diagrama de actividades para Selección tareas a realizar.

5.1.5. Diagrama de actividad - Extracción de Tweets

En las figuras 5.5 y 5.6 se muestran las actividades realizadas en el requisito funcional *Extracción de Tweets*.

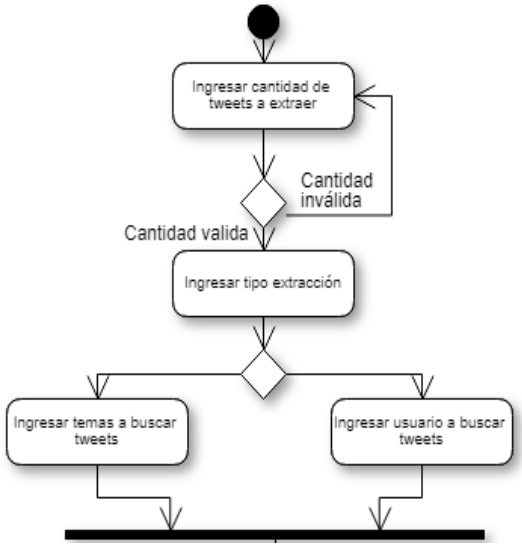


Figura 5.5: Diagrama de actividades para Extracción de tweets 1/2.

Figura 5.6: Diagrama de actividades para Extracción de tweets 2/2.

5.1.6. Diagrama de actividad - Preprocesamiento

En laS figuras 5.7 Y 5.8 se muestran las actividades realizadas en el requisito funcional *Preprocesamiento*.

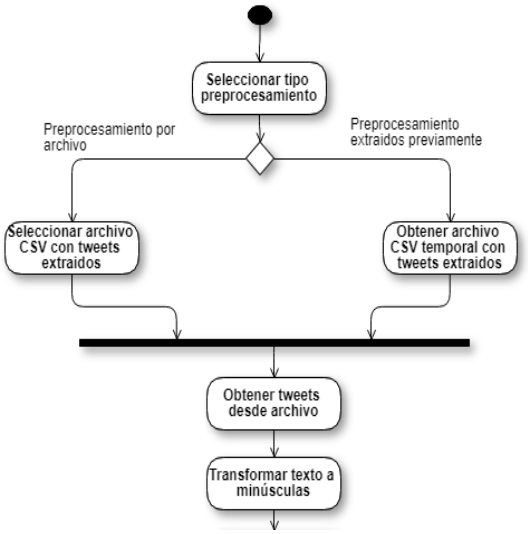


Figura 5.7: Diagrama de actividades para Preprocesamiento 1/2.

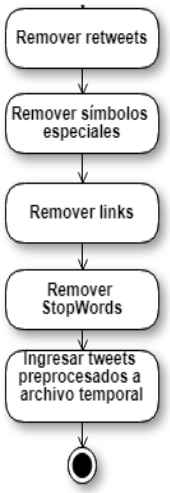


Figura 5.8: Diagrama de actividades para Preprocesamiento 2/2.

5.1.7. Diagrama de actividad - Clasificación

En figura 5.9 se muestran las actividades realizadas en el requisito funcional *Clasificación*.

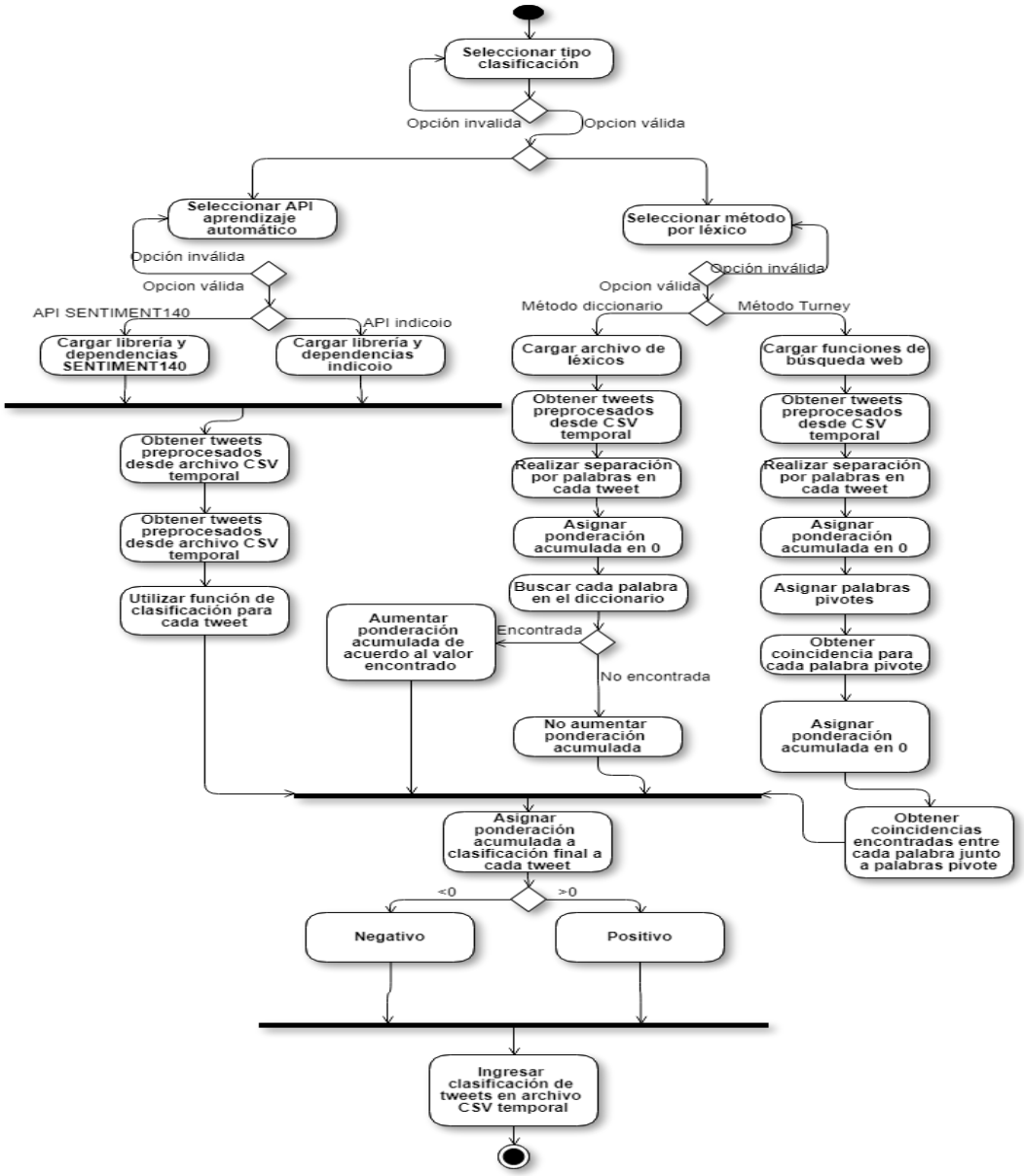


Figura 5.9: Diagrama de actividades para Clasificación.

5.1.8. Diagrama de actividad - Creación archivo CSV final

En la figura 5.10 se muestran las actividades realizadas en el requisito funcional *Creación archivo CSV final*.

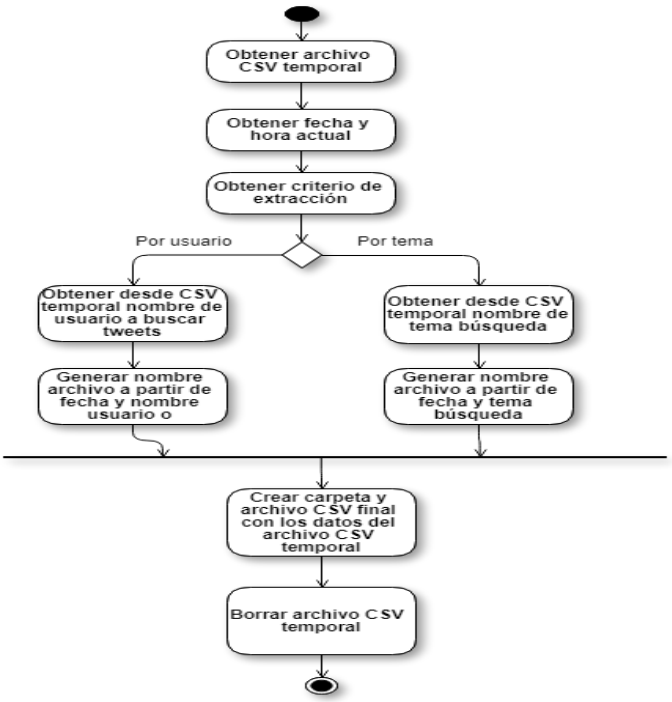


Figura 5.10: Diagrama de actividad para Creación archivo CSV final.

5.2. Modelo de datos

El modelo de datos es muy simple, ya que sólamente se requiere una tabla de configuración, la cual se encargará de almacenar el tamaño de desplazamiento y el nombre del archivo de lexicón indicado por el usuario.

CONFIG(ID, TAMANO_DESPLAZAMIENTO, ARCHIVO_LEXICON)

En la tabla 5.1 se muestra el diccionario de datos.

Campo	Tipo dato
ID	INTEGER
TAMANO_DESPLAZAMIENTO:	INTEGER
ARCHIVO_LEXICON	TEXT

Cuadro 5.1: Diccionario de datos

5.3. Modelo del Workflow desarrollado

El flujo de trabajo realizado se puede visualizar en las figuras 5.11, 5.12, 5.13,5.14, 5.15, 5.16 y A.22.

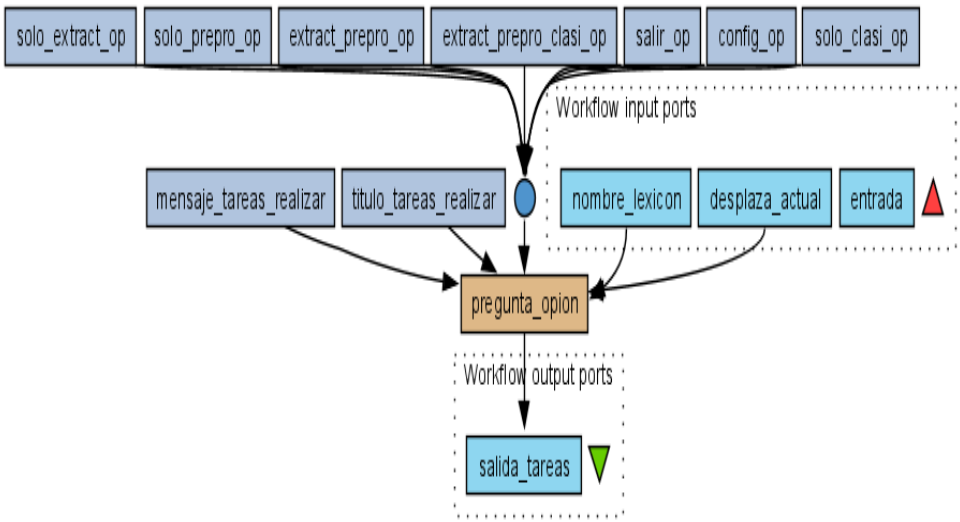


Figura 5.11: Diagrama Vista del modelo parte 1/6.

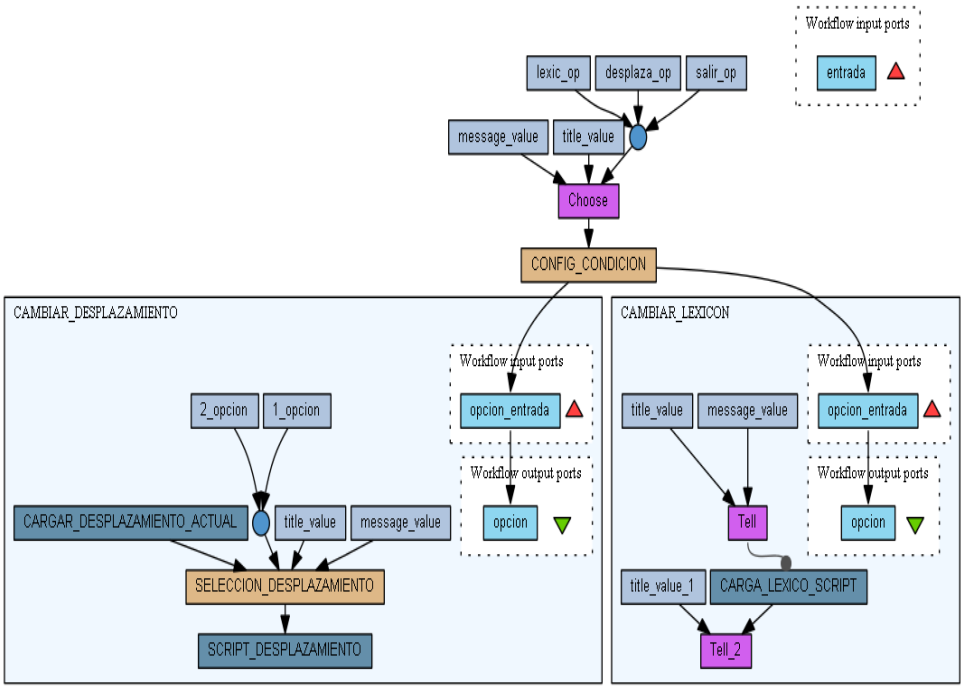


Figura 5.12: Diagrama Vista del modelo parte 2/6.

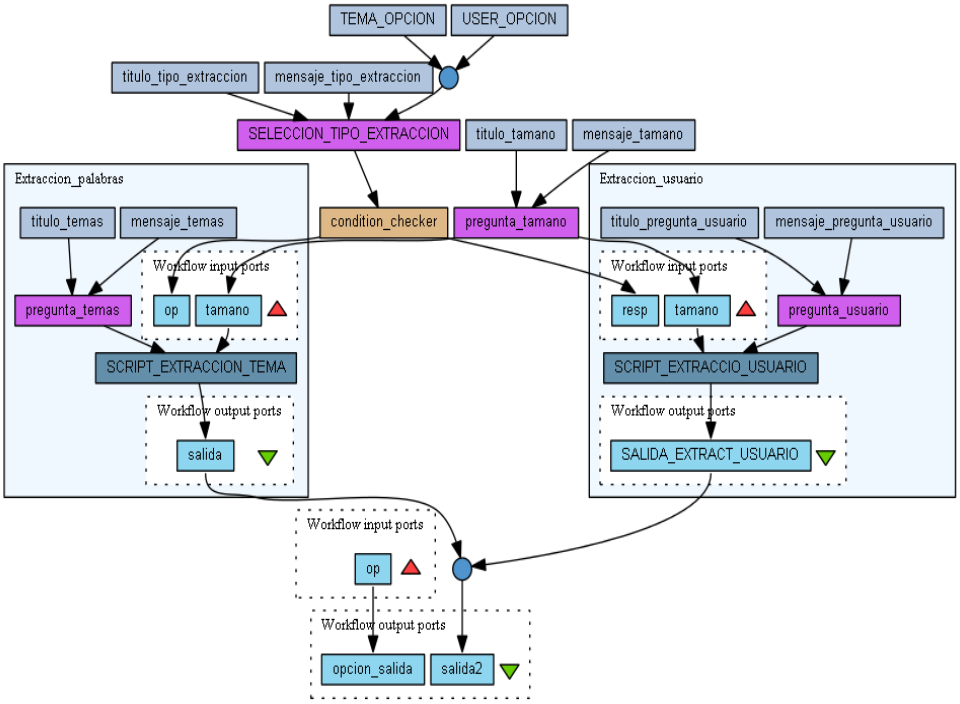


Figura 5.13: Diagrama Vista del modelo parte 3/6.

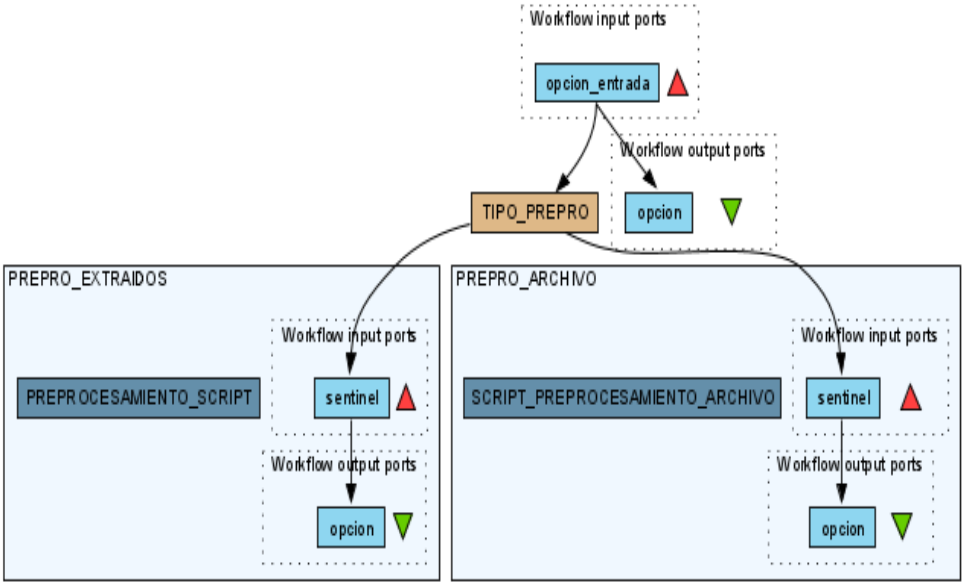


Figura 5.14: Diagrama Vista del modelo parte 4/6.

