Escuela Politécnica Superior de Linares

Grado en Ingeniería Telemática

UNIVERSIDAD DE JAÉN

Escuela Politécnica Superior de Linares

HERRAMIENTA PARA LA EXTRACCIÓN DE EMOCIÓN DE UNA OPINIÓN TEXTUAL.

Alumno: **Rafael Liébana Cazalla**

Tutor: María Dolores Molina González

Depto.: Ingeniería de Telecomunicación

Índice

1. Resumen……8

2. Introducción……9
   2.1 Estructura del documento……10
   2.2 Objetivos……11

3. Estado del Arte……11
   3.1 Análisis de opiniones y su importancia……11
   3.2 Emociones básicas según Paul Ekman……14
   3.3 Herramientas para la extracción de la emoción……16

4. Diccionarios utilizados para la clasificación de las emociones……24
   4.1 NRC……24
   4.2 SEL……26
   4.3 iSAL……26

5. Herramientas para el desarrollo del software……28
   5.1 Eclipse……28
   5.2 Lenguaje de programación Python……29
      5.2.1 Librerías y módulos utilizados……30

6. Desarrollo de la herramienta……41
   6.1 Generación del lexicón……41
   6.2 Métodos de entrada……42
      6.2.1 Lectura de documentos……43
      6.2.2 Búsqueda de Tweets……44
      6.2.3 Búsqueda en página web……46
      6.2.4 Introducción manual del texto……48
   6.3 Pre-procesamiento del texto……50
   6.4 Método de búsqueda de las emociones en el texto……51
   6.5 Comparativas en métodos de búsqueda……52
   6.6 Desarrollo de la interfaz……52
   6.7 Pruebas del desarrollo del software……58
   6.8 Problemas encontrados en el desarrollo……63

7. Líneas futuras……64

8. Conclusión……65

9. Referencias bibliográficas……66

10. Anexos……69
   10.1 Manual de usuario……69
   10.2 Manual de instalación de herramienta de extracción de emociones……79

Índice de figuras.

Figura 0. Procesamiento del Lenguaje Natural……..9
Figura 1. Icono Miedo……14
Figura 2. Icono Tristeza……....14
Figura 3. Icono Alegría……..15
Figura 4. Icono Ira……..15
Figura 5. Icono Sorpresa……..15
Figura 6. Icono Asco……..16
Figura 7. Interfaz Linguakit……...18
Figura 8. Polaridad obtenida con Linguakit……18
Figura 9. Interfaz Bitext……19
Figura 10. Polaridad obtenida con Bitext……19
Figura 11. Interfaz Senpy ……19
Figura 12. Emociones obtenidas con Senpy……20
Figura 13. Interfaz Tone Analyzer……...22
Figura 14. Alegría obtenida con Tone Analyzer……23
Figura 15. Miedo obtenido con Tone Analyzer……..23
Figura 16. Tristeza obtenida con Tone Analyzer……..23
Figura 17. Ira obtenida con Tone Analyzer……24
Figura 18. Emociones obtenidas con Tone Analyzer……...24
Figura 19. Icono Eclipse……28
Figura 20. Icono Python……30
Figura 21. Word Tokenize……30
Figura 22. Stop Words……..31
Figura 23. Leer Stop Words……31
Figura 24. Stemmer…… 32
Figura 25. Ejemplo fondo. ……33
Figura 26. Ejemplo ImageTk…… 33
Figura 27. Miniatura botón…….. 33
Figura 28. Botón creado……34
Figura 29. Ejemplo gráfico……35
Figura 30. Gráfico de barras……35

Figura 31. Credenciales Twitter……...36
Figura 32. Ejemplo de uso de las credenciales……37
Figura 33. Autentificación twitter……..37
Figura 34. Comparativa API Twitter……38
Figura 35. Búsqueda hashtag……..39
Figura 36. Búsqueda en web……39
Figura 37. Apertura del navegador……..40
Figura 38. Beautiful Soup……41
Figura 39. Icono documentos……...43
Figura 40. Globo documentos……..43
Figura 41. Icono Twitter……44
Figura 42. Globo Twitter……44
Figura 43. Error no hashtag……..44
Figura 44. Previsualizar tweets……...45
Figura 45. Extracción tweets……45
Figura 46. Icono Internet……...46
Figura 47. Globo Internet……..46
Figura 48. Error no url……46
Figura 49. Error certificado……...47
Figura 50. Error web……..47
Figura 51. Previsualizar web……48
Figura 52. Información web……..48
Figura 53. Icono teclado……48
Figura 54. Globo teclado……...49
Figura 55. Error no mensaje escrito……49
Figura 56. Error ninguna emoción……...49
Figura 57. Pre-procesamiento……..51
Figura 58. Salida pre-procesamiento……..51
Figura 59. Etiqueta……54
Figura 60. Etiqueta fondo……..54
Figura 61. Boton Español……55
Figura 62. Ejemplo Ballon……55

Figura 63. Ejemplo Entry……..56
Figura 64. Ejemplo Text……56
Figura 65. Ejemplo ListBox……..57
Figura 66. Ejemplo Scroll……..57
Figura 67. Pantalla de bienvenida……...69
Figura 68. Botón de comienzo……69
Figura 69. Métodos de entrada……70
Figura 70. Entrada por archivo……70
Figura 71. Ejemplo Filedialog……...71
Figura 72. Método entrada búsqueda web……71
Figura 73. Método entrada búsqueda tweets……72
Figura 74. Método entrada teclado……72
Figura 75. Borrar cuadro escrito……..72
Figura 76. Previsualización del texto……..73
Figura 77. Emociones encontradas…… 73
Figura 78. Icono mostrar gráfico……..74
Figura 79. Globo mostrar gráfico……74
Figura 80. Gráfico generado……75
Figura 81. Icono mostrar comparativa……76
Figura 82. Globo mostrar comparativa……...76
Figura 83. Comparativa……76
Figura 84. Icono mostrar gráfico comparativo……...77
Figura 85. Globo mostrar gráfico comparativo……..77
Figura 86. Gráfico generado comparativa……..77
Figura 87. Icono mostrar palabra comparativa……..78
Figura 88. Globo mostrar palabras comparativa……...78
Figura 89. Palabras comparadas……78
Figura 90. Directorio Python……79
Figura 91. Directorio Scripts……79
Figura 92. Intalación NLTK……...80
Figura 93. Importar Word tokenize……..80
Figura 94. Importar Stop Words……..80

Figura 95. Importar Snowball Stemmer……81
Figura 96. Importar Beautiful Soup……81
Figura 97. Instalación Pillow……81
Figura 98. Importar Image……81
Figura 99. Importar ImageTK……81
Figura 100. Instalación Matplotlib……82
Figura 101. Importar Pyplot……82
Figura 102. Importar Numpy……83
Figura 103. Instalación Tweepy……83
Figura 104. Importar Tweepy……84

1. RESUMEN

El Trabajo Fin de Grado expuesto a lo largo de esta memoria detalla cómo ha sido el desarrollo de una Herramienta para la Extracción de las Emociones en un texto.
Para el cumplimiento de este objetivo principal, se estudiará en profundidad que es y cuáles son las utilidades del Procesamiento del Lenguaje Natural (PLN).
Se obtendrá toda la información relevante acerca de las emociones principales, se detallará las características de cada una y se conocerá como pueden ser estas diferenciadas en un texto.

Tras la obtención de estos conocimientos previos, se detallará el procedimiento para generar un lexicón propio, creado manualmente y con la finalidad de comparar este con los lexicones más utilizados en la actualidad.