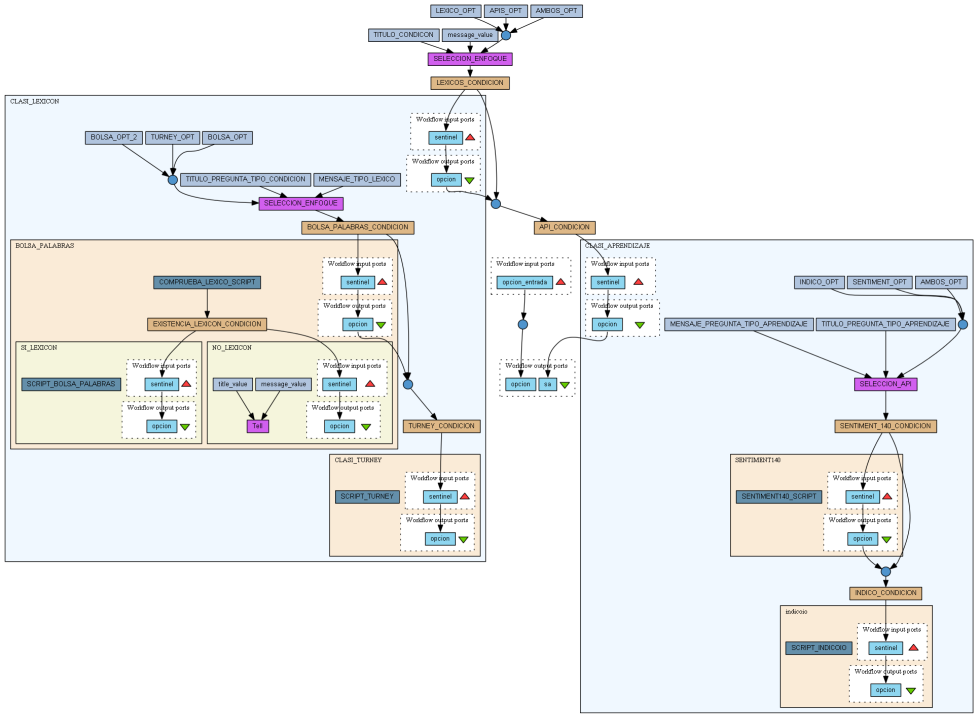


Figura 5.15: Diagrama Vista del modelo parte 5/6.

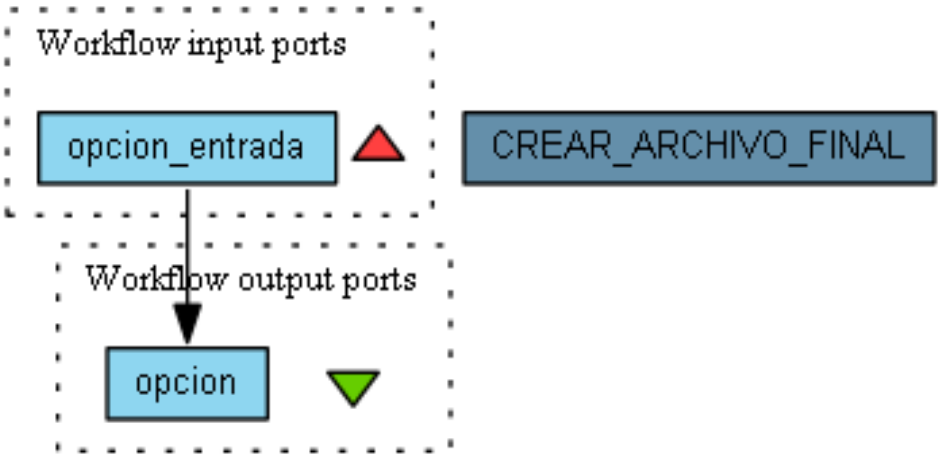


Figura 5.16: Diagrama Vista del modelo parte 6/6.

Capítulo 6

Implementación

En este capítulo se detallarán los pasos realizados en la implementación del sistema. Para ello, se describirá el cómo se compone esta aplicación en la siguiente sección.

6.1. Estructura de la aplicación

Como se vio en el modelo final del capítulo anterior. La aplicación es principalmente un flujo de trabajo (Taverna Workbench), el cual se compone de 4 módulos principales (Configuración, extracción, preprocesamiento y clasificación). Cada uno de estos elementos se considera como un flujo de trabajo contenido dentro del flujo principal. Además, estos se componen de diversos subflujos dependiendo de las acciones a realizar, dentro de estos encontraremos diversos Script de R Project, los cuales son los encargados en realizar las diversas taréas de configuración y del análisis de sentimiento.

6.1.1. Instalando paquetes necesarios

Para que el sistema inicie, se deben instalar ciertos paquetes. Para ello, se ha creado un Script de R-Project, el cual es el encargado en realizar la instalación en forma automática.

A continuación, se aprecia la codificación.

```
list.of.packages <- c("Rserve","readxl", "xlsx","RSQLite","tm",
"plyr","twitteR","data.table","stringr",
"XML","RCurl","devtools",)#lista de paquetes a installar
new.packages <- list.of.packages[!(list.of.packages %in%
    installed.packages()[,"Package"])] #se comprueba que no este
    instalado, y se agrega a un array ese paquete
if(length(new.packages)) install.packages(new.packages) #se instalan los
    paquetes perteneciente a la lista de paquetes no instalados

require(devtools)
install_github('sentiment140', 'okugami79') #instala sentiment140
devtools::install_github("IndicoDataSolutions/IndicoIo-R") #instala indico
```

Este Script permite la instalación de todos los paquetes necesarios para ejecutar la aplicación. El código comprueba que cada paquete de *list.of.packages* no esté instalado, luego se ingresa a una lista llamada *new.packages* que contiene el nombre de los paquetes por instalar. Finalmente, se instala cada paquete perteneciente a esta lista.

6.2. Antes de iniciar la aplicación

Para que la aplicación pueda funcionar, es importante correr un servidor de socket en forma local (a menos que se tenga un servidor web corriendo). Ello se facilita mediante el paquete Rserve, el cual se encarga de crear un espacio y directorio de trabajo para que distintas aplicaciones clientes puedan realizar solicitudes al servicio levantado por Rserve.

6.2.1. Iniciando el servidor

Observar el siguiente código:

```
library(Rserve)
Rserve()
```

El Script mostrado en el código anterior, es el encargado en iniciar el servidor de socket. Una vez iniciado, quedará a la escucha de solicitudes por parte de diversos clientes. En este caso, el cliente será Taverna.

6.3. Creación de la carpeta de trabajo y configuración inicial

Una vez iniciada la aplicación. Se ralizará la configuración inicial en donde se creará la carpeta principal de trabajo (en caso de no encontrarse), creación de tabla config (si no existe) y obtención de datos asociados a ella.

A continuación, se observa la codificación:

```
1  library(DBI)
2  library(RSQLite) #carga librerias
3
4  u<-path.expand('~') #obtener ruta usuario
5  dir.create(file.path(u, "sentimets_workflow"), showWarnings = FALSE)
       #Crear ruta de trabajo
6  setwd(file.path(u, "sentimets_workflow"))
7
8  u<-gsub("[/]","\\\\",u)
9  ruta_principal<-paste(u,"\\sentimets_workflow\\" ,sep="")
10
11 mydb <- dbConnect(RSQLite::SQLite(), "config.sqlite") #conectar a bd o
       crear en caso de no existir
12 query<-dbGetQuery(mydb, "CREATE TABLE IF NOT EXISTS CONFIG (ID INTEGER
       PRIMARY KEY, DESPLAZAMIENTO INTEGER, RUTA_LEXICO TEXT)") #crear tabla
       config si no existe
13 query_cont<-dbGetQuery(mydb, 'SELECT COUNT(*) as conta FROM CONFIG WHERE
       ID=1') #si no hay registro ingresar uno vacio
14 contador<-query_cont$conta
15
16 if(contador==0){
17   query_insert<-dbGetQuery(mydb, 'INSERT INTO CONFIG VALUES(1,0,"")') #se
```

```
    ingresa el registo vacio
}

query_desplaza<-dbGetQuery(mydb, 'SELECT * FROM CONFIG WHERE ID=1') #se
    obtiene el tamano desplazamiento y nombre archivo lexico
desplaza<-query_desplaza$DESPLAZAMIENTO
lexico<-query_desplaza$RUTA_LEXICO
unlink("config.sqlite") #$desconexion sqlite
```

Primero se llama al script para generar la carpeta de trabajo, el cual es muy corto y facil de explicar. Básicamente, se obtiene la ruta del usuario, un ejemplo de Windows es C:\Usuarios\NombreUsuario y luego, dentro de ella se crea la carpeta sentiment_workflow. Una vez creada la ruta de trabajo, se comprueba la existencia del archivo de configuración. En caso de ser la primera vez que se ejecuta la aplicación, es evidente que el archivo no existirá, por lo cual se creará este archivo de configuración, el cual consiste en una base de datos SQLITE de una tabla, la que posee el tamaño de desplazamiento para el manejo de negaciones y el nombre del archivo utilizado como lexicón.

6.4. Menú de opciones

Recordar que el menú de opciones quedo representado por el modelo de la figura 6.1:

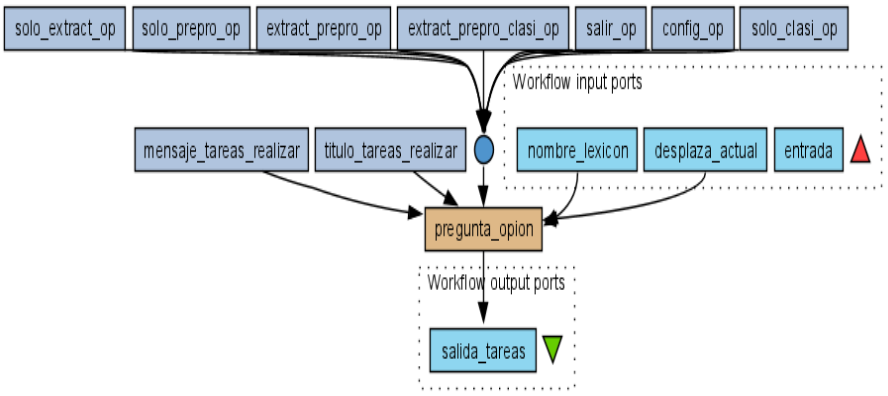


Figura 6.1: Modelo representativo Tareas a realizar

Al iniciar la aplicación, se mostrará la pantalla de la figura 6.2.

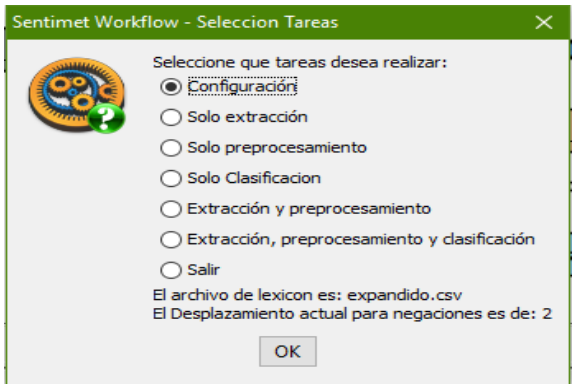


Figura 6.2: Pantalla tareas a realizar

Un poco de BeanShell

Antes de continuar, es necesario mencionar que Taverna utiliza este lenguaje de script similar a Java para realizar ciertas funcionalides. Una de estas, es la estructura condicional. Cada vez que se vaya a evaluar una condición, se debe crear un script que permita saber que acciones realizar. A continuación, se observa un ejemplo básico de condicional:

```
1  import java.util.ArrayList;
2
3  ArrayList pass_flags = new ArrayList();
4  ArrayList fail_flags = new ArrayList();
5
6
7  if (condition.equals("Solo extracci'on") || condition.equals("Solo
       preprocesamiento") || condition.equals("Extracci'on y
       preprocesamiento") || condition.equals("Extraccion, preprocesamiento y
       clasificacion") )
8  {
9   pass_flags.add(condition);
10 }
11 else
12 {
```

```
13   fail_flags.add(condition);
14 }
```

Este script evalúa si se debe realizar alguna tarea. Para ello, se recibe como parámetro de entrada el valor de la opción que se ha seleccionado. En este caso, se evalua que la opción seleccionada sea distinta a salir. En caso de ser verdadero, se almacena en un vector llamado pass flags el valor ingresado por entrada. Caso contrario, se ingresa este valor en el vector fail flags.

Finalmente, enlazar a que workflow se entrará en caso de ser verdera la condición (el vector pass flags) y a que workflow se entrará en caso de ser falsa la condición (vector fail flags)

6.5. Módulo de configuración

El modelo a visualizar es el presentado en la figura 6.3 :

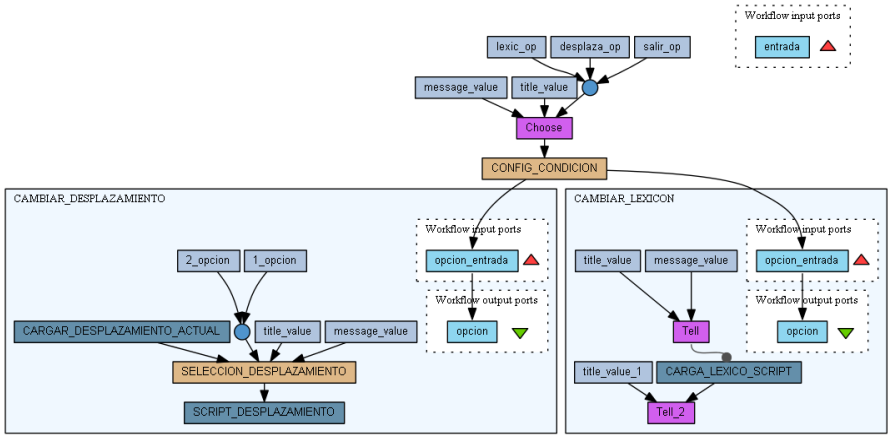


Figura 6.3: Modelo Módulo de configuración

En este se realizarán las configuraciones pertinentes para que el sistema funcione correctamente.

6.5.1. Selección archivo lexicón

Se solicita al usuario el ingreso de un archivo CSV que contiene el diccionario de palabras y polaridades. El uso de este es imprescindible si se desea utilizar clasifica-

ción mediante bolsa de palabras.

A continuación, se observa la codificación de esta funcionalidad:

```
library(DBI)
library(RSQLite) #carga librerias
u<-path.expand('~')
setwd(file.path(u,"sentimets_workflow"))
u<-gsub("[/]","\\\\",u)
ruta_principal<-paste(u,"\\sentimets_workflow\\" ,sep="")
repeat{
  result = tryCatch({
    ruta_excel<-file.choose()
    name <- basename(ruta_excel)
    #print(ruta_excel)
    ex <- read.csv(ruta_excel, sep=";", encoding = "UTF-8")
    ex$lexicon
    ex$Pal
    ex$polaridad
    se comprueba que existan todos los campos, si hay un error se lanza
       una excepcion
  },error = function(error_condition) {
    print("error de apertura")
    return (-1) #si hay una excepcion retornamos un -1
  })

  if(result!=-1){ #Si no hubo una excepcion quiere decir que el archivo
     es el correcto$
    file.copy(ruta_excel,ruta_principal)
    from1 <- c(name)
    to1 <- c("lexics.csv")
    file.rename(from1, to1)
      mydb <- dbConnect(RSQLite::SQLite(), "config.sqlite") #conectamos a
         la bd

   query_update<-dbGetQuery(mydb, 'UPDATE CONFIG SET RUTA_LEXICO= :x
```

```
      WHERE ID=1' ,
30          params = list(x = name)) #actualizamos el nombre del archivo
                lexicon para mostrar
31    unlink("config.sqlite") #desconexcion a bd
32    salida<-"Carga exitosa del archivo"
33    break
34  }
35 }
```

En el código anterior se aprecia la utilización de un ciclo repetitivo con la finalidad de volver a solicitar el archivo en caso de una lectura inválida. Esto se da en el caso que el archivo ingresado no corresponda al de lexicón. Para ello, se comprueba que los campos pertenecientes al archivo ingresado, correspondan al del archivo lexicón. En caso de correspondencia, se copia el archivo a la ruta principal de trabajo. Caso contrario, se lanza una excepción y se vuelve a repetir el proceso anterior.

Nota: Al lanzar la excepción, el tryCatch retorna un -1. Esto es a fin de cumplir la condición de continuidad para el ciclo repetitivo (el ciclo se detiene para un valor distinto a -1). Este se detiene en el caso de no entrar a la función error del tryCatch.

6.5.2. Cambiar tamaño desplazamiento para manejo de negaciones

Esta funcionalidad permite ajustar cuantas palabras como máximo se debe desplazar para encontrar una palabra positiva en caso de hallar una negación.

A continuación, se observa la codificación:

```
1 library(RSQLite) #se cargan librerias
2 u<-path.expand('~')
3 setwd(file.path(u, "sentimets_workflow"))
4 u<-gsub("[/]","\\\\",u)
5 ruta_principal<-paste(u,"\\sentimets_workflow\\" ,sep="") # se setea ruta
    de trabajo
6
7 mydb <- dbConnect(RSQLite::SQLite(), "config.sqlite") #abrimos la
```

```
    conexi'on a la bd
desplazabd<-query_select<-dbGetQuery(mydb, 'SELECT DESPLAZAMIENTO FROM
    CONFIG') #seleccionamos de la tabla config el desplazamiento actual
salida<-desplazabd$DESPLAZAMIENTO #$retornamos el desplazamiento actual
unlink("config.sqlite")
```

```
library(RSQLite)
u<-path.expand('~')
setwd(file.path(u, "sentimets_workflow"))
u<-gsub("[/]","\\\\",u)
ruta_principal<-paste(u,"\\sentimets_workflow\\" ,sep="") #seteamos ruta
    principal

#conexion a bd
mydb <- dbConnect(RSQLite::SQLite(), "config.sqlite") #abrimos conexion a
    la bd

query_update<-dbGetQuery(mydb, 'UPDATE CONFIG SET DESPLAZAMIENTO= :x
    WHERE ID=1' ,
           params = list(x = n)) # actualizamos el desplazamiento anterior
               por el actual
#desconexion a bd
unlink("config.sqlite")
salida<-1
```

Primero, se obtiene desde la tabla *config* el desplazamiento actual. Esto se realiza con la finalidad de informar al usuario el desplazamiento seteado antes de cambiar. Finalmente, se debe seleccionar el nuevo tamaño, actualizándolo en la tabla de configuración.

6.6. Primera etapa: Extracción

Primero, recordar el modelo para esta etapa en la figura 6.4:

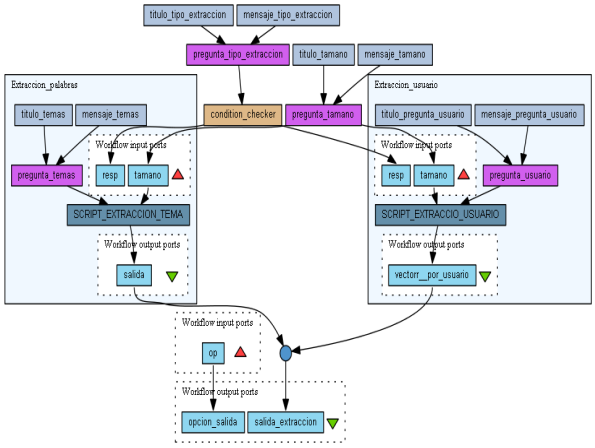


Figura 6.4: Modelo Extracción

Si el usuario ha seleccionado una opción que incluya la realización de la tarea extracción de Tweets. Se entrará un SubWorkflow llamado extracción. Dentro de este se solicitará la cantidad de Tweets a extraer y el tipo de extracción (figura 6.5).

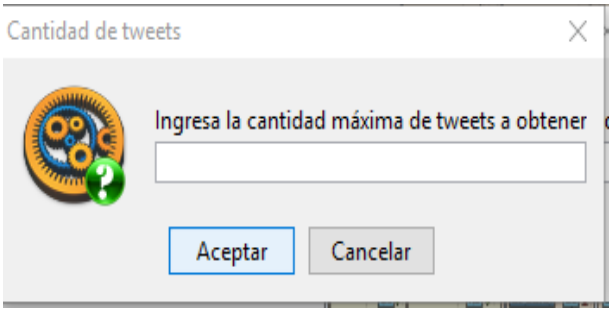


Figura 6.5: Solicitando cantidad de Tweets

Al seleccionar el tipo de extracción, nuevamente se llamará a un BeanShell el cual evaluará que tipo de extracción se va a realizar (figura 6.6).

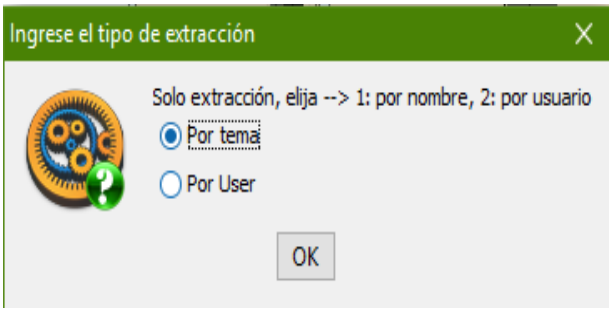


Figura 6.6: Solicitando tipo de extracción de Tweets

Si se selecciona extracción por palabras claves, se desplegará una pantalla en donde se ingresa la o las palabras que debe contener los Tweets a buscar. En caso de seleccionar extracción por usuario, se va a solicitar el ingreso del nombre de usuario a buscar Tweets (figuras 6.7 y 6.8).

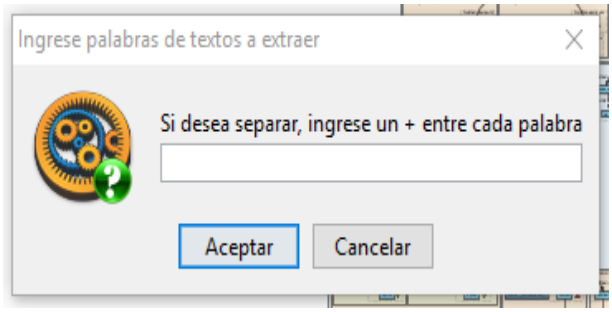


Figura 6.7: Solicitando palabras claves para extraer Tweets

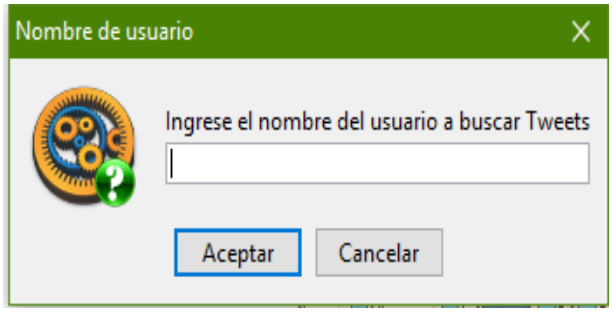


Figura 6.8: Solicitando Username para extraer Tweets

6.6.1. Como se extraen los Tweets?

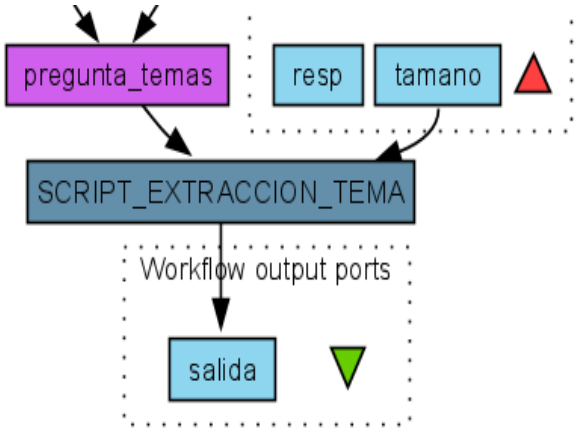


Figura 6.9: Solicitando Username para extraer Tweets

Como se visualiza en la figura 6.9, en el Workflow se ha implementado un Script llamado EXTRACCION, es importante considerar que la extracción se realizará por palabras claves o por usuario. En este caso se muestra la implementación por palabras claves, para la extracción por usuario, el modelo es exactamente igual.

Si observamos al detalle, podemos notar que el script recibe dos parámetros de entrada, los cuales son:

- Tamaño: Cantidad de Tweets a extraer.
- Palabras claves: La o las palabras que deben contener los Tweets a buscar.

Nota: En caso de seleccionar extracción por usuario, el segundo parámetro corresponde al Username

A continuación, se aprecia la codificación:

```
1 library(plyr)
2 library(twitteR) #se cargan librerias
3 consumer_key<-"***"
4 consumer_secret<-"***"
5 access_token<-"***"
6 access_secret<-"***"
7 setup_twitter_oauth(consumer_key, consumer_secret, access_token,
    access_secret) #autentifica en api twitter
```

```
8  Tweets<-searchTwitter("cadena_de_bsqueda",lang = "es",n=10) # funcion que
       busca Tweets
9  daf = twListToDF(Tweets) #transformar a dataframe
10 daf$palabra=p #guardar el criterio de b?squeda$
11 df = as.matrix(daf)
12 df_Place2 = data.frame(lapply(daf, as.character), stringsAsFactors=FALSE)
       #estas dos funciones permite guardar un excel
13 ruta_principal<-paste(u,"\\sentimets_workflow\\" ,sep="") #concatenar
       ruta de usuario con nombre carpeta trabajo
14 ruta_principal<-paste(ruta_principal,"temp.csv" ,sep="") #generar ruta
       absoluta archivo temporal
15 write.table(df_Place2, file=ruta_principal,row.names=FALSE,sep=";")
       #guardar CSV temporal
```

Para realizar la tarea de extracción, se debe realizar la autentificación en el API de Twitter mediante una llave de acceso provista por la misma plataforma. Para ello se utiliza la función *setup_twitter_oauth*. Una vez realizado este paso, se realiza un llamado a la función *searchTwitter*, la cual recibe los parámetros de entrada descritos anteriormente.

Una vez extraídos los Tweets, se guardarán en un dataframe:

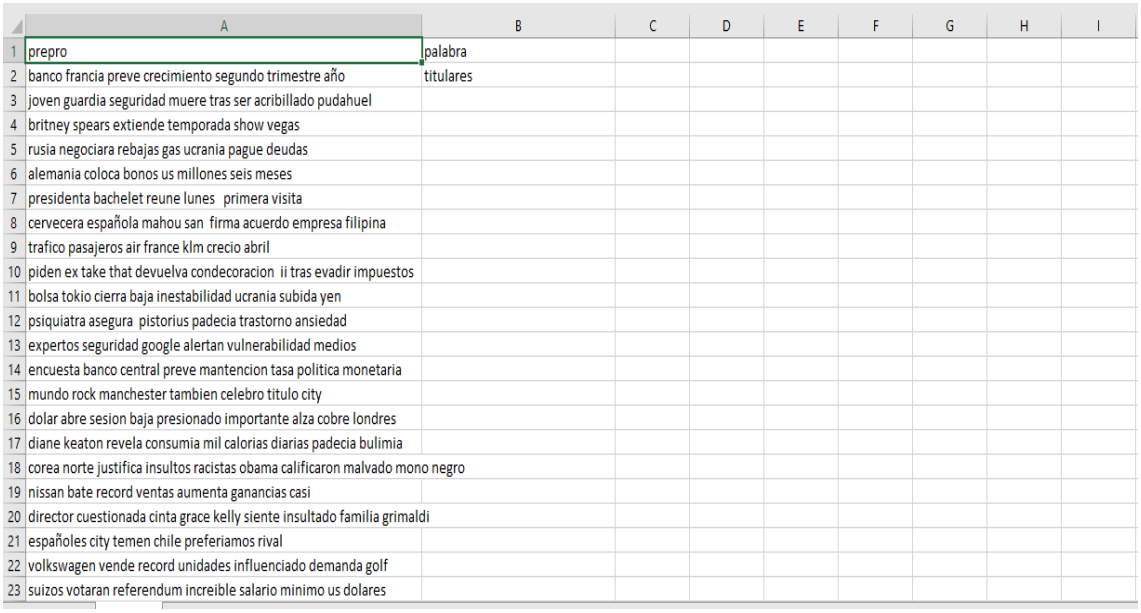
dataframe en R

Un daframe es una estructura de datos, la cual comparte muchas de las propiedades de las matrices. Permite almacenar datos en cuadrículas rectangulares. Cada fila representa un atributo, mientras que cada columna es una lista que contiene datos para ese atributo. En la figura 6.10 se aprecia un ejemplo de dataframe :

	prepro	palabra
1	banco francia preve crecimiento segundo trimestre año	titulares
2	joven guardia seguridad muere tras ser acribillado pudahuel	

Figura 6.10: Dataframe que contiene titulares de noticias

Finalmente, se guarda el dataframe en un archivo CSV temporal, almacenando como atributo los temas o usuario a buscar. O, en caso de haber seleccionado una opción que sólo incluya este proceso, simplemente se creará un archivo CSV final. En la figura 6.11 se observa la estructura de un archivo CSV con el dataframe obtenido.



	A	B	C	D	E	F	G	H	I
1	prepro	palabra							
2	banco francia preve crecimiento segundo trimestre año	titulares							
3	joven guardia seguridad muere tras ser acribillado pudahuel								
4	britney spears extiende temporada show vegas								
5	rusia negociara rebajas gas ucrania pague deudas								
6	alemania coloca bonos us millones seis meses								
7	presidenta bachelet reune lunes primera visita								
8	cervecera española mahou san firma acuerdo empresa filipina								
9	trafico pasajeros air france klm crecio abril								
10	piden ex take that devuelva condecoracion ii tras evadir impuestos								
11	bolsa tokio cierra baja inestabilidad ucrania subida yen								
12	psiquiatra asegura pistorius padecia trastorno ansiedad								
13	expertos seguridad google alertan vulnerabilidad medios								
14	encuesta banco central preve mantencion tasa politica monetaria								
15	mundo rock manchester tambien celebro titulo city								
16	dolar abre sesion baja presionado importante alza cobre londres								
17	diane keaton revela consumia mil calorias diarias padecia bulimia								
18	corea norte justifica insultos racistas obama calificaron malvado mono negro								
19	nissan bate record ventas aumenta ganancias casi								
20	director cuestionada cinta grace kelly siente insultado familia grimaldi								
21	españoles city temen chile preferiamos rival								
22	volkswagen vende record unidades influenciado demanda golf								
23	suizos votaran referendum increible salario minimo us dolares								

Figura 6.11: Estructura del archivo CSV que contiene titulares de noticias

6.7. Segunda etapa: Preprocesamiento

Esta tarea se desencadena tal como la anterior. El usuario debe haber seleccionado una opción que incluya la realización del preprocesamiento. Este SubWorkflow, tal como el anterior, hay dos posee contenidos dentro del principal, tal como se aprecia en la figura 6.12:

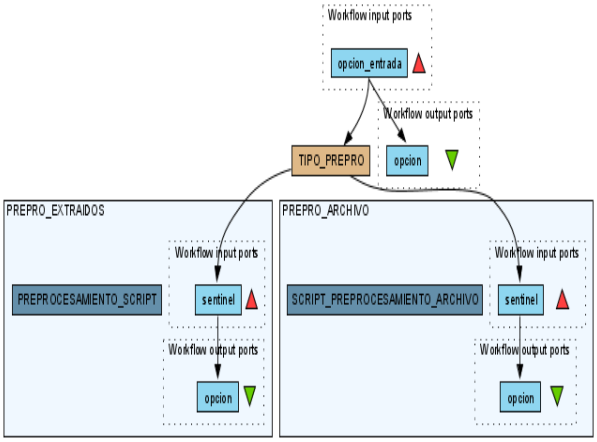


Figura 6.12: Estructura del archivo CSV que contiene titulares de noticias

El primer SubWorkflow ocurre cuando el usuario ha seleccionado una opción que previamente haya incluido el proceso de extracción. En este caso, una vez finalizada la etapa anterior, automáticamente se realiza el preprocesamiento (a través del archivo CSV temporal). Caso contrario, el sistema solicitará un archivo CSV, el cual debe contener la lista de Tweets a preprocesar (figura 6.13).

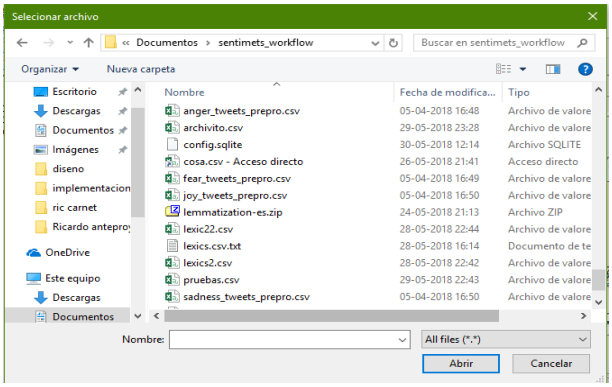


Figura 6.13: Seleccionando archivo CSV para preprocesar

A continuación, se observa la codificación:

```
1
2  x="Texto a preprocesar"
3  x=sapply(x, tolower)
4  require("tm")
5  x<-removeWords(x,stopwords("es"))
6  x = gsub("(rt|via)((?:\\b\\W*@\\w+)+)", "", x) #remueve reTweets
7  x = gsub("@\\w+", "", x) #remueve el @
8  x<- gsub("http[^[:space:]]*", "", x) # remueve links
9  x = gsub("<", "", x) #remueve simbolos especiales
10 x = gsub(">", "", x)
11 x = gsub("[[:punct:]]", " ", x) # remueve simbolos de puntuacin
12 x = gsub("[[:digit:]]", " ", x) # remueve nmeros
13 x = gsub("[ |\t]{2,}", " ", x) # remueve tabs
14 x = gsub("^ ", "", x) # remueve espacio en blanco al inicio
15 x = gsub("[ |\n]{2,}", " ", x) # remueve todo \n
```

El preprocesamiento consiste en obtener el atributo texto para cada Tweet. Luego, a cada String se le compara con ciertos patrones que se deben eliminar, como por ejemplo links, números, caracteres especiales, etc. Además, se eliminan los StopWords correspondientes al lenguaje español. Una vez limpios los textos, se gurdarán como un atributo llamado prepro.

Finalmente, se guardarán los datos en el archivo CSV temporal, o en caso de haber seleccionado una opción que incluya como paso final el preprocesamiento, se creará un archivo CSV final.

6.8. Tercera etapa: Clasificación

La realización de esta tarea depende de las mismas condiciones que las etapas anteriores. Si el usuario ha seleccionado una opción que la incluya, esta se va a desencadenar.