Generado el lexicón propio, se mostrará el desarrollo de la herramienta junto a su interfaz. Incluyendo en ella, las palabras extraídas del texto y como han sido asignadas a cada una de las emociones principales. Estas se mostrarán diferenciadas por el lexicón por las que han sido extraídas, inclusive se mostrará cuál es la emoción principal que refleja el texto y la comparativa de esta con los principales lexicones.

Finalmente, se mostrarán posibles líneas futuras para este Trabajo Fin de Grado y cuáles han sido las conclusiones obtenidas tras su realización.

2. INTRODUCCIÓN

Vivimos en un mundo conectado, inmersos en la Era Digital donde la información y la tecnología juegan un papel crucial en el desarrollo de la sociedad. El uso de internet y los dispositivos electrónicos como teléfonos móviles o *tablets* siguen en aumento.

En el mundo, existen más de 3000 millones de usuarios que utilizan a diario las redes sociales, eso equivale al 42% del total de la población mundial. En España, los usuarios que utilizan las redes sociales a diario son unos 25,5 millones, es decir, la mitad de la población de España utiliza este medio de comunicación para comunicarse con sus más allegados o para crear sus relaciones personales. [34]

Hoy día, no solo ha cambiado la forma en la que nos comunicamos o creamos nuestras relaciones personales, inclusive ha cambiado la forma en la que nos mantenemos informados, pasando a ser parte preferente las redes sociales o la prensa online. El 20% de la población utiliza las redes sociales como medio principal de información, a diferencia de un 16% que aún elije la prensa escrita. Por lo que, sería conveniente verificar que la información que obtenemos de medios online es subjetiva y que no se pretenden inculcar unos ideales o unos pensamientos previamente elaborados. [35]

Además, debido a los altos valores de usabilidad manejados por las redes sociales y los medios de comunicación online, sería conveniente poder extraer información relevante de los millones de datos que son generados a diario gracias al alto uso de internet. Por ejemplo, se podrían obtener conductas sexistas o raciales, así como pensamientos depresivos o ideales terroristas. Puntos importantísimos con los cuales no solo se consigue mejorar el uso de internet, si no lo más importante, se pueden salvar vidas.

Es por esto, que es necesario adentrarnos en el conocimiento del Procesamiento del Lenguaje Natural (PLN) e incluir en el desarrollo de este Trabajo Fin de Grado los diferentes campos que se incluyen en esta disciplina.

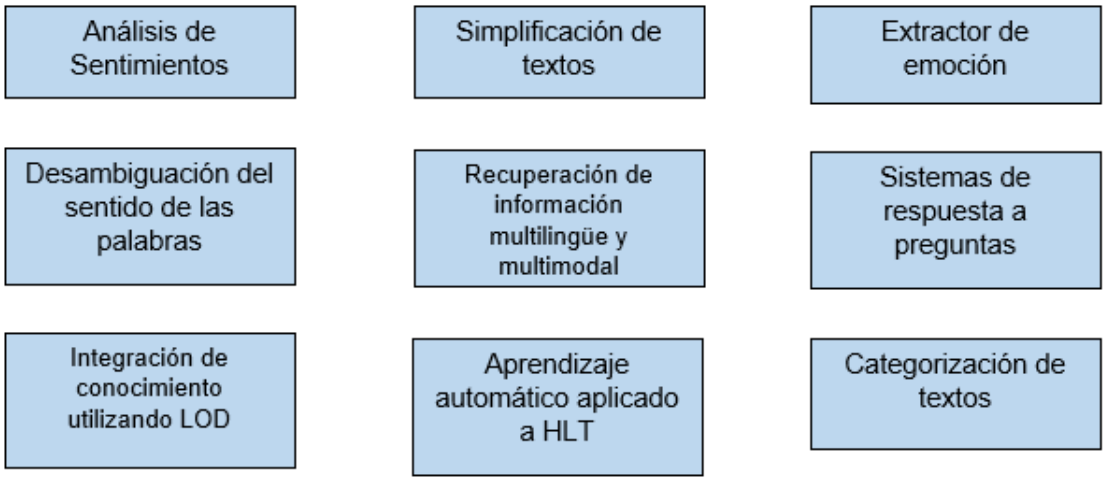


Figura 0. Procesamiento del Lenguaje Natural (PLN)
Fuente: Elaboración Propia.

En la Figura 0, podemos observar las principales tareas del PLN, nombraremos cada una de ellas en secciones posteriores y conoceremos en profundidad que es la Minería de Opiniones (o Análisis del Sentimiento) y porqué es la parte más importante en el desarrollo de este Trabajo Fin de Grado.

2.1 Estructura del documento

Este Trabajo Fin de Grado se encuentra dividido en cinco grandes bloques, en un primer bloque, vamos a estudiar en que consiste el Procesamiento del Lenguaje Natural (PLN) y cuáles son sus principales tareas. Detallaremos, dentro de estas, las asociadas a la Minería de Opiniones (o también conocido como Análisis del Sentimiento) descubriremos las diferentes variantes que en ella se encuentran y mostraremos como gracias a ellas seremos capaces de extraer la polaridad o las emociones asociadas a un texto.

En un segundo bloque, conoceremos quién es el autor que definió los seis rangos principales en los que podemos dividir las emociones, se detallarán como pueden ser encontradas estas en un texto e incluso detallaremos como es el rostro facial que se muestra con cada una de ellas. Continuaremos mostrando algunos ejemplos de herramientas comercializadas para la minería de opiniones, se introducirá un texto de prueba en cada una de ellas y se mostrará cuáles son las funcionalidades y limitaciones que presentan.

En un tercer bloque, se estudiarán cuáles son los lexicones más utilizados en la actualidad, se mostrará cómo ha sido el desarrollo de cada uno de ellos y cuál es la extensión por la que cada uno está compuesto. Seguidamente, se generará un lexicón propio, de forma manual, basándonos en los métodos de creación de los lexicones mostrados anteriormente. Se detallará la finalidad de la creación de este lexicón, así como la cantidad de palabras que lo componen.

En un cuarto bloque, se desarrollará la herramienta, describiendo cuales han sido los módulos utilizados para su implementación, se mostrará un ejemplo de utilización para cada uno de ellos, así como la implementación para su uso. Se detallará cada uno de los caminos de navegación que esta permite, y los diferentes errores y avisos que se muestran si su uso no es el correcto. Finalmente, se describen cuáles han sido las pruebas realizadas para la verificación del correcto funcionamiento de la herramienta.

Para concluir este Trabajo Fin de Grado, se propondrán líneas futuras para su continuación y futura actualización y se muestra cuáles han sido las conclusiones obtenidas tras su finalización.

2.2 Objetivos

Los objetivos docentes para la superación de este Trabajo de Fin de Grado son los siguientes:

- Estudiar en que consiste el Análisis de Opiniones y su importancia.
- Realizar un estudio de diferentes herramientas que ayuden a la extracción de la emoción.
- Desarrollar una herramienta para poder extraer la emoción de una opinión textual.
- Redactar una memoria

Además de estos objetivos, se ha propuesto un objetivo personal tras la finalización de la herramienta, el cual ha sido la generación de un lexicón propio, de pequeñas dimensiones con el que obtener unos valores óptimos y parecidos a los que obtendríamos con unos lexicones más extensos, estos lexicones serán estudiados en secciones posteriores.