A modo de recordar, el modelo se aprecia en la figura 6.14:

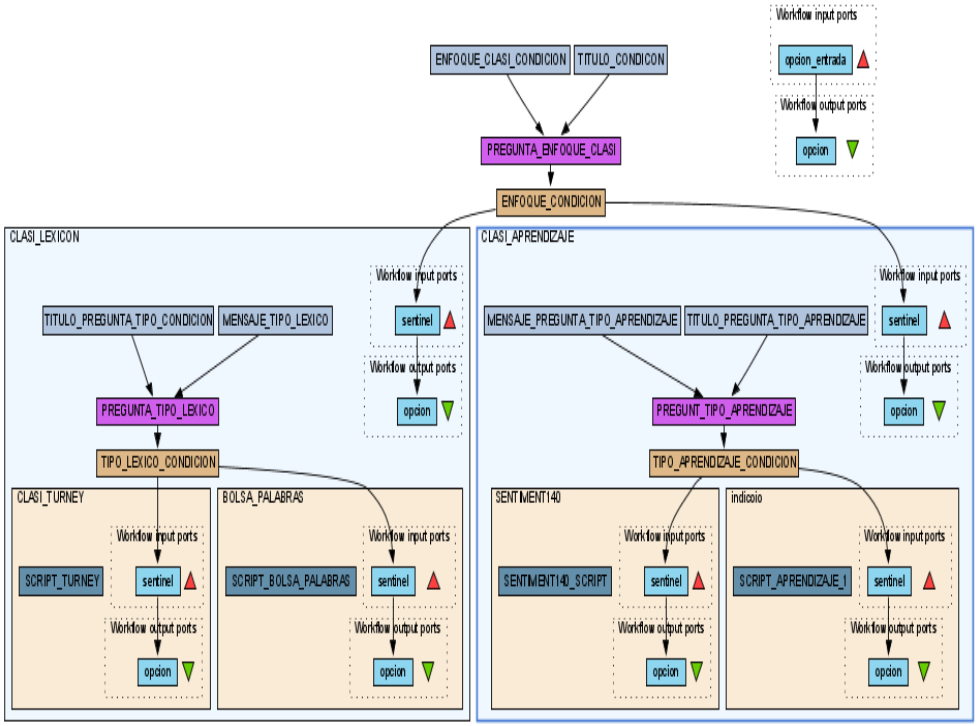


Figura 6.14: Modelo Clasificación

Este WorkFlow se compone de dos SubWorkflows principales, cada uno representa un enfoque de clasificación. Al llegar a esta etapa, se le preguntará al usuario que enfoque utilizar.

6.8.1. SubWorkflow Léxico

La figura 6.15 representa el enfoque de clasificación mediante lexicones. De este enfoque se descomponen dos métodos de clasificacón, cada uno representado por su SubWorkflow correspondiente.

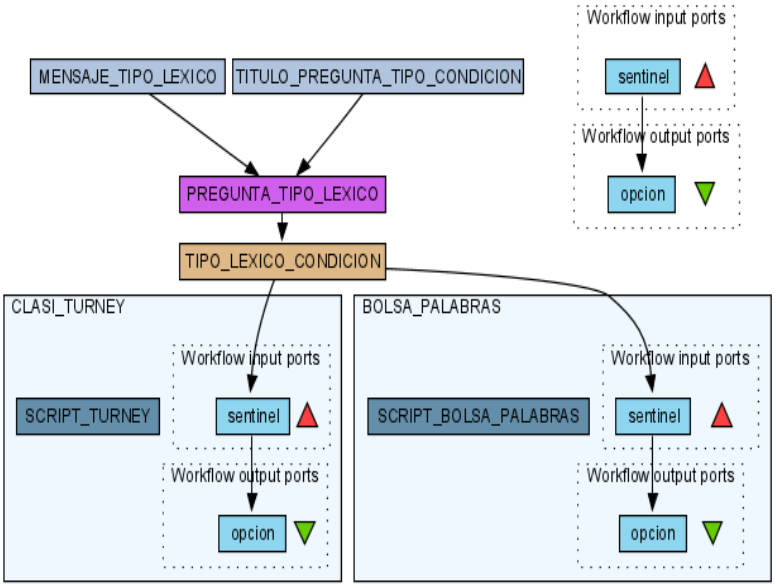


Figura 6.15: Modelo Clasificación por lexicón.

Nota: Se le preguntará al usuario que método/s de clasificación utilizar (figura 6.16).

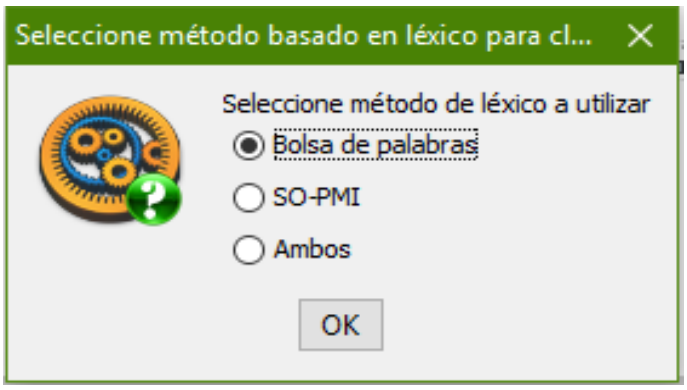


Figura 6.16: Seleccionando método de clasificación.

Método bolsa de palabras

Este método se basa en el uso de un diccionario, el cual es un archivo CSV que posee una lista de palabras, cada una enlazada a un valor de polaridad (positivo - negativo - neutro).

La estructura del archivo es la ilustrada en la figura 6.17:

	A	B	C	D	E	F	G	H	I	J	Q
1	lexicon	Pal	EX_1	EX_2	EX_3	EX_4	EX_5	EX_6	EX_7	EX_8	polaridad
2	ch	amasado									positivo
3	ch	automotor									neutro
4	ch	blanquillo									positivo
5	ch	boxeril									positivo
6	ch	breva									neutro
7	ch	canastero									neutro
8	ch	cancelación									negativo
9	ch	catita									neutro
10	ch	caturra									neutro
11	ch	cebado									positivo
12	ch	cebollín									neutro
13	ch	cestero									neutro
14	ch	dengue									negativo
15	ch	hípico									neutro
16	ch	kinésica									neutro
17	ch	laucho									neutro
18	ch	lengüeta									neutro
19	ch	lomaje									neutro
20	ch	maitén									positivo
21	ch	malla									neutro
22	ch	malva									positivo
23	ch	manejador									neutro
24	ch	martillero									neutro
25	ch	masticable									neutro

Figura 6.17: Diccionario de palabras, cada una con su polaridad asignada. Cabe destacar que este diccionario está adaptado para el uso de palabras chilenas.

En caso de no encontrarse el diccionario, el sistema arrojará un mensaje indicando que no se encuentra dicho archivo, omitiendo la clasificación mediante bolsa de palabras.

Observar el siguiente código:

```
u<-path.expand('~')
u<-gsub("[/]","\\\\",u)
ruta_principal<-paste(u,"\\sentimets_workflow\\" ,sep="")
lexi_ruta<-paste(ruta_principal,"lexic.csv" ,sep="")
input.good<-lexi_ruta
existe=file_test("-f", input.good) #comprobamos si existe el archivo
    lexico

#si no existe se retorna un 0
if (existe==FALSE) {
  salida<-0
}else{
  salida<-1
}
```

Se comprueba la existencia del diccionario de lexicón. En caso de existir se re-

torna un 1 permitiendo la ejecución del clasificador. Caso contrario, se retorna un 0, mostrando el mensaje de error, para posteriormente continuar la ejecución de los demas clasificadores.

Una vez detectado el diccionario de lexicón, se puede realizar el proceso de clasificación. Eso si, previamente, se deben setear ciertos parámetros en el diccionario para realizar la ejecución del clasificador.

A continuación, se muestra la codificación, la cual se divide en 6 partes debido a lo extensa que es. Cada parte será descrita a fin de comprender el funcionamiento del clasificador:

```
1
2  library(data.table)
3  library(corpus)
4  library(stringr)
5  library(dplyr)
6  library(stringi)
7  library(DBI)
8  library(RSQLite) #cargamos librerias
9
10 u<-path.expand('~')
11 setwd(file.path(u,"sentimets_workflow"))
12 u<-gsub("[/]","\\\\",u)
13 ruta_principal<-paste(u,"\\sentimets_workflow\\",sep="")
14 archivo_lexico<-paste(ruta_principal,"lexics.csv",sep="") #obtenemos
       archivo de lexicon
15 ex <- read.csv(archivo_lexico, sep=";", encoding = "UTF-8")
16
17 mydb <- dbConnect(RSQLite::SQLite(), "config.sqlite")
18 desplazabd<-query_select<-dbGetQuery(mydb, 'SELECT DESPLAZAMIENTO FROM
       CONFIG')
19 desplazabd<-desplazabd$DESPLAZAMIENTO
20 unlink("config.sqlite") #$obtenemos el tamano de desplazamiento actual
21 ex$EX_1_individual=""
22 ex$EX_2_individual=""
23 ex$EX_3_individual=""
```

```
ex$EX_4_individual=""
ex$EX_5_individual=""
ex$EX_6_individual=""
ex$EX_7_individual=""
ex$EX_8_individual=""
ex$EX_9_individual=""
ex$EX_10_individual=""
ex$EX_11_individual=""
ex$EX_12_individual=""
ex$EX_13_individual=""
ex$Pal_es=""
ex$EX_1_es=""
ex$EX_2_es=""
ex$EX_3_es=""
ex$EX_4_es=""
ex$EX_5_es=""
ex$EX_6_es=""
ex$EX_7_es=""
ex$EX_8_es=""
ex$EX_9_es=""
ex$EX_10_es=""
ex$EX_11_es=""
ex$EX_12_es=""
ex$EX_13_es="" #$

}
```

Primero, se debe obtener el archivo que contiene el lexicón y añadir ciertos campos adicionales con la finalidad de mejorar el desempeño del clasificador.

A continuación, se observa la segunda parte de la codificación:

```
posi=1
for(x in ex$EX_1){ #recorremos la primera expansion
  if(!is.na(x) && x != ''){
    p1<-(strsplit(x, "[[:punct:] ]")[[1]]) #si no esta vacio el campo,
```

```
        extramos la/las palabras que posee
    if(length(p1)==1){ #en caso de poseer una palabra, significa que es un
        sinonimo de la palabra
      ex$EX_1_individual[posi]=p1 #almacenamos en la posicion i-esima
    }
    else{
      ex$EX_1_individual[posi]=""
    }

  }
  posi=posi+1 #$
}
```

Como se aprecia en el código anterior, se realiza un recorrido a través de los 13 posibles campos de expansión para cada palabra. Cada campo, se separa en términos, procurando guardar aquellos que sólo posean un elemento, ya que esto significa que la expansión actual, es un sinónimo de la palabra clave. Por lo cual, se debe guardar y considerar para las búsquedas

Ahora, se muestra la tercera parte de la codificación que corresponde al stemizado de palabras:

```
posi=1
for(x in ex$Pal){
  if(ex$X.U.FEFF.lexicon[posi]!="ch"){ #stemizar palabras no chilenismos
    ex$Pal_es[posi]=text_tokens(x, stemmer = "es")
  }
  posi=posi+1

   posi=1
for(x in ex$EX_1_individual){
  if(ex$X.U.FEFF.lexicon[posi]!="ch" && x!=""){ #comprobando cada campo
      de expansion individual

    ex$EX_1_es[posi]=text_tokens(x, stemmer = "es") # english stemmer
  }
```

```
  else{
    ex$EX_1_es[posi]=""
  }

  posi=posi+1
}#$
}
```

El stemizado permite obtener la raíz de la palabra, mejorando el diccionario ya que permite obtener una mayor cantida de aciertos en las búsquedas. Este proceso es sencillo, s'lo se cosiderará cada palabra clave no sea un chilenismo. Este proceso se realiza también para los sinónimos de cada palabra clave.

A continuación, se observa la cuarta parte de la codificación:

```
archivo_ruta<-paste(ruta_principal,"temp.csv" ,sep="")
e<-read.table(file =archivo_ruta, header=TRUE,sep=";",fill = TRUE )
textos<-e$prepro #obtenemos archivo csv temporal

cont<-1
p<-list()
pos_lexico<-1
pol<-0 #inicializamos variables
for (i in textos){ #recorrido por cada texto
  print(pos_lexico)
  pola<-comprobar_polaridad(i,desplazabd)
  p[pos_lexico]<-pola
  pos_lexico=pos_lexico+1
   #$

}
```

Como se aprecia en el código anterior, para cada texto se le aplica una función llamada comprobar_polaridad, la cual es la encargada en determinar la polaridad de cada Tweet.

En el siguiente código, se observa sólo las partes mas importantes, ya que su

implementación es demasiado extensa:

```
1
2  p<-(strsplit(text, "[[:punct:] ]")[[1]]) #separar en palabras el texto
3   coincidencia=-1 #bandera que indica palabra encontrada en diccionario
4   for (i in p){ #recorrido por palabras
5     if(i!=""){ #se ignoran campos vacios
6
7       if((i=="no" || i=="nunca" ) && negacion==0 &&
            limite_desplazamiento!=0){ #en caso de encontrar una negacion
            sin estar desplazando previo a una negacion anterior
8         negacion=1 #se asigna variable de negacion
9         desplazador=0 #se inicializa el desplazamiento en 0
10      }else{
11        if(i=="sin" && negacion1==0){
12          #en caso de encontrar una negacion debil
13          negacion1=1
14          desplazador=0
15        }
16        else{
17
18          coincidencia =match(i, ex$Pal) #se busca la palabra en el campo
                Pal$
19          if(!is.na(coincidencia)){ #si se encontro la palabra en el dic
20
21
22            if(ex$polaridad[coincidencia]=="positivo"){ #si es positiva
23              if(negacion==1){ #si hay una negacion previa antes del limite
                    del desplazamiento
24                pola=pola-4 #se penaliza la polaridad del texto en 4 en
                      caso de haber previamente una negacion fuerte
25                negacion=0 #se restablece la bandera de negador
26              }
27              else{
28                if(negacion1==1){ #si hay una negacion debil
```

```
29                  pola=pola-3.5 #penalizamos en 3.5 la polaridad
30                  negacion1=0 #se restablece la bandera de negador
31                }
32                else{
33                  pola=pola+1 #si la palabra es positiva sin haber un
                        negador previo, se aumenta la polaridad en 1
34                }
35              }
36
37            }
38            if(ex$polaridad[coincidencia]=="negativo"){ #si es negativa,
                  simplemente disminuimos la polaridad en 1, y restablecemos
                  negadores
39              pola=pola-1
40              negacion=0
41              negacion1=0
42            }
43            if(ex$polaridad[coincidencia]=="neutro"){
44              desplazador=desplazador+1 #si es neutra, aumentamos en 1 el
                    desplazamiento$
45            }
46          }
47            }
48          }
49      }
50  }
```

Primero, se divide el texto en palabras. Luego, se realiza un recorrido por cada palabra perteneciente a este texto, en caso de ser una negación, se activa una variable bandera", la cual indica presencia del negador. Para este caso, se procede a analizar la siguiente palabra, si esta posee una polaridad positiva, se obtiene su polaridad y se le descuenta la polaridad total -4 o -3.5 dependiendo del negador encontrado. En caso de ser negativa, se desactiva la bandera y se disminuye la polaridad en 1. En caso que la palabra encontrada sea neutra, simplemente se desplaza en una unidad

la variable de desplazamiento.

Los pasos anteriores se repetirán en caso de no haber sobrepasado el tamaño de desplazamiento máximo. Caso contrario, simplemente se desactiva la bandera y todo el proceso anterior se realiza en caso de encontrar una negación.

En caso de no haber negación, simplemente se obtendrá la polaridad para cada palabra encontrada. Si no se encuentra, se búscara en los campos de expansión (sinónimo), como también se stemizará la palabra del texto y se búscara en los campos stemizado que fueron preparados previamentes en el lexicón. Una vez obtenida la polaridad de cada texto, se guardará en el archivo CSV temporal

Método de Orientación Semántica basado en Point mutual Information (SO - PMI)

Este método se basa en la búsqueda web, aplicando una fórmula de probabilidades, se estimará la polaridad de los Tweets previamente preprocesados.

A continuación, se observa la codificación del método:

```
1  #funcion que recibe un string a buscar en la web
2  GoogleHits <- function(input)
3  {
4
5  GoogleHits <- function(input)
6  {
7
8    #input="bachelet"
9    require(XML)
10   require(RCurl)
11   url <- paste("https://www.google.com/search?q=", input, sep = "")
12   CAINFO = paste(system.file(package="RCurl"), "/CurlSSL/ca-bundle.crt",
        sep = "")
13   script <- getURL(url, followlocation = TRUE, cainfo =
        CAINFO,ssl.verifyhost = 0L, ssl.verifypeer = 0L)
14   doc <- htmlParse(script)
15   res <- xpathSApply(doc, '//*/div[@id="resultStats"]', xmlValue)
16   cat(paste("\nYour Search URL:\n", url, "\n", sep = ""))
```

```
  cat("\nNo. of Hits:\n") # get rid of cat text if not wanted
  return(as.double(gsub("[^0-9]", "", res)))
}
```

Como el método de SO-PMI se basa en la búsqueda web, lo primero a implementar es una función que permita realizar tales búsquedas. GoogleHits recibe como parámetro de entrada un texto, el cual puede ser una palabra o combinación de ellas (separadas con un símbolo +). La función realizará la búsqueda en Google, recibiendo el HTML correspondiente a los resultados de esta. Finalmente, se evaluan las etiquetas HTML, buscando un div con id=resultStats. Este contenedor tiene el número de coincidencias encontradas para tal búsqeda, se extrae este valor y se retorna.

A continuación, la segunda parte del código:

```
#lista de palabras positivas
positivo1 ="bueno"
positivo2 ="agradable"
positivo3 ="excelente"
positivo4 ="positivo"
positivo5 ="afortunado"
positivo6 ="correcto"
positivo7 ="superior"

#lista de palabras negativas
negativo1 ="malo"
negativo2 ="desagradable"
negativo3 ="pesimo"
negativo4 ="negativo"
negativo5 ="desafortunado"
negativo6 ="incorrecto"
negativo7 ="inferior"

#busqueda palabras pos
busquedapositiva1=GoogleHits(positivo1)
Sys.sleep(3)
busquedapositiva2=GoogleHits(positivo2)
```

```
23 Sys.sleep(3)
24 busquedapositiva3=GoogleHits(positivo3)
25 Sys.sleep(3)
26 busquedapositiva4=GoogleHits(positivo4)
27 Sys.sleep(3)
28 busquedapositiva5=GoogleHits(positivo5)
29 Sys.sleep(3)
30 busquedapositiva6=GoogleHits(positivo6)
31 Sys.sleep(3)
32 busquedapositiva7=GoogleHits(positivo7)
33
34 #busqueda palabras negativas
35 busquedanegativa1=GoogleHits(negativo1)
36 Sys.sleep(3)
37 busquedanegativa2=GoogleHits(negativo2)
38 Sys.sleep(3)
39 busquedanegativa3=GoogleHits(negativo3)
40 Sys.sleep(3)
41 busquedanegativa4=GoogleHits(negativo4)
42 Sys.sleep(3)
43 busquedanegativa5=GoogleHits(negativo5)
44 Sys.sleep(3)
45 busquedanegativa6=GoogleHits(negativo6)
46 Sys.sleep(3)
47 busquedanegativa7=GoogleHits(negativo7)
48 Sys.sleep(3)
```

Tal como en el método anterior, se leerá el archivo CSV temporal y obtener los Tweets ya preprocesados. Fijarse que para este método, se debe realizar una búsqueda en Google para las palabras pivote (positiva y negativas). También es importantísimo destacar que se utilizará una función de pausa entre cada búsqueda ya que se realizarán muchas busquedas, y en este caso el buscador detecta solicitudes masivas a sus servidores (boot), por lo cual las serán bloqueadas, no pudiendo retornar la cantidad de coincidencias encontradas. El empleo de la pausa, permite simular

búsquedas realizadas por humanos, realizándolas cada cierto tiempo.

Es evidente que al utilizar una pausa entre cada búsqeda, el tiempo de ejecución de este método de clasificación será mucho mas largo que cualquier otro.

A continuación, se aprecia la tercera parte del código:

```
u<-path.expand('~')
setwd(file.path(u,"sentimets_workflow"))
u<-gsub("[/]","\\\\",u)
ruta_principal<-paste(u,"\\sentimets_workflow\\" ,sep="")
archivo_ruta<-paste(ruta_principal,"temp.csv" ,sep="")
e<-read.table(file =archivo_ruta, header=TRUE,sep=";",fill = TRUE )
textos<-e$prepro


#productoria palabras negativas
valor2=1
valor2=valor2*busquedanegativa1
valor2=valor2*busquedanegativa2
valor2=valor2*busquedanegativa3
valor2=valor2*busquedanegativa4
valor2=valor2*busquedanegativa5
valor2=valor2*busquedanegativa6
valor2=valor2*busquedanegativa7


#productoria palabras positivas

valor4=1
valor4=valor4*busquedapositiva1
valor4=valor4*busquedapositiva2
valor4=valor4*busquedapositiva3
valor4=valor4*busquedapositiva4
valor4=valor4*busquedapositiva5
valor4=valor4*busquedapositiva6
valor4=valor4*busquedapositiva7
```

```
31
32 cont<-1
33 polaridades<-list()
34 pos<-1
35 #recorro cada Tweet y obtengo cada palabra, luego cada una buscarla en el
       diccionario
36 for (i in textos){
37
38   acum=0
39   cont=0
40   print(i)
41   p<-(strsplit(i, "[[:punct:] ]")[[1]])
42   for (j in p){
43     j<-gsub("[...]", "", j)
44     print(j)
45     #parejas con positivos
46     pareja_buena1<-paste(j,"bueno" ,sep="+")
47     pareja_buena2<-paste(j,"agradable" ,sep="+")
48     pareja_buena3<-paste(j,"excelente" ,sep="+")
49     pareja_buena4<-paste(j,"positivo" ,sep="+")
50     pareja_buena5<-paste(j,"afortunado" ,sep="+")
51     pareja_buena6<-paste(j,"correcto" ,sep="+")
52     pareja_buena7<-paste(j,"superior" ,sep="+")
53
54     #busqueda parejas positivas
55     busqueda1=GoogleHits(pareja_buena1)
56     Sys.sleep(1)
57     busqueda2=GoogleHits(pareja_buena2)
58     Sys.sleep(1)
59     busqueda3=GoogleHits(pareja_buena3)
60     Sys.sleep(1)
61     busqueda4=GoogleHits(pareja_buena4)
62     Sys.sleep(1)
63     busqueda5=GoogleHits(pareja_buena5)
64     Sys.sleep(1)
```

```
65      busqueda6=GoogleHits(pareja_buena6)
66      Sys.sleep(1)
67      busqueda7=GoogleHits(pareja_buena7)
68
69      #productoria combinacion pareja positiva
70      valor1=1
71      valor1=valor1*busqueda1
72      valor1=valor1*busqueda2
73      valor1=valor1*busqueda3
74      valor1=valor1*busqueda4
75      valor1=valor1*busqueda5
76      valor1=valor1*busqueda6
77      valor1=valor1*busqueda7
78
79      #parejas con negativos
80      pareja_mala1<-paste(j,"malo" ,sep="+")
81      pareja_mala2<-paste(j,"desagradable" ,sep="+")
82      pareja_mala3<-paste(j,"pesimo" ,sep="+")
83      pareja_mala4<-paste(j,"negativo" ,sep="+")
84      pareja_mala5<-paste(j,"desaforunado" ,sep="+")
85      pareja_mala6<-paste(j,"incorrecto" ,sep="+")
86      pareja_mala7<-paste(j,"inferior" ,sep="+")
87
88      #busqueda pareja negativa
89      busqueda8=GoogleHits(pareja_mala1)
90      Sys.sleep(1)
91      busqueda9=GoogleHits(pareja_mala2)
92      Sys.sleep(1)
93      busqueda10=GoogleHits(pareja_mala3)
94      Sys.sleep(1)
95      busqueda11=GoogleHits(pareja_mala4)
96      Sys.sleep(1)
97      busqueda12=GoogleHits(pareja_mala5)
98      Sys.sleep(1)
99      busqueda13=GoogleHits(pareja_mala6)
```

```
100     Sys.sleep(1)
101     busqueda14=GoogleHits(pareja_mala7)
102
103     #productoria parejas negativas
104     valor3=1
105     valor3=valor3*busqueda8
106     valor3=valor3*busqueda9
107     valor3=valor3*busqueda10
108     valor3=valor3*busqueda11
109     valor3=valor3*busqueda12
110     valor3=valor3*busqueda13
111     valor3=valor3*busqueda14
112     numerador=valor1*valor2
113     denominador=valor3*valor4
114
115     division=numerador/denominador
116     log=log2(division) #mayor a 0 indica palabra positiva
117     print(log)
118     acum=acum+log
119     cont=cont+1
120   }
121
122   promedio=acum/cont
123   polaridades[pos]=promedio
124   pos=pos+1 #$
125
126 }
```

Como ya se obtuvieron los resultados de búsqueda para cada palabra pivote, se puede realizar la productoria para ambos conjuntos. El siguiente paso es concatenar cada palabra con todos los elementos de cada pivote y obtener la cantidad de aciertos encontrados en la web. Una vez obtenidos estos elementos, se puede realizar el cálculo de la polaridad mediante la fórmula de Turney para cada palabra (Turney y Peter D, 2002). Finalizado este proceso, se obtiene el promedio para todas las palabras.

Un resultado positivo, indica que el texto es de esa polaridad. Caso contrario, es negativo.

6.8.2. SubWorkflow APIs Aprendizaje automático

Este SubWorkflow representa el enfoque de aprendizaje automático. Para ello se utilizó diversas APIs que permiten realizar la clasificación mediante este enfoque.

El flujo de trabajo se puede visualizar en la figura 6.18:

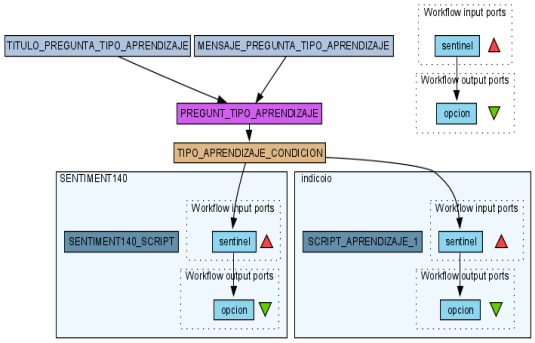


Figura 6.18: Modelo APIS aprendizaje automático.

Se solicita al usuario que API de aprendizaje automático utilizar (figura 6.19)

Figura 6.19: Seleccionando que API de aprendizaje utilizar

A continuación, el código que implementa cada API.

API sentiment140

API basada en clasificador basado en supervisión a distancia. Observemos el código:

```
library(sentiment) #cargar libreria

x<-"Texto de prueba"
sentimiento<-sentiment(x) #llamada a la API
sentimiento$polarity #obtencin polaridad $
```

Los pasos son simples, ya que una vez obtenidos los Tweet preprocesados, sólo se debe hacer un llamado a la función sentiment de la API, enviando como parámetro de entrada los textos. Esta retornará una lista de objetos que contiene los siguientes elementos:

- text: Los textos enviados como parámetro de entrada (Tweet preprocesado).
- polarity: La polaridad que predijo el clasificador (neutral, positive, negative).
- languaje: Idioma en el que fue escrito el texto.

Observar un poco el comportamiento de la función sentiment. Para ello, se ha extraido los elementos del paquete para llegar al código.

```
sentiment <- function(text, verbose=FALSE)
{
  require(RCurl)
  require(rjson)
  require(plyr) #carga dependencias

  r <- dynCurlReader() #determina el el tipo de contenido del cuerpo del
      encabezado HTTP

  # get rid of single quote
  text <- gsub("'", ' ' ,text)
  text <- gsub('"', ' ' ,text)

```

```
13  x  <- paste( sprintf("{'text': '%s'}", text),
14        collapse = "," )
15
16  curlPerform(postfields =
17              sprintf('{"language": "auto", "data": [%s]}', x),
18            url = "http://www.sentiment140.com/api/bulkClassifyJson",
19            verbose = verbose,
20            post = 1L,
21            writefunction = r$update) #realiza solicitud a la url con los
                  textos pasados por entrada
22
23  r$value() #se obtiene un json con la respuesta a la solicitud
24   }
```

Básicamente lo que se hace, es realizar una solicitud HTTP con los textos a ser clasificados. Una vez realizada, se recibirá como respuesta un archivo json, el cual contiene los textos, polaridad e idioma.

```
1  "{\"data\":[{\"text\":\"acaso hizo donde trabajando hijo estalls caso
       cava\"l\"polarity\":2,\"meta\":{\"language\":\"es\"}}
```

Para obtener la lista de objetos detallada como salida de esta función. Simplemente, se recorre el arreglo json, añadiendo a la lista de objetos por cada posición los datos correspondiente, reemplazando el número de polaridad. Por ejemplo, si el valor es 0, se reemplaza por negative.

Una vez obtenida la lista de objetos, se extraw de ella el atributo polarity y se anáde al dataframe que contiene todos los Tweets. Finalmente se guarda este dataframe en el archivo CSV temporal.

API indico

API basada en aprendizaje por transferencia. Esta es una API cerrada, por la cual, no se podrá visualizar el código que realiza la tarea de clasificación. Sólo se visualizará el cómo se llama a esta API.

```
1
```

```
library(indicoio)
library(stringi) #carga librerias


x="texto de prueba"

x<-stri_trans_general(x,"Latin-ASCII") #permite remover tildes y
x<-iconv(x, "ASCII", "UTF-8") #transforma a UTF-8
sent<-sentiment(x, api_key = '***',language='detect') #funcion que
    determina la polaridad, se pasa como parametro el listado de textos,
    la llave de registro unica y el lenguaje del texto
```

Como siempre, hay que obtener el archivo CSV temporal y los Tweets preprocesados. Lo siguiente no requiere tanta explicación, ya que sólo basta con hacer un llamado a la API indico mediante la función sentiment, la cual se encarga en devolver una lista de polaridades, correspondiente a la lista de Tweets que se han enviado. Las polaridades obtenidas son números decimales desde 0 a 1, indicando la probabilidad de que un texto dado sea positivo.

6.9. Cuarta etapa: Creación archivo CSV final

Una vez realizadas todas las tareas que se han seleccionado, se debe crear el archivo de salida que permita al usuario su visualización y/o modificación.

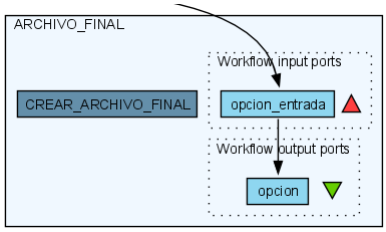


Figura 6.20: Proceso de creación archivo CSV final. Se crea al final de todo el proceso.

A continuación se observa la codificación:

```
#funcion que abre un archivo dada una ruta como entrada (abrimos csv
    final) valida para windows y linux
```

```
2  opendir <- function(dir){
3    if (.Platform['OS.type'] == "windows"){
4      shell.exec(dir)
5    } else {
6      system(paste(Sys.getenv("R_BROWSER"), dir))
7    }
8  }
9
10 u<-path.expand('~') #ruta absoluta del usuario
11 u<-gsub("[/]","\\\\",u)
12 ruta_principal<-paste(u,"\\sentimets_workflow\\" ,sep="")
13 archivo_ruta<-paste(ruta_principal,"temp.csv" ,sep="") #generar ruta al
       archivo temporal
14 e<-read.table(file =archivo_ruta, header=TRUE,fill = TRUE,sep=";") #abrir
       archivo temporal
15
16
17 palabra=e$palabra
18 dnow <- data.frame(e)
19 v<-head(dnow,1)
20 v<-v$palabra
21 archivo<-v
22 #obtiene la palabra que se busco (temas o usuarios)
23 fecha<-format(Sys.time(), "%d-%m-%Y %T") #fecha y hora del sistema
24 fech<-sub(":", "", fecha)
25 fech<-sub(":", "", fech) #sacamos los : de la fecha
26 ruta_sentiment<-paste(archivo,fech ,sep=" ") #concatenar palabra busqueda
       y fecha
27 dir.create(file.path(ruta_principal, ruta_sentiment), showWarnings =
       FALSE) #crear directorio
28 setwd(file.path(ruta_principal, ruta_sentiment)) #establecer como
       directorio de trabajo el creado
29 direct<-paste(ruta_principal,ruta_sentiment ,sep="")#concatena ruta
       principal con nuevo directorio
30 ruta_final<-paste(direct,"\\",sep="")
```

```
ruta_final<-paste(ruta_final,ruta_sentiment,sep="") #se genera la ruta en
    donde guardamos el archivo
extension<-".csv"
ruta_final<-paste(ruta_final,extension ,sep="") #aadimos .csv al nombre
    del archivo
e$palabra=NULL
df = as.matrix(e)
df_Place2 = data.frame(lapply(e, as.character), stringsAsFactors=FALSE)
write.table(df_Place2, file =ruta_final,row.names=FALSE,sep=";")
#guardamos el archivo csv en la ruta generada
if (file.exists(archivo_ruta)) file.remove(archivo_ruta) #borramos
    archivo temporal

opendir(direct) #abre carpeta del archivo CSV final y el archivo mismo
opendir(ruta_final) #$
```

El primer paso a realizar es obtener el archivo CSV temporal. Luego, extraer el atributo palabra (añadido en extracción, puede ser el nombre de usuario a buscar Tweets o la/las palabras claves a buscar). Para crear el archivo final, se concatena la palabra junto a la fecha y hora actual del sistema. A partir de eso, se genera una carpeta, en la cual, se ingresará el archivo CSV final. Para realizar esto, se debe generar la ruta absoluta en donde se guardará el archivo. Esto se consigue concatenando la ruta principal de trabajo con la nueva carpeta generada, obtenido esto,se vuelve a concantenar, esta vez, la ruta generada junto a la palabra de búsqueda y hora de sistema que se ha obtenido anteriormente. Finalmente anádir la extensión csv. Ya se tiene la ruta absoluta, sólo queda guardar el dataframe del archivo CSV temporal a la ruta generada y borrar el archivo CSV temporal.