3 ESTADO DEL ARTE

3.1 Análisis de opiniones y su importancia

En un primer momento, nos centraremos en definir que es el Procesamiento del Lenguaje Natural, tal y como se define en Vicomtech, “El procesamiento del lenguaje natural (PLN) es el campo que combina las tecnologías de la ciencia computacional (como la inteligencia artificial, el aprendizaje automático o la inferencia estadística) con la lingüística aplicada, con el objetivo de hacer posible la comprensión y el procesamiento asistidos por ordenador de información expresada en lenguaje humano para determinadas tareas, como la traducción automática, los sistemas de diálogo interactivos, el análisis de opiniones, etc.” [40]

Las principales tareas que podemos realizar con el PLN, entre muchas otras, son las siguientes:

- Análisis de opiniones (o conocido por sus diferentes variantes en inglés como *Sentiment Classification*, *Sentiment Analysis* o *Opinion Mining*).
- Extractor de información.
- Categorización de textos.
- Simplificación de textos.
- Recuperación de información multilingüe y multimodal.
- Aprendizaje automático aplicado a *Human Learning Technology* (HLT)
- Integración de conocimiento utilizando *Linked Open Data* (LOD)
- Sistemas de respuesta a preguntas.
- Desambiguación del sentido de las palabras.

Nosotros nos centraremos en definir la primera tarea nombrada anteriormente, esta es el análisis de opiniones o también conocida como minería de opiniones. Es de vital importancia conocer esta utilidad para el correcto desarrollo de este Trabajo Fin de Grado, ya que será ampliamente utilizada para la extracción de emociones en un texto.
La tarea del análisis de opiniones engloba al conjunto de herramientas o técnicas dentro del PLN con las que podemos extraer los sentimientos que se encuentran en un texto y para ello será necesario conocer cuáles son las emociones principales que se encuentran

en un texto, como se categorizan cada una de ellas y como seremos capaces de diferenciarlas unas de otras. Una vez mostrado esto, será necesario utilizar alguna de las herramientas que se incluyen en la minería de opiniones para categorizar el texto según nuestras necesidades.

Conoceremos a fondo dos tipos de herramientas que se incluyen dentro de la minería de opiniones y que son necesarias estudiarlas ya que serán las utilizadas para el desarrollo de la herramienta. Estas son:

- Herramientas basadas en la detección de la polaridad: estas son capaces de determinar si un texto es de carácter positivo o negativo. Para ello se analiza cada una de las palabras y verifica si un ámbito general el texto influye una opinión negativa o positiva.

- Herramientas basadas en el análisis de emoción: son capaces de determinar si alguna de las emociones básicas, o todas ellas, se encuentran en el texto.
  Para ello, analizan cada palabra y verifican si estas se encuentran asociadas alguna emoción. Finalmente, con las palabras encontradas, verifican cuál es la emoción mayoritaria que muestra el texto.
  Estas herramientas suelen ser mucho más complejas que las anteriores y necesitan más tiempo computacional para llevar a cabo la lectura del texto y la extracción en caso de que la hubiera de la emoción a la que se encuentra asociada cada palabra.

Además, ambas herramientas tienen como finalidad la subjetividad del texto, ya que la clasificación de las emociones o de la polaridad de un texto es cada vez más compleja, debido a que incluso para cada autor estas difieren. La interpretación de un texto es diferente para cada persona, ya que se ven influenciadas por muchos factores, ya sean por diferencias culturales, motivos personales o incluso la experiencia personal de cada autor.

Conocidas cuáles son las dos herramientas posibles para el desarrollo de esta tarea, debemos saber, además, cuales son los dos enfoques principales, ya que como se ha comentado, es necesario que la herramienta conozca que palabras tienen un ámbito positivo o negativo, para la detección de la polaridad, o cuáles son las palabras asociadas a cada emoción, para la detección de los sentimientos. Por lo que, estos enfoques se detallan a continuación:

- Los enfoques semánticos, los cuales se caracterizan por la necesidad de incluir un diccionario o un lexicón para su uso. Estas bases de datos léxicas pueden ser más o menos complejas. Así podemos encontrarnos con diccionarios que incluyen definiciones, sinónimos, antónimos, polaridad, emociones o incluso a veces intensidad de esa polaridad o emoción. Por otro lado, nos podemos encontrar lexicones muy simples basados en listas de palabras con polaridad positiva y negativa.

- Los enfoques basados en tratamiento computacional (más conocido como *Machine Learning*): como su propio nombre indica, la finalidad de este enfoque es que la herramienta aprenda de forma automática. Con esto nos referimos a que la herramienta va mejorando y afinando de forma automática la salida con su uso.

En estos casos se le introduce a la herramienta unos datos de entrenamiento etiquetados, con los cuales se le intenta enseñar como son generadas las frases en el idioma proporcionado, así como las características que tiene cada uno de los datos introducidos. A partir de aquí la herramienta será capaz de reconocer cuales son las características que debe extraer y vaya obteniendo mejores resultados según va realizando pruebas. Finalmente, será capaz de extraer la polaridad o los sentimientos asociados al texto sin la necesidad de introducirle diccionarios como al enfoque semántico.

Conociendo las diferentes herramientas para el análisis de opiniones, así como los tipos de enfoques que se pueden utilizar, vamos a detallar el que se ha elegido para realizar este trabajo. La herramienta elegida para este Trabajo Fin de Grado ha sido una herramienta para la extracción del sentimiento, ya que se nos propone extraer las emociones que refleja un texto y no solo la polaridad. Además, se utilizará un enfoque semántico incluyéndole a este un lexicón propio.

El principal problema que presenta la herramienta elegida, es derivado a la propia semántica del lenguaje, ya que al incluir en ella la necesidad de introducirle un diccionario o un lexicón, sería necesario incluir todas las variantes posibles. Por lo que, la dificultad añadida a esta herramienta es cuando en un texto se muestra ironía, donde textualmente se escribe un conjunto de palabas, pero se requiere una interpretación diferente de estas.

Ejemplo de ironía:
Hoy ha sido un día perfecto: mi ordenador se ha actualizado en el momento idóneo.

Como podemos observar, en esta frase encontramos las palabras “perfecto” e “idóneo”. En una primera instancia, podemos pensar que el autor de esta frase mostraba un sentimiento de felicidad por las palabras que había utilizado, pero la realidad no es esta. En este ejemplo se muestra una desconformidad utilizando palabras que en un ámbito general se utilizan para mostrar felicidad.
Debido a esto, las herramientas para el análisis de opiniones son más complejas de lo que en un primer momento pueden llegar a parecer, ya que tenemos que tener en cuenta que un texto puede estar expresado irónicamente.

En este trabajo obviaremos la posibilidad de que un texto muestre ironía, ya que, de no ser así, la herramienta sería mucho más extensa y necesitaría unos requerimientos computacionales más complejos de los objetivos propios de este trabajo.
Se muestra a continuación un ejemplo de una opinión clara, en el que las palabras utilizadas reflejan la opinión por la que son utilizadas en un ámbito general.

Ejemplo:
Mi ordenador tiene poca capacidad de almacenamiento, aunque consta de una buena memoria *RAM y una tarjeta gráfica inmejorable.*

Se puede observar que en ejemplo mostrado encontramos las palabras “poca” que tiene una valoración negativa y las palabras “buena” e “inmejorable” que tienen una valoración positiva.
Por lo que, podríamos definir que la oración tiene una valoración positiva, aunque se encuentren palabras que expresen una valoración negativa, generalmente la oración muestra una valoración positiva hacia un producto recientemente adquirido.

En este ejemplo, tan solo nos centraríamos en detectar la polaridad del texto, para mostrar un ejemplo de la extracción de sentimientos vamos a utilizar el siguiente ejemplo:

Ejemplo:
Hoy estoy muy feliz, ha sido un día perfecto. He obtenido un premio a mi expediente y he firmado un contrato con una buena empresa.

En el ejemplo, podemos encontrar las palabras “feliz”, “perfecto”, “premio”, “contrato”. Todas ellas asociadas a la emoción alegría, por lo que sin lugar a dudas se puede determinar que el autor de la oración muestra un sentimiento de alegría.