Para entender esto de mejor manera, se visualizará el siguiente ejemplo:

Se han buscado Tweets realizados por el usuario ABC. Se creará una carpeta llamada ABC 20 04 2018 000000(20 04 2018 000000 corresponde a fecha y hora del sistema). Una vez creada esta carpeta se concatena la ruta principal de trabajo con la ya generada, quedando como C:\Usuarios\NombreUsuario\sentimet_workflow\ABC 20 04 2018 000000\. Finalmente, concatenar esta ruta nuevamente con ABC 20 04 2018 000000.CSV, obteniendo la ruta absoluta en donde se guarda el archivo CSV

final C:\Usuarios\NombreUsuario\sentimet_workflow\ABC 20 04 2018 000000\ABC 20 04 2018 000000.CSV

Nota: Se realiza este mismo proceso para generar el archivo CSV temporal. La diferencia va en que el nombre de este archivo es temp.CSV y se almacena en la carpeta principal de trabajo

Capítulo 7

Pruebas

7.1. Pruebas de usabilidad

Para la prueba de usabilidad se ha seleccionado un grupo 4 personas, todas con conocimientos informáticos en distintos niveles. Por turno, se les facilitó un computador portatil a cada una de estos individuos para que evaluen el funcionamiento del Workflow final. El cuestionario utilizado está basado en la propuesta por John Brooke.(Brooke, 1986)

Pruebas de usabilidad					
Nivel de acuerdo	1	2	3	4	5
Utilizaría con frecuencia el programa					
Encontré el programa muy complejo					
Fue fácil utilizarlo					
Necesitaria de un experto para utilizarlo					
Las diversas funciones están bien integradas					
Hubo demasiado inconsistencia visual					
Lo encontré muy difícil de usar					
Las personas lo aprenderían a usar rapidamente					
Me sentí muy confiado en la nevegación					
Necesitaría aprender más antes de utilizarlo					

El significado de cada número es el siguiente:

Significado	
1	Muy en desacuerdo
2	En desacuerdo
3	Ni de acuerdo ni en desacuerdo
4	De acuerdo
5	Muy de acuerdo

Esta prueba se desarrolló de acuerdo al método *Protocolo del pensamiento manifestado*, en la que se solicita que un usuario exprese en voz alta sus pensamientos, sensaciones y opiniones mientras interactúa con el sistema.

7.2. Resultados prueba de usabilidad

Esta prueba consistió en 4 fases. Primero se les explicó la finalidad de la realización de esta prueba. En la segunda fase, se les explicó el funcionamiento de la aplicación y sus aspectos básicos, realizando un ejemplo de uso supervisado. En la tercera fase, se les solicitó que utilizaran la aplicación en forma autónoma. Como última fase, los usuarios debieron expresar sus impresiones.

Los resultados individuales se encuentran en el Anexo. En la tabla 7.1, se aprecian los resultados promediados.

Pomedio de los cuestionarios de usabilidad	
Utilizaría con frecuencia el programa	3,75
Encontré el programa muy complejo	1,75
Fue fácil utilizarlo	4,0
Necesitaria de un experto para utilizarlo	2,0
Las diversas funciones están bien integradas	4,25
Hubo demasiado inconsistencia visual	1,5
Lo encontré muy difícil de usar	1,75
Las personas lo aprenderían a usar rapidamente	4,0
Me sentí muy confiado en la nevegación	4,0
Necesitaría aprender más antes de utilizarlo	2,25

Cuadro 7.1: Resultado promediado pruebas de usabilidad

7.3. Pruebas de requerimiento

Para realizar esta prueba, se ha seleccionado un individuo con sólidos conocimientos en desarrollo de software. Se le solicitó leer la especificación de requisitos y posteriormente utilizar el sistema. En la tabla 7.3 se muestran los elementos a considerar en la prueba.

Pruebas de requerimientos						
	R1	R2	R3	R4	R5	R6
El requisito se encuentra debidamente documentado						
El requisito no tiene errores de sintaxis y morfológicos						
El requisito cumple con las expectativas del cliente						
El requisito no posee ambigüedades						

Cuadro 7.2: Resultados pruebas de requerimientos

C: Cumple NC: No cumple

7.4. Resultado Pruebas de requerimientos

En la tabla 7.3 se aprecian los resultados de la prueba de requerimiento para la aplicación desarrollada.

Pruebas de requerimientos						
	R1	R2	R3	R4	R5	R6
El requisito se encuentra debidamente documentado	C	C	C	C	C	C
El requisito no tiene errores de sintaxis y morfológicos	C	C	C	C	C	C
El requisito cumple con las expectativas del cliente	C	C	C	C	C	C
El requisito no posee ambigüedades	C	C	C	C	C	C

Cuadro 7.3: Resultados pruebas de requerimientos

Según los resultados obtenidos, se puede comprobar el buen funcionamiento de la aplicación.

Capítulo 8

Conclusión

La minería de opiniones es un tema que está en constante desarrollo, muchas empresas están interesadas en saber que opina la gente sobre sus productos. Ante esto, muchas empresas desarrolladoras de software han estado creando soluciones para satisfacer estas necesidades. Sin embargo, un gran porcentaje de estas son de pago. Si bien, existen soluciones Open Source, estas requieren conocimientos de programación, provocando un obstáculo para que estas personas puedan desarrollar sus temas, desencadenando frustración y ralentizando el avance de este tema.

Otro inconveniente en el desarrollo de esta área, es la poca variedad de soluciones para textos escritos en español, existiendo muy pocas posibilidades de avance este ámbito.

8.1. Cumplimiento de objetivos

Como se ha observado en el transcurso de este informe, el objetivo principal de desarrollar una interfaz gráfica utilizando el paquete TwitterR de Rproject ha sido cumplido a su cabilidad. Ahora, se verá el cumplimiento de los objetivos específicos propuestos en el la sección 1.4.

Para cada uno se puede concluir lo siguiente:

- **Objetivo específico 1:** El estudió sobre el Procesamiento del Lenguaje natural y Análisis de sentimiento se realizó en el transcurso de la asignatura anteproyecto de título. Se estudío las distintas etapas, fuentes de extracción de

datos, preprocesamiento de estos y los distintos enfoques con que se pueden clasificar los textos escritos (Véase capítulo 2).

- **Objetivo específico 2:** Se elaboró una especificación de requisitos basada en algunos aspectos de la IEEE830, en ella se detallaron todos los requisitos funcionales y no funcionales del sistema. Esta especificación se expuso a un grupo de personas con conocimientos en desarrollo de software, fue sometida a un feedback, mejorada y posteriormente aprobada (Véase capítulo 5).

- **Objetivo específico 3:** Se realizó un diagrama de actividad por cada funcionalidad, detallando el funcionamiento a nivel gráfico de la aplicación, tanto en forma integra, como individual, mostrando el funcionamiento de cada una de sus partes. Junto a ello, se elaboraró un modelo datos, como también, las distintas pantallas, mostrando un anticipo del como es la aplicación.

- **Objetivo específico 4:** Para el desarrollo de esta etapa, fue necesario estudiar R Project, ya que su forma de realizar ciertas operaciones es distinta a la de lenguajes de programación mas comunes, por ejemplo el uso de vectorización. Además fue necesario el estudio de Taverna Workflow, ya que es una plataforma jamás antes vista, ocasionando dificultades en el desarrollo de esta etapa. Esto fue debido a su enfoque de programación distinto al de aplicaciones tradicionales. Junto a ello, la poca documentación existente en la web.(Véase capítulo 7).

- **Objetivo específico 5:** Se realizó la prueba de usabilidad, funcionamiento y de requisitos. Si bien, existen algunos aspectos a mejorar, los resultados de las pruebas arrojan que la aplicación funciona correctamente. (Véase capítulo 8).

- **Objetivo específico 6:**No ha sido necesario crear un paquete de R Project, ya que en la etapa implementación, se realizaron llamadas a funciones provenientes de otros paquetes. Por lo cual, para evitar el "re-llamado de funciones", se ha decidido no crear un paquete.

- **Objetivo específico 7:** El Workflow final es compartido con la comunidad de Taverna, como también con los usuarios finales. Esto permitirá la recepción de críticas y/o sugerencias con la finalidad de realizar posibles mejoras al Software.

La idea de este desarrollo es haber construido una solución que permita que personas realicen análisis de sentimientos en Twitter de forma rápida, efectiva y sin la necesidad de codificar. Esta aplicación está optimizada para trabajar con textos escritos en español. Además es de fácil transporte, ya que funciona bajo cualquier plataforma de escritorio, sólo se requieren instalar unos programas y paquetes para su correcto funcionamiento. Finalmente, esta aplicación también está orientada a personas con conocimientos en programación, ya que los códigos de la distintas funcionalidades son fácilmente accesibles, por lo cual, este sistema puede ser expandido, mejorado o personalizado a gusto del usuario.

Como recomendación, es importante mencionar que el desarrollo con Taverna Workflow se puede hacer complejo debido a la poca documentación y complejidad de realizar ciertas acciones que son relativamente fáciles en otras platafomas.

Finalmente, el desarrollo de esta aplicación se puede considerar como una base para una mejora en el análisis de sentimiento para textos escritos en español. Queda abierta la puerta hacia personas interesadas en mejorar este software, facilitando las herramienta y documentación requerida para un desarrollo cómodo y entusiasta.

Bibliografía

Alec Go Lei Huang, Richa Bhayani, y Lei Huang. Sentiment140 - A Twitter Sentiment Analysis Tool. 2017. URL http://help.sentiment140.com/.

Bitext- Official Page. Bitext - NLP to enhance the understanding between humans and machines. 2017. URL https://www.bitext.com/sentiment-analysis/.

Bo Pang, Shivakumar Vaithyanathan, y Lillian Lee. Thumbs up? Sentiment Classification using Machine Learning Techniques. 2017. URL https://www.cs.cornell.edu/home/llee/papers/sentiment.pdf.

John Brooke. Usability evaluation in industry. 1986.

Jaime Carbonell. El procesamiento del lenguaje natural, tecnologa en transicin. 5, 2018. URL https://cvc.cervantes.es/obref/congresos/sevilla/tecnologias/ponenc_carbonell.htm.

Chris Okugami. sentiment140 - R package for Twitter sentiment text analysis. 2017a. URL https://github.com/okugami79/sentiment140.

Chris Okugami. sentiment140 - R trainingandtestdata. 2017b. URL https://docs.google.com/file/d/0B04GJPshIjmPRnZManQwWEdTZjg/edit.

Indico. indico - text and image analysis powered by machine learning. 5, 2017. URL https://indico.io/.

Jeff Gentry. R Based Twitter Client. 2016. URL https://cran.r-project.org/web/packages/twitteR/twitteR.pdf.

Jos Pino-Daz. Tutorial de R-Text Mining Solution. 2017. URL https://www.researchgate.net/publication/307546514_Tutorial_de_R-Text_Mining_Solution.

Katherine Wolstencroft, Robert Haines, Donal Fellows, Alan Williams, y David Withers... The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud. 2013.

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, y Manfred Stede. Lexicon-based methods for sentiment analysis. 2011.

Robert D. Stevens, Alan J. Robinson, y Carole A. Goble. mygrid: personalised bioinformatics on the information grid. 2003.

Roberto Gonzlez-Ibez, Smaranda Muresan, y Nina Wacholder. Identifying Sarcasm in Twitter: A Closer Look. 2017.

RStudio. RStudio. 2017. URL https://www.rstudio.com/products/rstudio/.

Taverna. Taverna Workflow. 2017. URL http://www.taverna.org.uk/documentation/taverna-2-x/.

Taverna Official page. APACHE Taverna. 2017. URL https://taverna.incubator.apache.org/introduction/.

The R Foundation. What is R? 2017. URL https://www.r-project.org/about.html.

The University of Manchester y University of Southampton. myexperiment - workflows. 5, 2018.

Turney y Peter D. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. 5, 2002.

Twitter. Cmo empezar con Twitter-Centro de ayuda. 2017. URL https://support.twitter.com/articles/332061.

Yanchang Zhao. Twitter Data Analysis with R { Text Mining and Social Network Analysis. 2016. URL http://www.RDataMining.com.

()

Anexos

Anexo A

A.1. Manual de usuario

Este manual está destinado que poseen muy pocos conocimientos en programación. A continuación, se explican los pasos para la correcta ejecución de la aplicación.

A.1.1. Instalando R Project:

Como la aplicación se ejecutará en forma local, se debe instalar este software para su funcionamiento.

- Paso 1: Descargar R Project desde `https://dirichlet.mat.puc.cl/`. Debe seleccionar que sistema operativo está usando. Para este manual, se utilizará Windows 10 64 bits.

- Paso 2: Ejecutar el instalador y seguir los pasos del asistente tal como cualquier instalación de Windows.

- Paso 3: Iniciar la aplicación. Para ello, ir a Inicio, ubicar la ruta de instalación de R y ejecutar (figura A.1).

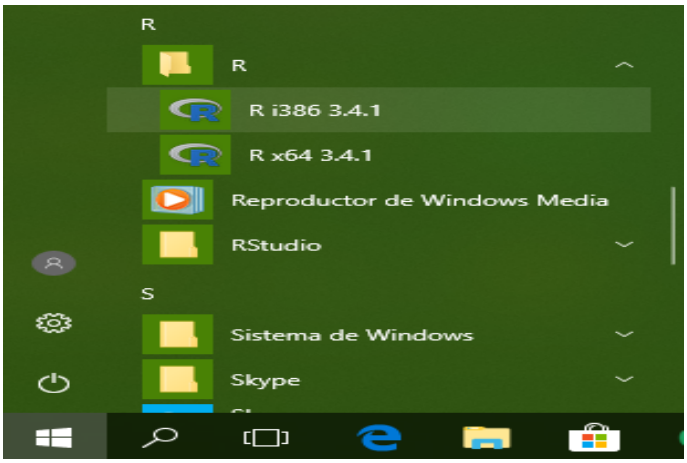


Figura A.1: Iniciando R desde el menú inicio

Una vez ejecutado R Project, se verá la siguiente pantalla (figura A.2).

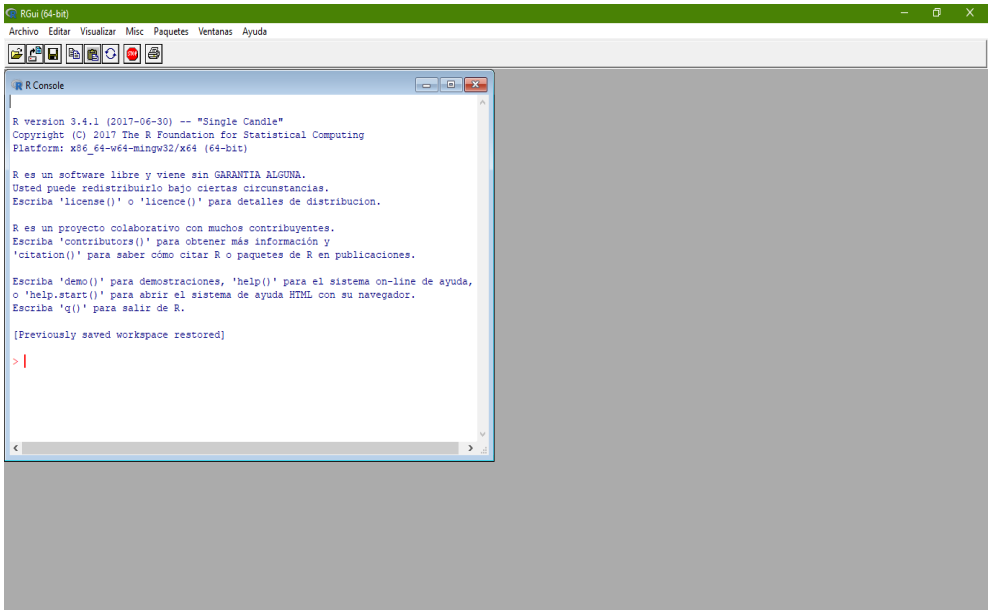


Figura A.2: Pantalla principal de R

A.1.2. Instalando Taverna Workflow

Taverna es la plataforma en donde se ejecutará la aplicación. Por este motivo, su intalación es indispensable. Los pasos para instalar son los siguientes:

- Paso 1: Ingresar a `https://bitbucket.org/taverna/taverna-workbench-product/downloads/`, seleccionar su sistema operativo y descargar.
- Paso 2: Ejecutar el instalador previamente descargado y seguir los pasos del asistente instalación.
- Paso 3: Iniciar Taverna Workflow, buscando su ruta desde el menú inicio (figura A.1.2).

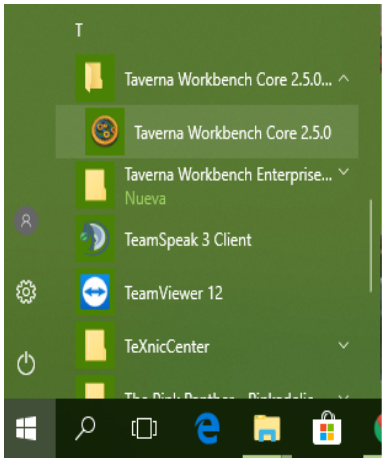


Figura A.3: Iniciando Taverna desde el menú inicio

Una vez iniciada la aplicación, aparecerá la siguiente pantalla (figura A.4).

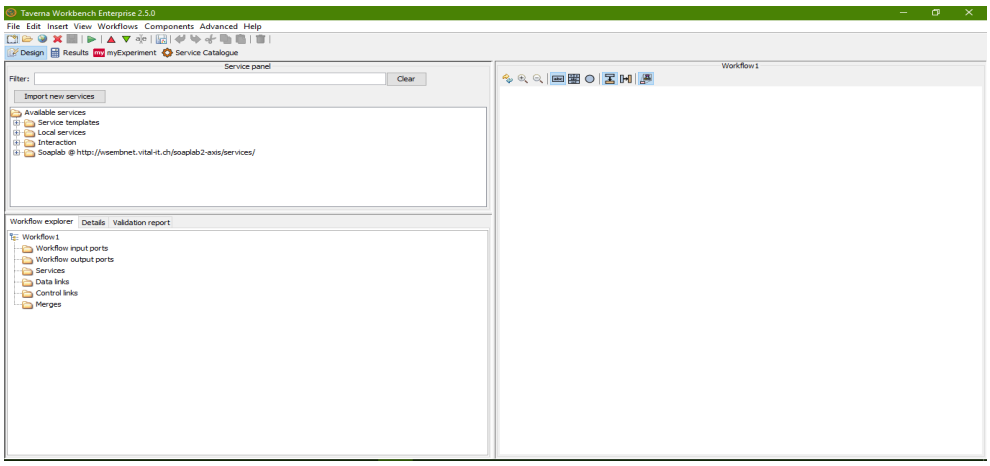


Figura A.4: Vista principal Taverna

A.1.3. Instalando paquetes necesarios

Una vez instalado R Project y Taverna, se deben instalar todos los paquetes necesarios para la ejecución del sistema. Los pasos a seguir son los siguientes.