3.2 Emociones básicas según Paul Ekman

Paul Ekman es un psicólogo estadounidense nacido en Washington en 1934. Es pionero en el estudio de las emociones y de las expresiones faciales. Además, está entre los cien psicólogos más destacados del siglo XX. [4]
En 1972, fue cuando Ekman definió las seis emociones básicas: ira, asco, miedo, alegría, tristeza y sorpresa.

Nos centraremos en definir cada una de ellas además de incluir cómo podemos encontrarlas en cada uno de nuestros textos.

Figura 1. Icono Miedo.
Fuente: www.freepik.com

Se define el miedo como aquello que sentimos cuando nos encontramos bajo una amenaza. No todo el mundo percibe las amenazas de igual forma, ya que estas pueden ser reales o algo que hemos ido generando con nuestro desarrollo. Por lo que, en un ámbito general, se define el miedo como aquello que nos incomoda o nos hace sentir inseguridad.
Según Ekman, la expresión facial asociada al miedo es “la elevación de las cejas y de los párpados superiores, con la retracción de los labios y con la tensión de los párpados inferiores”. [5]

Figura 2. Icono Tristeza.
Fuente: www.freepik.com

Sentimos tristeza cuando perdemos algo valioso en nuestra vida, ya sea material o meramente personal. También sentimos tristeza al no conseguir un objetivo profesional o personal.
Según Ekman, la expresión facial asociada a la tristeza es el “descenso de los párpados superiores y de los extremos de los labios. También se observa una menor focalización de la mirada en el punto de atención.” [5]

Figura 3. Icono Alegría.
Fuente: www.freepik.com

Se define la alegría como la satisfacción plena debida a un desarrollo personal o por la obtención de un reto propuesto. Podemos diferenciar la alegría de la felicidad por su duración, ya que se conoce que la felicidad es un estado más prolongado en el tiempo mientras que la alegría es definida en un intervalo corto del tiempo.
Según Ekman, la expresión facial asociada a la alegría es “la elevación de las mejillas y la aparición de arrugas en la comisura de los ojos.” [5]

Figura 4. Icono Ira.
Fuente: www.freepik.com

Mostramos ira al estar en desacuerdo con una situación generada a nuestro alrededor o en la sociedad en general. La ira se muestra en forma de enfado.
Según Ekman, la expresión facial asociada a la ira es “cuando nuestras cejas se acercan y descienden, mientras que los labios se aprietan”. [5]

Figura 5. Icono Sorpresa.
Fuente: www.freepik.com

Se muestra sorpresa cuando ocurre un suceso que no esperamos o cuando ocurre una actividad momentánea de la cual no estábamos avisados. Generalmente, la sorpresa no se considera una emoción positiva ni negativa, a diferencia de las demás que si se clasifican de esta forma.
Según Ekman, la expresión facial asociada a la sorpresa es “la apertura de la boca y de los ojos junto con la elevación de la musculatura asociada a las cejas”. [5]

Figura 6. Icono Asco.
Fuente: www.freepik.com

Mostramos asco cuando algo nos resulta desagradable, ya sean objetos, animales o incluso acciones realizadas por otra persona.
Según Ekman, la expresión facial asociada al asco se asemeja mucho a la de la ira, ya que para ambas “arrugar la nariz o levantar el labio superior son indicadores tanto de lo que denominamos asco como de lo que denominamos ira”. [5]

3.3 Herramientas para la extracción de la emoción

Como se ha detallado en el apartado 3.1 “Análisis de Opiniones y su Importancia”, podemos diferenciar claramente distintos tipos de herramientas para la minería de opinión.
En un primer lugar se encuentran las herramientas capaces extraer la polaridad de un texto, diferenciando en estas tan solo si el texto es positivo o negativo.
En un segundo lugar, se encuentran las herramientas capaces de extraer las emociones asociadas a un texto, siendo estas semejantes a la herramienta desarrollada en este trabajo.

Aunque se encuentran muchos más tipos de herramientas, como aquellas que extraen entidades, la simplificación de textos o el sistema de respuesta a preguntas, nosotros nos vamos a centrar en estas dos modalidades ya que son las que resultan más importantes en el desarrollo y conocimiento de este Trabajo Fin de Grado.

Para la correcta visualización de ambas, tomaremos como ejemplo dos herramientas para cada una de ellas, exponiendo un uso de utilización como ejemplo.
Primero vamos a exponer los ejemplos de herramientas extractoras de la polarización, para ello tomaremos de ejemplo las herramientas Linguakit y Bitext.
Para mostrar un ejemplo de las herramientas extractoras de emociones vamos a exponer las herramientas Senpy y Tone Analyzer.

Además, se utilizará un texto de prueba de generación propia en todas las herramientas, para una correcta visualización de la salida proporcionada por cada herramienta individualmente y a modo de comparativa entre ellas.

El texto de ejemplo es el siguiente:

Que alegría y que chiste lo que me cuentas, ya que me encuentro muy contento, con mucha energía ya que esto es un entretenimiento, la verdad es que en este bullicio es como una jarana.
Que miedo que crueldad y que alarma ya que esto es de cobardía, parece un apocalipsis y que viene el demonio, aunque también tengo emoción y desengaño.
Tengo un cabreo, un rencor y una rabia que no puedo con ella. Este enfado no se me olvida ya que me encuentro de muy malhumor.
Esto me parece una indigestión y repulsión ya que tengo un empacho y un disgusto muy grande, es que me dan arcadas.
Cuidado con la alarma ya que puede ser un atraco, o un asalto. También puede ser que haya una colisión o un choque, ya puede diferir el concepto de la bofetada.
He tenido un desmayo de la angustia por tu contrariedad, eso ha sido un dolor muy grande y no es dramatismo, siento aspereza.

Herramientas extractoras de polaridad.

a) Linguakit

Linguakit es fruto de la empresa gallega Cilenis Language Technology, tras años de desarrollo y estudios en el área del Procesamiento del Lenguaje Natural, PLN.
La web permite hasta 20 consultas a su herramienta de forma gratuita, una vez cumplidas estas podemos obtener su versión completa

Es una web con múltiples características y funciones, entre ellas podemos encontrar la extracción de palabras claves de un texto, así como el análisis de sentimientos o incluso hasta una función para realizar un resumen de los textos introducidos.
La función que nosotros vamos a utilizar a modo de ejemplo para la extracción de la polarización es la llamada “Analizador de sentimiento”.
Esta función permite 3 métodos de entrada; entrada manual, entrada mediante una dirección web o búsqueda de un archivo en nuestros directorios.
A modo de ejemplo realizaremos una prueba mediante la entrada manual, para ello introduciremos un texto de generación propia en el que se encuentran todas las emociones.