Importante: Este paso se realiza por única vez. En caso de haber realizado la instalación de paquetes, ir al A.1.4.

- Paso 1: Ejecutar R Project ya previamente instalado
- Paso 2: En el menú superior, ir a "File" y hacer clic en "Open File":
- Paso 3: Se abrirá una ventana de selección de archivo. Debe ubicar la ruta en donde está el Script de instalación de paquetes
- Paso 4: Presionar el botón "Run" para comenzar el proceso de instalación (figura A.5).

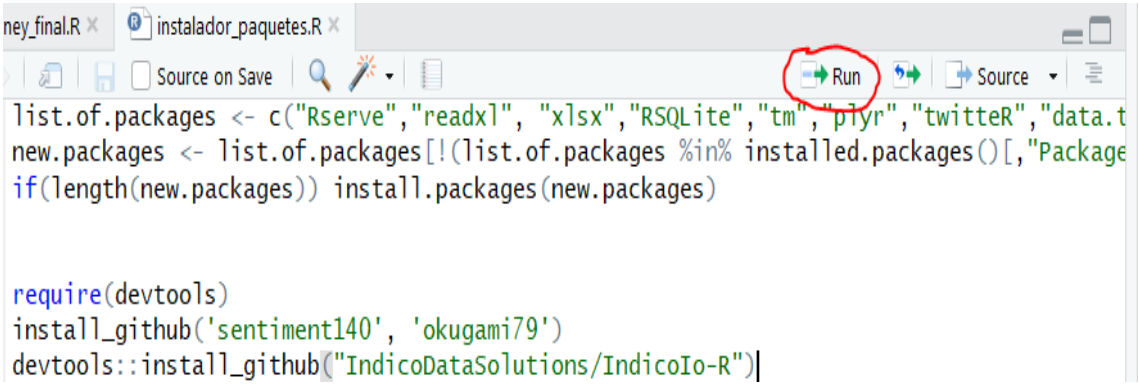


Figura A.5: Instalando los paquetes en R-Project

Debe esperar a que se realice el proceso de instalación. Una vez finalizado, está en condiciones de ejecutar el servidor para correr la aplicación en forma local.

A.1.4. Iniciando el servidor

Una vez instalados los paquetes, se debe correr un servidor en forma local. Este permitirá que la aplicación Taverna pueda ejecutar instrucciones de R. Por lo cual,

la funcionalidad del software es dependiente de esta etapa y se debe realizar cada vez que se desea ejecutar el Software.

Los pasos a seguir son los siguientes:

- Paso 1: Ejecutar R Project ya previamente instalado
- Paso 2: Ingresar las siguientes instrucciones:

```
1 library(Rserve)
2 Rserve()
```

Si han realizado correctamente los pasos anteriores, deberá visualizar el siguiente mensaje (figura A.6).

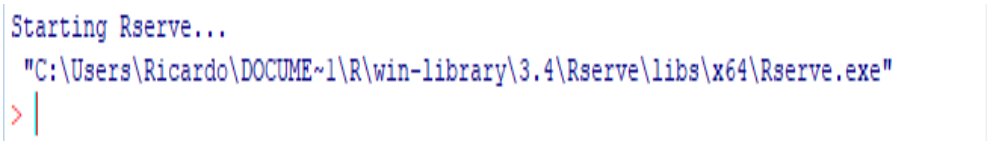


Figura A.6: Servidor iniciado correctamente

A.1.5. Iniciando la aplicación

Esta todo listo para abrir la aplicación. Para ello, ejecutar los siguientes pasos:

- Paso 1: Ejecutar la plataforma Taverna Workflow ya previamente instalado.
- Paso 2: Ir al menú File y seleccionar Open workflow... (figura A.7)

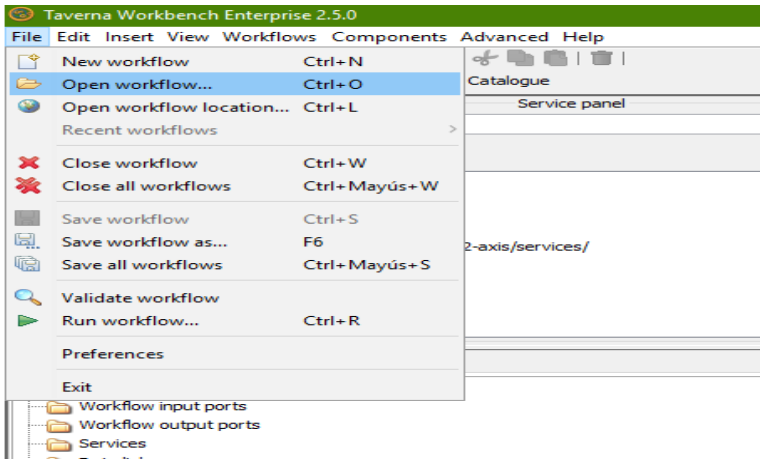


Figura A.7: Opción para abrir un Workflow

- Paso 3: Seleccionar el archivo app.t2flow (Todo archivo Taverna tendrá esta extensión, figura A.8).

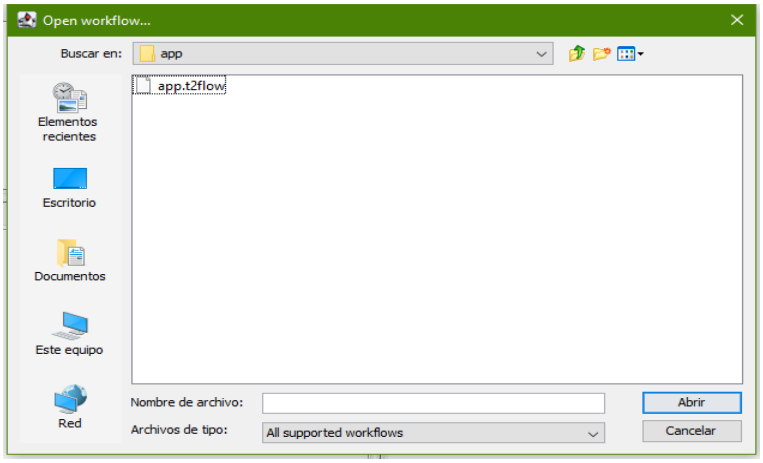


Figura A.8: Buscando el Workflow

Una vez completado estos pasos, podrá visualizar la interfaz (figura A.9).

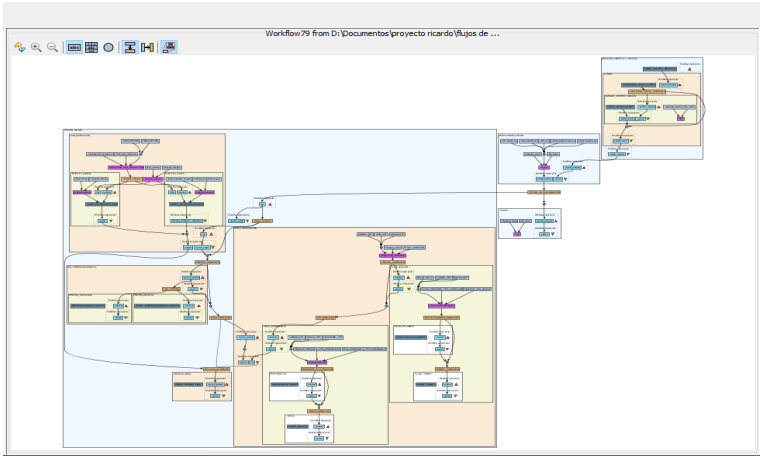


Figura A.9: Vista de la aplicación

A.1.6. Ejecutando la aplicación

Para ejecutar, presionar el botón "Play" ubicado en la barra de herramientas (figura A.10)

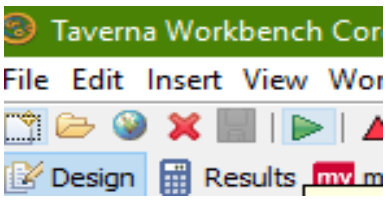


Figura A.10: Vista del botón "Play" para ejecutar la aplicación

A.1.7. Menú de opciones

El menú de opciones se compone de 5 tareas posibles a realizar (A.11). Estas son las siguientes:

- Sólo extracción: Como su nombre lo indica. Esta tarea consiste en sólo extraer una cierta cantidad de Tweets que cumplan con los requerimientos dados por el usuario. Se mostrarán los textos en su estado puro sin ninguna alteración. Al finalzar esta tarea, se obtendrá un archivo CSV con los Tweets extraidos y sus datos asociados.

- Sólo preprocesamiento: Tarea que consiste en seleccionar un archivo CSV que contenga Tweets ya extraídos (Solo extracción) a fin de realizarle una serie de

limpiezas de tal manera que estén preparados para ser clasificados. Se obtendrá el mismo archivo CSV de entrada, incluyendo los textos ya preprocesados.

- Sólo Clasificación: Tarea que consiste en seleccionar un archivo CSV que contenga Tweets ya preprocesados y los clasifique de acuerdo a su polaridad.
- Extracción y preprocesamiento: Estas tareas permiten en extraer una serie de Tweets que cumplan con los requisitos dados por el usuario para posteriormente preprocesarlos. Se obtendrá un archivo CSV con los Tweets extraidos, sus datos asociados y adicionalmente, los textos ya preprocesados.
- Extracción, preprocesamiento y clasificación: Esta opción, incluye la realización de todas las etapas en el análisis de sentimiento. Se realizan las etapas de extracción y preprocesamiento, y posteriormete, la obtención de polaridad en estos Tweets. Cabe destacar que los valores de polaridad pueden variar de un método a otro.
- Módulo de configuración: Dentro de esta opción se podrá seleccionar el archivo de léxico a utilizar, como también setear el tamaño máximo de desplazamiento para el manejo de negaciones.
- Salir: Finalizar la aplicación.

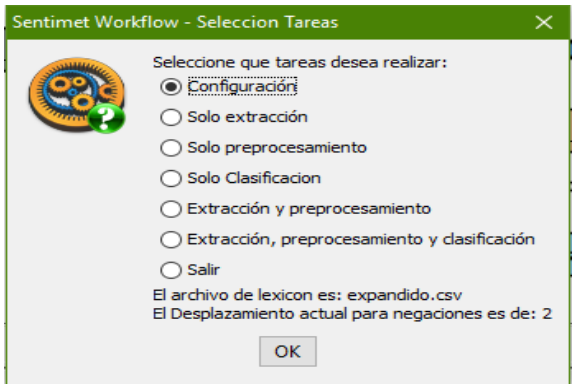


Figura A.11: Vista del menú de opciones

A.1.8. Módulo de configuración

Dentro de este menú se podrán configurar los siguientes parámetros (figura A.12).

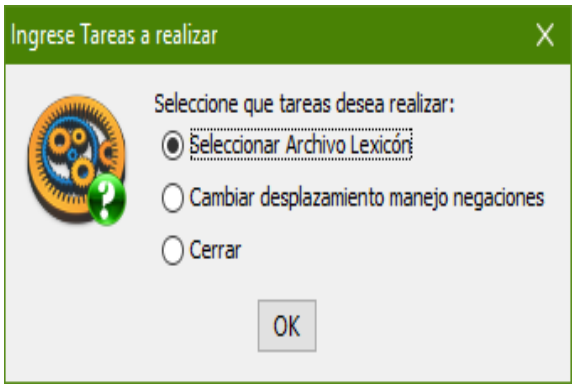


Figura A.12: Menú de configuración

- Cambiar tamaño desplazamiento para manejo de negaciones: Dentro de esta opción, se puede seleccionar solo 3 valores posibles. El valor 0, indica el no realizar manejo de negaciones para la clasificación mediante bolsa de palabras (figura A.13).

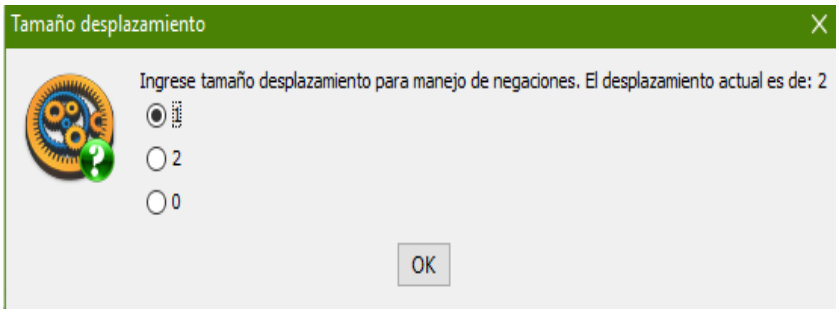


Figura A.13: Seleccionando tamaño de desplazamiento en caso de negaciones

- Seleccionar archivo de lexicón: Dentro de esta opción, se puede ingresar el archivo CSV que corresponda al diccionario de lexicón. Cabe destacar que el archivo será valido, siempre y cuando posea tres campos imprescindibles, los cuales son:

  - palabra: Como su nombre lo indica, este campo incluye la lista de palabras que tendrán su polaridad.
  - polaridad: Campo que incluirá la polaridad dada para cada palabra. Estas polaridades deben ser de valor positivo, negativo o neutro.

- lexicon: Este campo indica a que tipo de lexicón pertenece la palabra.

Nota: Los tres campos deben tener encabezado y llamados tal como se nombraron anteriormente. Recalcar que cada uno de ellos debe ser del mismo tamaño. Además el archivo debe ser guardado con codificación UTF-8 en caso de poseer tildes o carácteres pertenecientes al español. Un ejemplo de lexicón correcto se aprecia en la figura A.14.

	A	B	Q
1	lexicon	Pal	polaridad
2	red	abrazo	positivo
3	red	ángel	positivo
4	red	aplausos	positivo
5	red	arma	negativo
6	red	ascensor	negativo
7	red	asediado	negativo
8	red	avión	positivo
9	red	azúcar	positivo
10	red	boxeador	negativo
11	red	cachorro	positivo
12	red	casita	positivo
13	red	chaleco	negativo
14	red	corredor	negativo
15	red	cotilleo	negativo
16	red	cucaracha	negativo
17	red	culpable	negativo
18	red	diamante	positivo
19	red	enfurecido	negativo
20	red	esperanzadc	positivo
21	red	esposo	positivo
22	red	furcia	negativo
23	red	gimnasta	positivo

Figura A.14: Ejemplo archivo lexicón

A.1.9. Extracción de Tweets

Se explicará el como se realiza una extracción de Tweets en forma correcta. Estos pasos son válidos para toda opción que incluya este proceso. Los pasos a seguir son los siguientes

- Paso 1: Ingresar el tipo de extracción (figura A.15). Se puede realizar realizar dos tipos: Por palabras claves y por usuario. La descripción de cada uno es la siguiente:

  - Por tema: Extracción de Tweets por palabras claves de interés para el usuario.

- Por usuario: Extracción de los ultimos N Tweets emitidos por un usuario dado por parámetro.

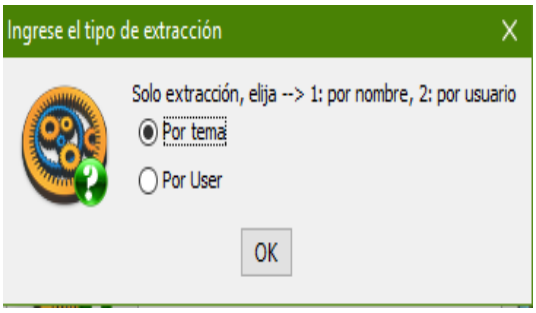


Figura A.15: Seleccionando tipo extracción. Como ejemplo, se realizará una extracción por palabras claves

- Paso 2: Ingresar la cantidad de Tweets a extraer. Por limitaciones de Twitter, se pueden extraer hasta 100 Tweets (figura A.16).

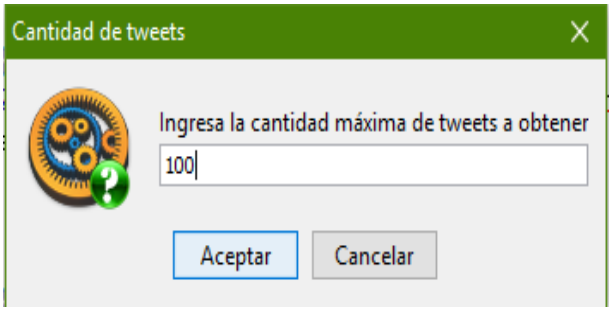


Figura A.16: Ingresando cantidad de Tweets

- Paso 3: Ingresar palabras claves de interés a buscar Tweets (figura A.17).

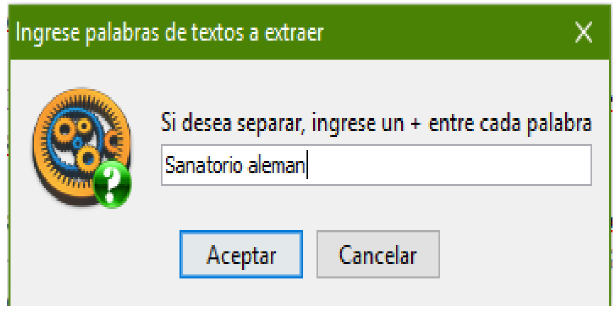
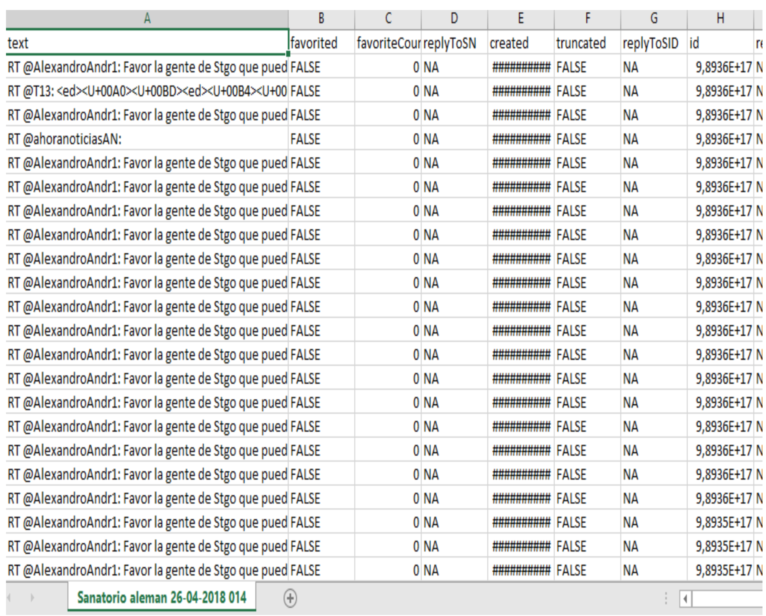


Figura A.17: Ingresando palabras claves

Si solo se ha realizado la extracción, se abrirá la carpeta contenedora del archivo CSV junto a este para ser visualizado. En caso de seleccionar una opción que incluya procesos posteriores, seguir leyendo los pasos para estos procesos (figura A.18).



A	B	C	D	E	F	G	H	
text	favorited	favoriteCour	replyToSN	created	truncated	replyToSID	id	r
RT @AlexandroAndr1: Favor la gente de Stgo que pued	FALSE	0	NA	##########	FALSE	NA	9,8936E+17	N
RT @T13: <ed><U+00A0><U+00BD><ed><U+00B4><U+00	FALSE	0	NA	##########	FALSE	NA	9,8936E+17	N
RT @AlexandroAndr1: Favor la gente de Stgo que pued	FALSE	0	NA	##########	FALSE	NA	9,8936E+17	N
RT @ahoranoticiasAN:	FALSE	0	NA	##########	FALSE	NA	9,8936E+17	N
RT @AlexandroAndr1: Favor la gente de Stgo que pued	FALSE	0	NA	##########	FALSE	NA	9,8936E+17	N
RT @AlexandroAndr1: Favor la gente de Stgo que pued	FALSE	0	NA	##########	FALSE	NA	9,8936E+17	N
RT @AlexandroAndr1: Favor la gente de Stgo que pued	FALSE	0	NA	##########	FALSE	NA	9,8936E+17	N
RT @AlexandroAndr1: Favor la gente de Stgo que pued	FALSE	0	NA	##########	FALSE	NA	9,8936E+17	N
RT @AlexandroAndr1: Favor la gente de Stgo que pued	FALSE	0	NA	##########	FALSE	NA	9,8936E+17	N
RT @AlexandroAndr1: Favor la gente de Stgo que pued	FALSE	0	NA	##########	FALSE	NA	9,8936E+17	N
RT @AlexandroAndr1: Favor la gente de Stgo que pued	FALSE	0	NA	##########	FALSE	NA	9,8936E+17	N
RT @AlexandroAndr1: Favor la gente de Stgo que pued	FALSE	0	NA	##########	FALSE	NA	9,8936E+17	N
RT @AlexandroAndr1: Favor la gente de Stgo que pued	FALSE	0	NA	##########	FALSE	NA	9,8936E+17	N
RT @AlexandroAndr1: Favor la gente de Stgo que pued	FALSE	0	NA	##########	FALSE	NA	9,8936E+17	N
RT @AlexandroAndr1: Favor la gente de Stgo que pued	FALSE	0	NA	##########	FALSE	NA	9,8936E+17	N
RT @AlexandroAndr1: Favor la gente de Stgo que pued	FALSE	0	NA	##########	FALSE	NA	9,8936E+17	N
RT @AlexandroAndr1: Favor la gente de Stgo que pued	FALSE	0	NA	##########	FALSE	NA	9,8936E+17	N
RT @AlexandroAndr1: Favor la gente de Stgo que pued	FALSE	0	NA	##########	FALSE	NA	9,8936E+17	N
RT @AlexandroAndr1: Favor la gente de Stgo que pued	FALSE	0	NA	##########	FALSE	NA	9,8936E+17	N
RT @AlexandroAndr1: Favor la gente de Stgo que pued	FALSE	0	NA	##########	FALSE	NA	9,8935E+17	N
RT @AlexandroAndr1: Favor la gente de Stgo que pued	FALSE	0	NA	##########	FALSE	NA	9,8935E+17	N
RT @AlexandroAndr1: Favor la gente de Stgo que pued	FALSE	0	NA	##########	FALSE	NA	9,8935E+17	N

Sanatorio aleman 26-04-2018 014

Figura A.18: Vista archivo generado

A.1.10. Preprocesando Tweets

Nota: Si ha seleccionado una opción que haya incluido la extracción, este proceso se realizará automáticamente sin requerir la intervención por parte del usuario. Ahora, en caso de requerir sólo procesamiento, deberá indicar el archivo CSV a procesar (figura A.19). Una vez ingresado, este será sometidos a un proceso de limpieza.

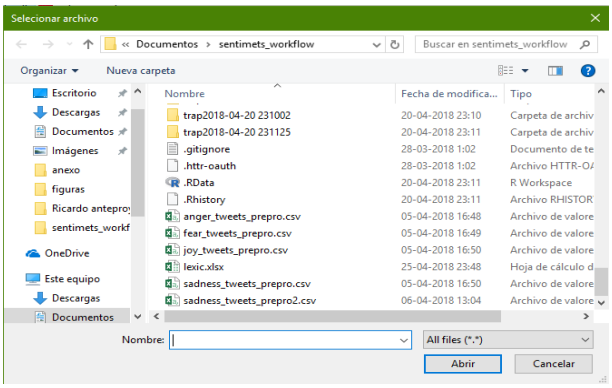


Figura A.19: Seleccionando archivo CSV a preprocesar

Si ha realizado preprocesamiento como último paso, se abrirá la carpeta contenedora junto al archivo CSV en donde podrá visualizar los Tweets ya preprocesados.

A.1.11. Clasificación de Tweets

Esta es considerada como la última etapa en el análisis de sentimiento. En caso de haber seleccionado "Sólo extracción", se solicitará el archivo CSV que contenga los Tweets ya preprocesados. Caso contrario, se considerará como elementos los obtenidos en la etapa anterior (Preprocesamiento). Una vez activado este proceso, se deben realizar son los siguientes:

- Paso 1: Seleccionar el enfoque de clasificación (figura A.20).

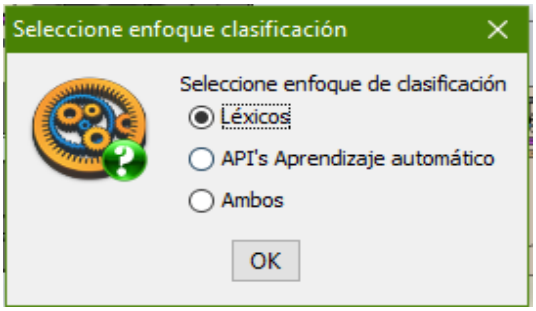


Figura A.20: Seleccionando el enfoque de clasificación. Por ejemplo, léxicos

- Paso 2: Seleccionar Método de clasificación de acuerdo al enfoque seleccionado (figura A.21).

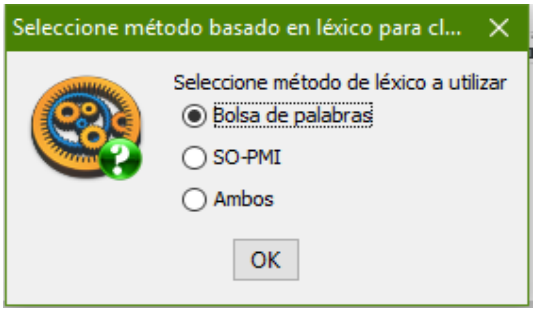


Figura A.21: Seleccionando el método bolsa de palabras

Se debe esperar a que finalice el proceso. En caso de seleccionar el Método de SO-PMI en enfoque por léxico, el procesamiento de Tweets tardará mucho más versus otros métodos. Esto es porque se realiza una búsqueda a través de la web.

A.2. Diagramas modelo desarrollado

En las figuras A.22 y A.23 se aprecia el modelo del workflow desarrollado.

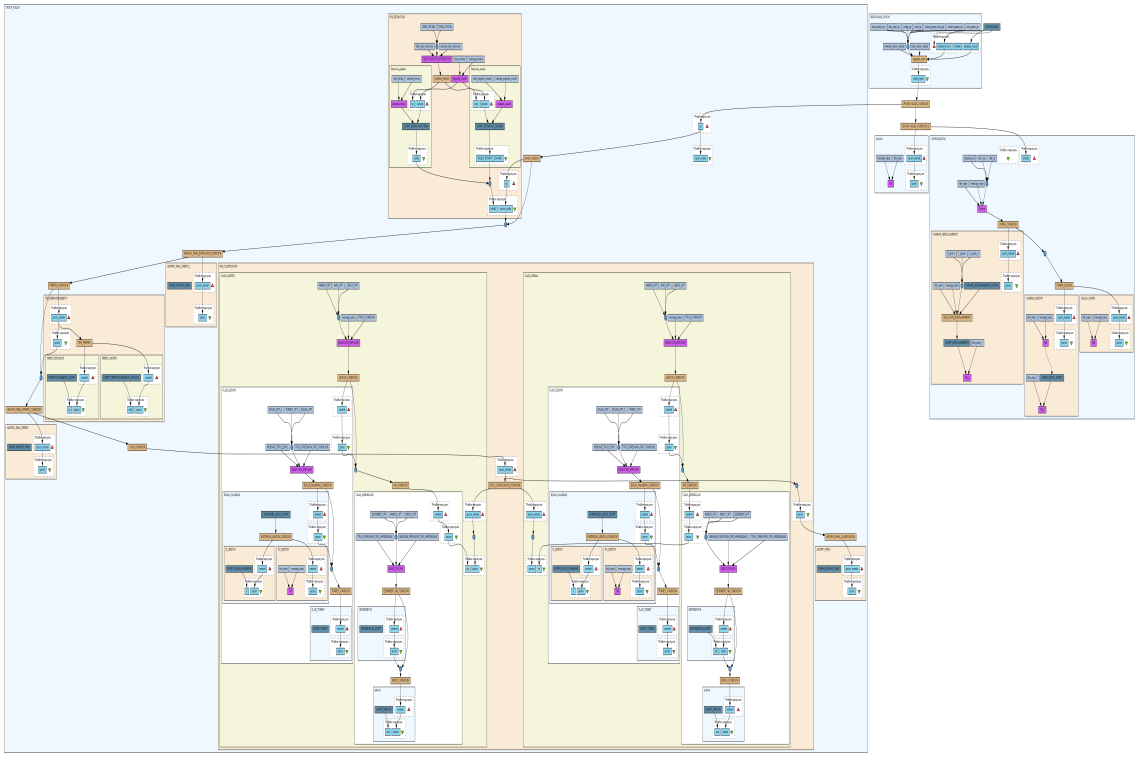


Figura A.22: Modelo del workflow desarrollado

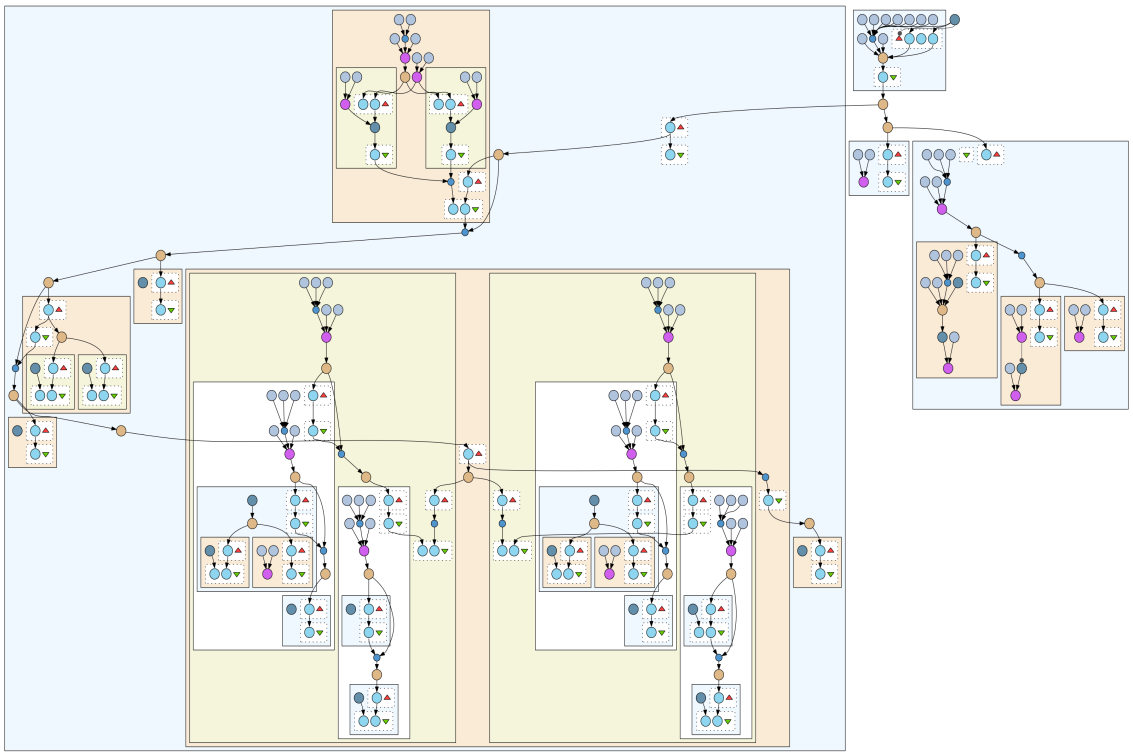


Figura A.23: Diagrama nodos del modelo y conexiones entre ellos

A.3. Pruebas de usabilidad

En las tablas A.1, A.2, A.3 y A.4, se muestran las apreciaciones para cada individuo que participó en las pruebas de usabilidad. Cabe destacar que todos los participantes poseen diversos conocimientos de informática, ya que se desempeñan en diversas actividades del área.

Pruebas de usabilidad					
Nivel de acuerdo	1	2	3	4	5
Utilizaría con frecuencia el programa				X	
Encontré el programa muy complejo	X				
Fue fácil utilizarlo					X
Necesitaria de un experto para utilizarlo	X				
Las diversas funciones están bien integradas					X
Hubo demasiado inconsistencia visual	X				
Lo encontré muy difícil de usar	X				
Las personas lo aprenderían a usar rapidamente			X		
Me sentí muy confiado en la nevegación			X		
Necesitaría aprender más antes de utilizarlo				X	

Cuadro A.1: Resultado pruebas de usabilidad individuo 1

Pruebas de usabilidad					
Nivel de acuerdo	1	2	3	4	5
Utilizaría con frecuencia el programa			X		
Encontré el programa muy complejo	X				
Fue fácil utilizarlo					X
Necesitaria de un experto para utilizarlo	X				
Las diversas funciones están bien integradas			X		
Hubo demasiado inconsistencia visual	X				
Lo encontré muy difícil de usar	X				
Las personas lo aprenderían a usar rapidamente					X
Me sentí muy confiado en la nevegación					X
Necesitaría aprender más antes de utilizarlo	X				

Cuadro A.2: Resultado pruebas de usabilidad individuo 2

Pruebas de usabilidad					
Nivel de acuerdo	1	2	3	4	5
Utilizaría con frecuencia el programa			X		
Encontré el programa muy complejo			X		
Fue fácil utilizarlo				X	
Necesitaria de un experto para utilizarlo			X		
Las diversas funciones están bien integradas				X	
Hubo demasiado inconsistencia visual			X		
Lo encontré muy difícil de usar			X		
Las personas lo aprenderían a usar rapidamente					X
Me sentí muy confiado en la nevegación			X		
Necesitaría aprender más antes de utilizarlo			X		

Cuadro A.3: Resultado pruebas de usabilidad individuo 3

Pruebas de usabilidad					
Nivel de acuerdo	1	2	3	4	5
Utilizaría con frecuencia el programa					X
Encontré el programa muy complejo		X			
Fue fácil utilizarlo		X			
Necesitaria de un experto para utilizarlo			X		
Las diversas funciones están bien integradas					X
Hubo demasiado inconsistencia visual	X				
Lo encontré muy difícil de usar		X			
Las personas lo aprenderían a usar rapidamente			X		
Me sentí muy confiado en la nevegación				X	
Necesitaría aprender más antes de utilizarlo	X				

Cuadro A.4: Resultado pruebas de usabilidad individuo 4