Ejemplo de la interfaz de Linguakit:

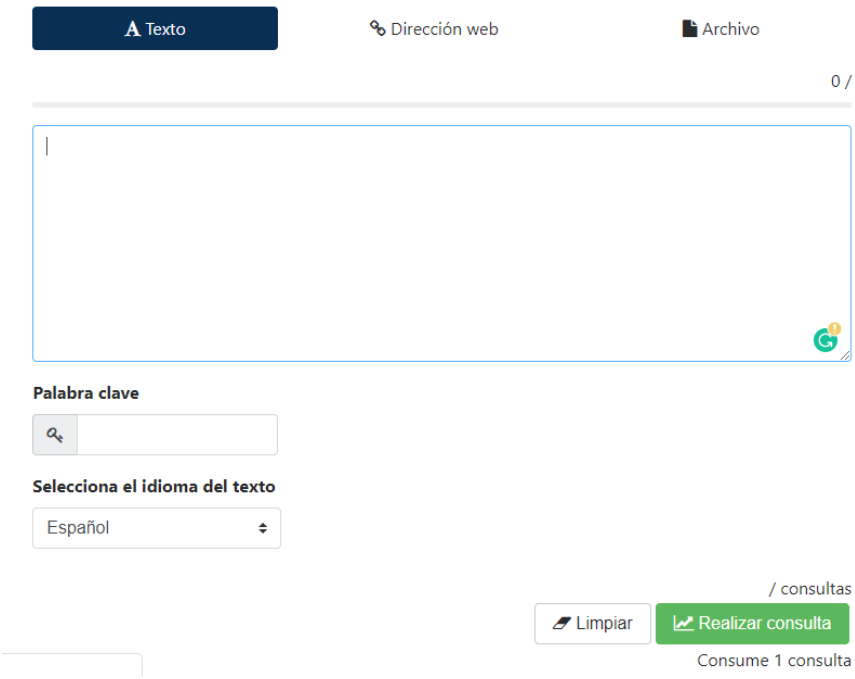


Figura 7. Interfaz linguakit
Fuente: https://linguakit.com

Salida de la herramienta al texto introducido:

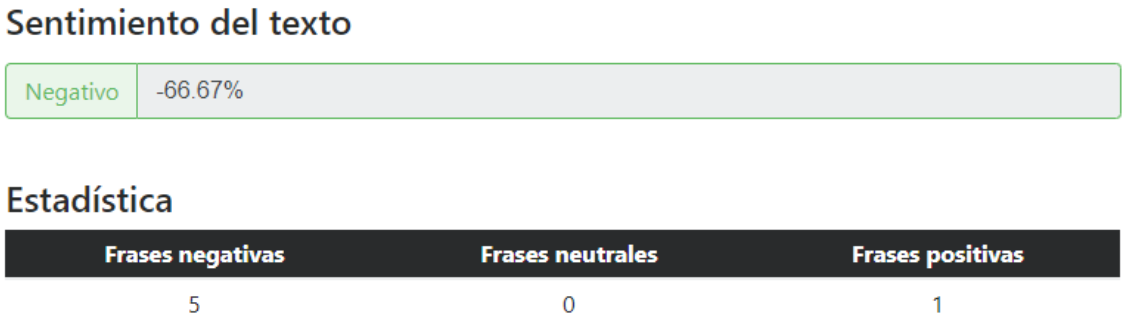


Figura 8. Polaridad obtenida con Linguakit.
Fuente: https://linguakit.com

La salida que muestra la herramienta se puede encontrar en la Figura 8, por lo que podemos concluir que el texto muestra una polaridad negativa, ya que esta ha obtenido un 66,67 %.

b) Bitext

Bitext ha sido desarrollado por la empresa Bitext Innovations SL, la cual tiene su sede principal en Madrid. Se encuentran especializados en más de 80 lenguajes y variantes de estos, siendo conocidos por su amplia variedad en diccionarios y gramáticas.

Es un software especializado en el Procesamiento del Lenguaje Natural, PLN.
En el podemos encontrar tres áreas principales:

- Bots para chats y asistentes virtuales.
- Servicios NLP, tales como la lematización.
-Análisis de sentimientos.

La funcionalidad de su software que vamos a utilizar a modo de ejemplo es el "Sentiment Analysis", está se puede utilizar mediante un periodo de prueba de 4 días además de que solo se pueden incluir 400 caracteres en la entrada manual.
Esta funcionalidad tan solo permite un método de entrada, este es mediante entrada manual del texto, aunque si permite dos tipos de salida, ellos son modo; gráfico o json.

Como anteriormente, vamos a realizar la prueba introduciendo un texto de generación propia en el cual se incluyen todas las emociones, la salida de la herramienta hacia este texto es la siguiente:

Ejemplo de la interfaz de la herramienta:

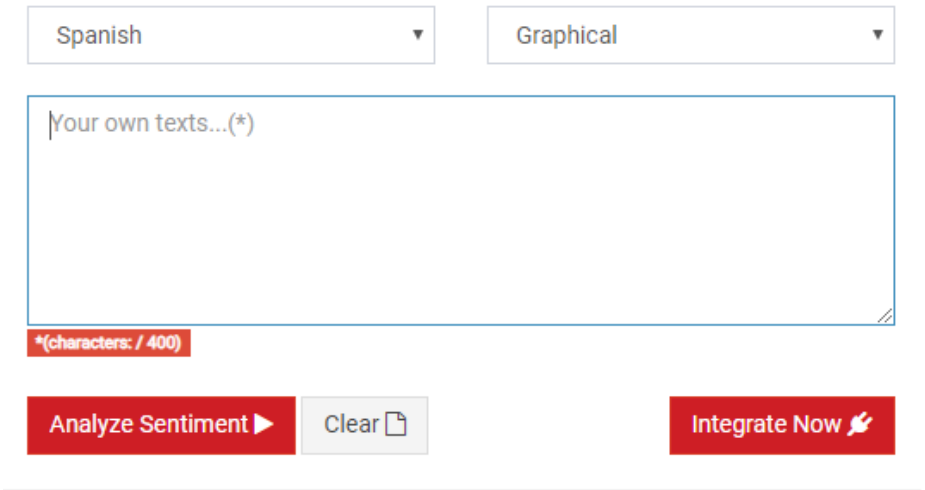


Figura 9. Interfaz Bitext.
Fuente: https://api.bitext.com

Ejemplo de la salida de la herramienta:

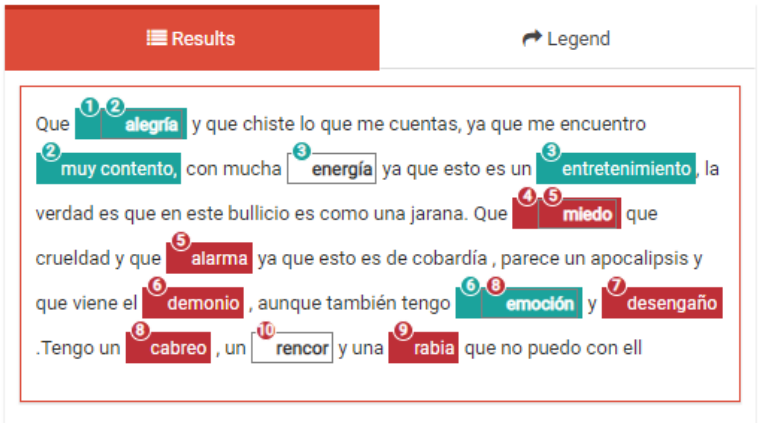


Figura 10. Polaridad obtenida con Bitext.
Fuente: https://api.bitext.com

Se puede observar que la herramienta ha detectado que la mayoría de palabras son de carácter negativo, por lo tanto, se puede determinar que, en un ámbito general, el texto muestra una polaridad negativa.

Herramientas extractoras de emociones.

a) *Senpy*

Senpy se encuentra dentro del Mixed Emotions Project. En este proyecto, además de este software se encuentran muchos otros, como pueden ser Audio Analysis, Emotion Lexicon o Entity Linking.

Es un framework especializado en el análisis de sentimientos y el procesamiento del lenguaje natural. Principalmente se encuentra especializado en la interconexión de todos sus servicios.
Entre todas estas funcionalidades, se comparte una interfaz común, todas ellas se encuentran disponibles de forma libre y gratuita en Github.
Senpy incluye 3 tipos de entrada a modo de ejemplo, en caso de que queramos realizar la prueba sin introducir un texto propio. Estos tres métodos de entrada son:

- Tweet Regular
- Tweet Político
- Extracto de un artículo de noticias.

La salida se muestra mediante un Json.

Se va a realizar una prueba mediante la entrada manual, introduciendo el mismo texto que se ha utilizado en las anteriores herramientas.

Ejemplo de la interfaz de la herramienta:

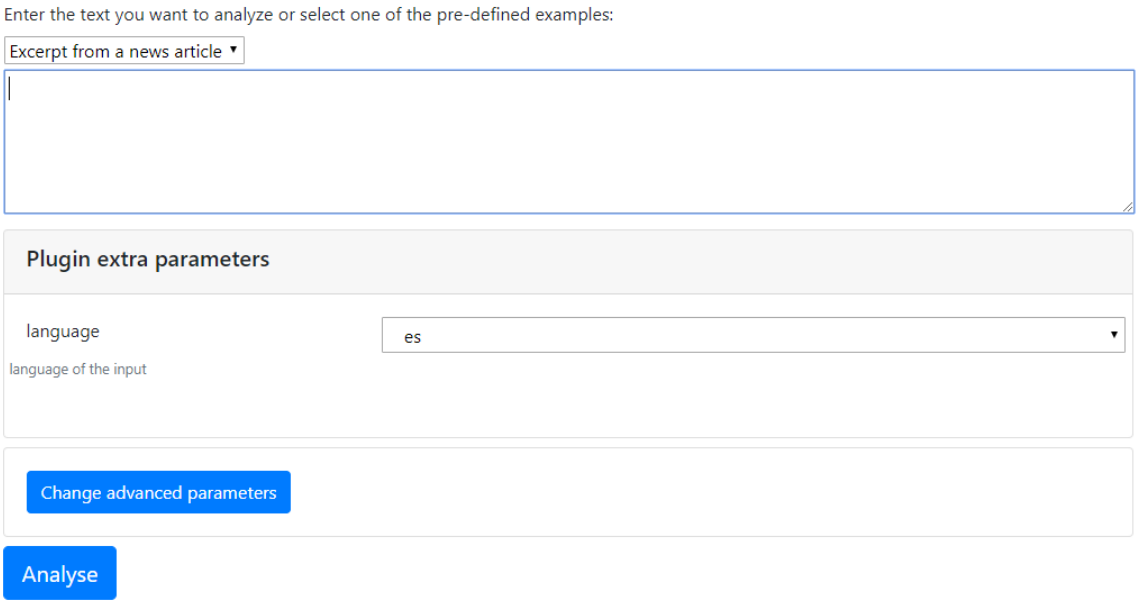


Figura 11. Ejemplo interfaz Senpy.
Fuente: https://senpy.readthedocs.io

Ejemplo de la salida de la herramienta:


```
  "@type":  "Emotion",
  "onyx:hasEmotionCategory":  "wna:negative-fear",
  "onyx:hasEmotionIntensity":  0.07932197996786643
},
{
  "@type":  "Emotion",
  "onyx:hasEmotionCategory":  "wna:amusement",
  "onyx:hasEmotionIntensity":  0.16269030441947765
},
{
  "@type":  "Emotion",
  "onyx:hasEmotionCategory":  "wna:anger",
  "onyx:hasEmotionIntensity":  0.11394466353829155
},
{
  "@type":  "Emotion",
  "onyx:hasEmotionCategory":  "wna:annoyance",
  "onyx:hasEmotionIntensity":  0.13933803391889452
},
{
  "@type":  "Emotion",
  "onyx:hasEmotionCategory":  "wna:indifference",
  "onyx:hasEmotionIntensity":  0.13025930535952843
},
{
  "@type":  "Emotion",
  "onyx:hasEmotionCategory":  "wna:joy",
  "onyx:hasEmotionIntensity":  0.10699460397065558
},
{
  "@type":  "Emotion",
  "onyx:hasEmotionCategory":  "wna:awe",
  "onyx:hasEmotionIntensity":  0.1417332794905439
},
{
  "@type":  "Emotion",
  "onyx:hasEmotionCategory":  "wna:sadness",
  "onyx:hasEmotionIntensity":  0.12571782933474204
}
```


Figura 12. Emociones obtenidas con Senpy.
Fuente: https://senpy.readthedocs.io

Tras el uso de la herramienta se encuentra los siguientes porcentajes asociados a las emociones:

- Miedo: 7,93 %
- Ira: 11,39 %
- Alegría: 10,69 %
- Tristeza: 12,57 %
- Miedo: 14,17 %
- Entretenimiento: 16,26 %
- Indiferencia: 13,02 %
- Molestia: 13,93 %

b) Tone Analyzer

Tone Analyzer es propiedad de IBM y se encuentra incluida en su programa Watson. IBM es una conocida multinacional, con sede principal en Nueva York y es una de las principales marcas reconocidas a nivel mundial en todo el ámbito de la tecnología.

Esta es una herramienta para el análisis de texto con el que se pueden extraer las siguientes emociones; alegría, miedo, tristeza e ira. La herramienta solo se puede utilizar en un formato de prueba, ya que para la completa utilización de esta es necesario obtener el paquete completo.

El módulo de la herramienta que vamos a utilizar incluye tres tipos de ejemplo de usos, entre los que podemos seleccionar, Tweets, Review de un producto e incluso emails. Además de estos tres métodos de entrada de prueba se permite la introducción de un texto propio, por lo que, a modo de prueba, introduciremos un texto propio para realizar la comparativa con las otras herramientas que hemos utilizado.
Debemos realizar un pequeño cambio para la introducción de nuestro texto propio, ya que la herramienta solo permite textos en inglés o en francés, por lo tanto, en un primer momento realizaremos la traducción del texto y después lo introduciremos para visualizar las emociones que en él se encuentran.

Además, en esta herramienta se van a introducir diferentes textos de prueba, cada uno de ellos mostrando una emoción, para una correcta visualización de la salida.
Finalmente, se introducirá el texto correspondiente a todas las emociones para la verificación de que todas ellas las detecta.

Ejemplo de la interfaz de la herramienta:

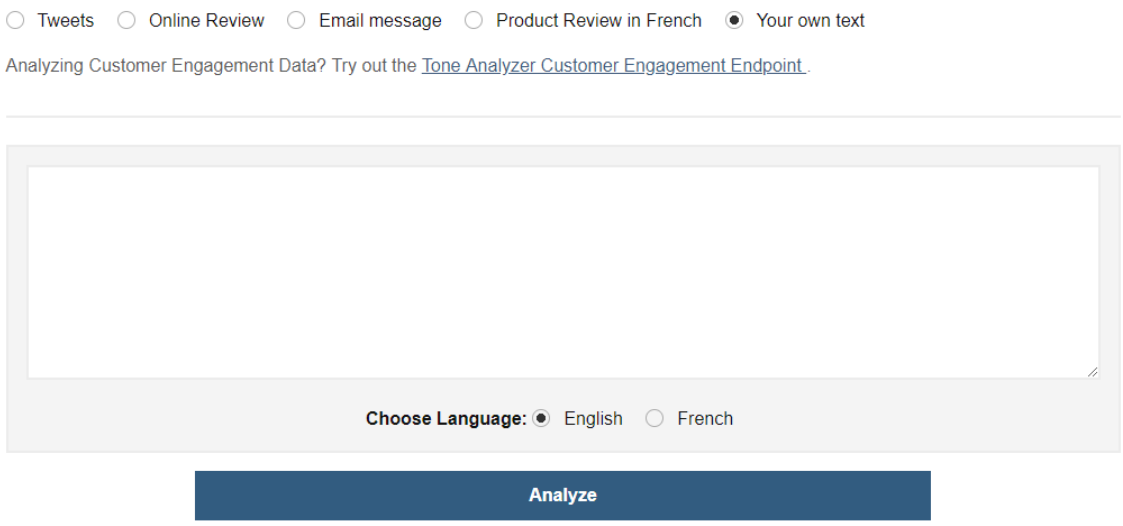


Figura 13. Alegría obtenida con Tone Analyzer.
Fuente: https://tone-analyzer-demo.ng.bluemix.net/

Texto de prueba introducido que contiene palabras de alegría:

Que alegría y que chiste lo que me cuentas, ya que me encuentro muy contento, con mucha energía ya que esto es un entretenimiento, la verdad es que en este bullicio es como una jarana.

Salida de la herramienta hacia este texto:

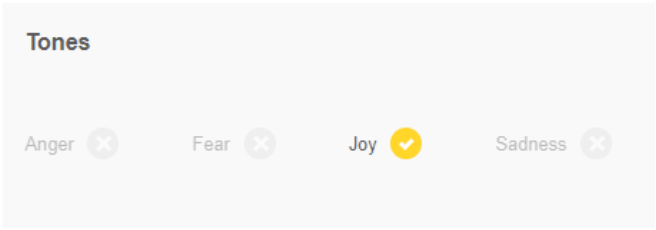


Figura 14. Alegría obtenida con Tone Analyzer.
Fuente: https://tone-analyzer-demo.ng.bluemix.net/

Texto de prueba introducido que contiene palabras de miedo:

Que miedo que crueldad y que alarma ya que esto es de cobardía, parece un apocalipsis y que viene el demonio, aunque también tengo emoción y desengaño.

Salida de la herramienta hacia este texto:

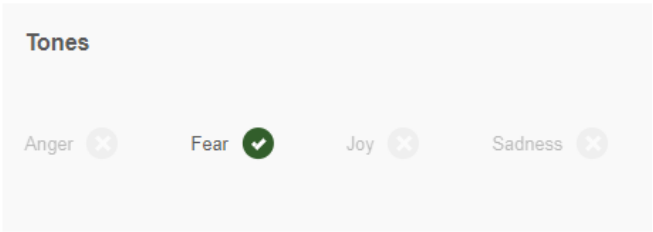


Figura 15. Miedo obtenido con Tone Analyzer.
Fuente: https://tone-analyzer-demo.ng.bluemix.net/

Texto de prueba introducido que contiene palabras de tristeza:

He tenido un desmayo de la angustia por tu contrariedad, eso ha sido un dolor muy grande y no es dramatismo, siento aspereza.

Salida de la herramienta hacia este texto:

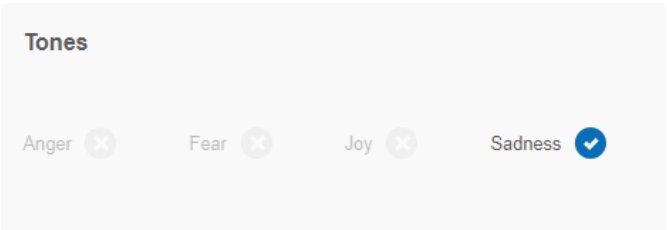


Figura 16. Tristeza obtenida con Tone Analyzer.
Fuente: https://tone-analyzer-demo.ng.bluemix.net/

Texto de prueba introducido que contiene palabras de ira:

Tengo un cabreo, un rencor y una rabia que no puedo con ella. Este enfado no se me olvida ya que me encuentro de muy malhumor.

Salida de la herramienta hacia este texto:

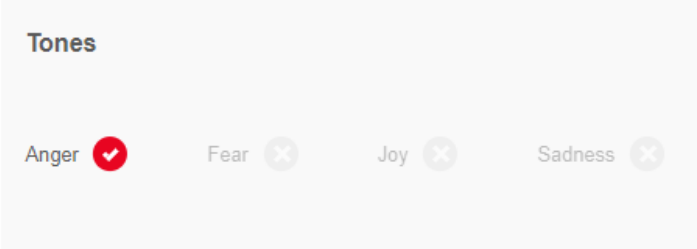


Figura 17. Ira obtenida con Tone Analyzer.
Fuente: https://tone-analyzer-demo.ng.bluemix.net/

Texto de prueba introducido que contiene las emociones, ira, miedo, alegría y tristeza:

Que alegría y que chiste lo que me cuentas, ya que me encuentro muy contento, con mucha energía ya que esto es un entretenimiento, la verdad es que en este bullicio es como una jarana.
Que miedo que crueldad y que alarma ya que esto es de cobardía, parece un apocalipsis y que viene el demonio, aunque también tengo emoción y desengaño.
He tenido un desmayo de la angustia por tu contrariedad, eso ha sido un dolor muy grande y no es dramatismo, siento aspereza.
Tengo un cabreo, un rencor y una rabia que no puedo con ella. Este enfado no se me olvida ya que me encuentro de muy malhumor.

Salida de la herramienta hacia este texto de prueba:

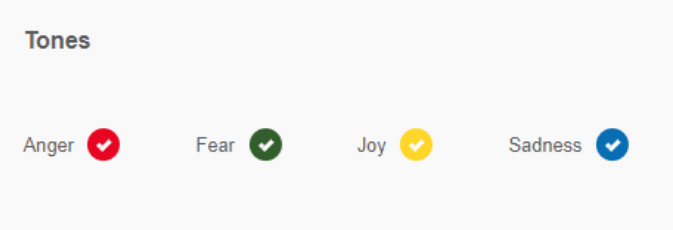


Figura 18. Emociones obtenidas con Tone Analyzer.
Fuente: https://tone-analyzer-demo.ng.bluemix.net/

4 DICCIONARIOS UTILIZADOS PARA LA CLASIFICACIÓN DE EMOCIONES.

4.1 NRC

El diccionario NRC ha sido desarrollado por el Doctor Saif M. Mohammad, el cuál es un investigador científico senior en el National Research Council Canada (NRC). Además, fue investigador en el Instituto de Estudios Avanzados en Informática de la

Universidad de Maryland. Sus campos de estudio son la lingüística computacional y el procesamiento del lenguaje natural. Este ha publicado más de 80 artículos científicos, incluido el Léxico Emotivo NRC. [36]

El Léxico Emotivo NRC se compone por casi unas 6000 palabras en inglés, las cuales están asociadas a las emociones básicas ira, miedo, alegría, asco, sorpresa y tristeza. Los términos adheridos a este diccionario han sido obtenidos mediante búsquedas en tweets, así como otros términos comúnmente utilizados en redes sociales. Tal y como declara Saif "Twitter tiene una base de usuarios grande y diversa, lo que conlleva un rico contenido textual".
Se han seguido los siguientes pasos para generar el diccionario:

- La palabra encontrada es conocida por estar asociada a una emoción.
- La palabra tiene tendencia a aparecer en tweets que expresan una emoción.

Por lo tanto, conociendo esto, los términos se han ido anotando en dos fuentes separadas:

- Las palabras que están marcadas como asociadas a una emoción.
- Las palabras que tienden a coexistir más a menudo en tweets con hashtags con la palabra emoción tales como (#ira, #miedo, #alegría, #tristeza, #asco, #sorpresa)

Una vez generado el diccionario, se anota la intensidad de emoción que presenta cada palabra, siendo estos 4 niveles diferentes. Así las palabras se pueden asociar con diferentes grados de una emoción. Por ejemplo, según define Saif, "la palabra condenar se asocia con un mayor grado de Ira que la palabra irritar. El objetivo de esta tarea es determinar los grados de ira asociados con las palabras, ya que es difícil dar una puntuación numérica que indique el grado de enojo."

Como ya hemos comentado, para diferentes personas puede tener diferentes signos de emoción una palabra, por lo tanto, un diccionario no siempre va a ser totalmente cierto o totalmente válido, así que se realizan bastantes pruebas para que este pueda ser utilizado de una forma correcta.

Para este trabajo, se han utilizado las palabras de NRC en español, ya que en las últimas versiones del léxico NRC se han incluido las traducciones literales de todas las palabras que lo componía, y se ha generado un diccionario con las siguientes combinaciones:

Palabras alegría: 661
Palabras asco: 1001
Palabras ira: 1187
Palabras miedo: 1395
Palabras sorpresa: 511
Palabras tristeza: 1123
Total de palabras utilizadas con este diccionario = 5878

Aunque el diccionario en un principio fue desarrollado en inglés, se ha obtenido que, salvo algunas diferencias culturales, las normas afectivas son iguales en todos los idiomas, y las traducciones para las diferentes versiones han sido bajo la utilización de

Google Translate. Si es verdad que, hay que verificar una a una estas traducciones ya que algunas no son correctas al ser traducidas literalmente de los términos en inglés.

4.2 SEL

Este lexicón ha sido generado por el Dr Grigori Sidorov, el cual imparte clase en el Instituto Politécnico Nacional de la Ciudad de México, además de pertenecer al Centro de Investigación en Computación y al Laboratorio de Procesamiento del Lenguaje Natural.
Además de esto, es también un miembro regular de la Academia de las Ciencias de México. [14]

El lexicón está compuesto por 2036 palabras, agrupadas en las seis emociones principales; alegría, miedo, tristeza, sorpresa, ira y asco.
La creación de este lexicón se realizó gracias a diecinueve personas que anotaron su percepción hacia cada palabra, para esto debían además anotar en que escala se encontraba esta palabra hacia esa emoción, la escala posible era nulo, bajo, medio o alto.

Tras la realización de una tabla que mostrará la percepción que había obtenido cada persona hacia cada palabra de la lista, se realizó una media de entre todas, mostrando así una escala general y unas puntuaciones generales para cada palabra.
Además, también se creó el Factor de probabilidad de uso afectivo (PFA), con el cual el total de este factor será 1, que se refiere a que el 100 % de los anotadores relacionaban la palabra hacia el valor de la escala “alto” hacia una emoción, y con un 0 si el 100 % de los anotadores lo relacionaban con el valor “nulo”.
De esta forma podemos determinar que a un valor más alto de PFA se asegura que es más probable que la palabra presente la emoción a la que se ha asignado.

Por lo que, tras conocer cómo se han ido incluyendo cada una de las palabras en su emoción correspondiente, incluimos el número de palabras que se ha generado para cada emoción y que, por lo tanto, han sido utilizadas en esta herramienta:

Palabras alegría: 668
Palabras asco: 209
Palabras ira: 382
Palabras miedo: 211
Palabras sorpresa: 175
Palabras tristeza: 391
Total de palabras utilizadas con este diccionario = 2036

4.3 iSAL

Este lexicón ha sido desarrollado por el grupo de investigación de Sistemas Inteligentes de Acceso a la Información (SINAI) de la Universidad de Jaén. [37]

El lexicón está generado por 5016 palabras, las cuales se encuentran en cuatro grupos según la emoción que presentan, estas son; alegría, ira, miedo y tristeza.

La finalidad de generar este lexicón era la de adaptar al español el lexicón NRC, para de esta forma, poder ser utilizado en herramientas para la minería de opiniones en este idioma. Para esto, se han utilizado dos estrategias principales, estas son tanto la traducción de forma automática gracias a herramientas externas, como las revisiones manuales.

Tras una primera evaluación de las traducciones generadas del lexicón NRC, se encontraron los siguientes problemas:

- Había 1267 términos repetidos.
- 110 palabras sin traducción en español.
- 45 expresiones sin traducción.
- 326 n-gramas generados en español de términos en inglés.

Para solucionar estos problemas principales, se han ido generando diferentes versiones del lexicón, siendo estas:

- iSALv1a: donde se eliminaban los términos repetidos, quedando tras esta eliminación un lexicón con 2338 palabras asociadas a una sola emoción.

- iSALv2a: en esta segunda versión, se mejoró el problema derivado de la repetición de términos. Además, se incluyeron los términos repetidos que estaban asociados a varias emociones, siendo estos 2053, formando un lexicón de 4391 términos.

- iSALv3a: en la tercera versión, se siguió mejorando el problema relacionado con la repetición de palabras, donde se tuvo en cuenta las palabras repetidas que se encontraban con diferentes puntuaciones asociadas a una emoción. Por lo que, se realizó una media de estas puntuaciones y se agregó un solo término común. Estos forman 764 términos, quedando un lexicón con 5155 palabras.

Tras solucionar el problema con las palabras repetidas, se propuso la meta de solucionar los problemas con las palabras sin traducir, siendo estas traducidas manualmente según la interpretación que se obtenía en español, por lo que se desarrollaron las siguientes versiones.

- iSALv1m: de las 110 palabras sin traducción que se encontraron en la primera versión del lexicón, tan solo 33 pudieron ser correctamente traducidas manualmente, ya que las demás no correspondían a ningún termino en español que las definiera completamente.
  De las 45 expresiones sin traducción encontradas, 40 fueron traducidas eficazmente.
  Para los n-gramas encontrados, se realizó una búsqueda para traducirlos por una sola palabra en español, por lo que de los 266 encontrados, 237 fueron traducidos.
  Formando el nuevo lexicón v1m, que estaría formado por 2227 términos.

- iSALv2m: en esta segunda versión, se mejora el problema derivado de los n-gramas.

Por lo que, finalmente en esta versión se añadieron los términos de las traducciones manuales de los n-gramas, 2031 términos, que sumados a los ya generamos en la versión v1m, quedaría un lexicón con 4258 palabras.

- iSALv3m: en esta versión final, se terminó de mejorar el problema derivado de los n-gramas, generando 754 nuevos términos de los n-gramas que se habían encontrados repetidos en varias emociones. Quedando finalmente un lexicón de 5012 palabras.

Finalmente se generó un lexicón de 5012 palabras, las cuales se encuentran divididas de la siguiente forma en las cuatro emociones principales:

Palabras alegría: 1072
Palabras ira: 1308
Palabras miedo: 1528
Palabras tristeza: 1104
Total de palabras utilizadas con este diccionario = 5016

En un primer momento, este lexicón fue utilizado a modo de comparativa en el desarrollo de la herramienta, pero al incluir esta un diccionario en el que se incluyen las seis emociones principales según Ekman, los datos obtenidos de las comparativas no eran totalmente visuales, ya que se comparaba un diccionario con seis rangos y otro con cuatro, por lo que finalmente se optó por detallar todo el procedimiento para la generación de este lexicón pero no utilizarlo en la comparativa de la herramienta.

5 HERRAMIENTAS PARA EL DESARROLLO DEL SOFTWARE

5.1 Eclipse

Eclipse es una herramienta para el desarrollo de software, generalmente usado para desarrollar aplicaciones escritas en el lenguaje “Java”, aunque gracias a su instalador de paquetes puede ser usado para el desarrollo de otros lenguajes de programación.

Figura 19. Icono Eclipse
Fuente: www.eclipse.org

Aunque en un primer momento, Eclipse pertenecía a la multinacional IBM, en la actualidad se encuentra en desarrollo por la Fundación Eclipse. Esta es una organización sin ánimo de lucro con la única finalidad que la generación de herramientas de código abierto. [38]

Eclipse incorpora módulos que podemos ir incluyendo según lo necesitemos, para esto utilizaremos el entorno de desarrollo integrado (IDE).
Gracias a este mecanismo de módulos, Eclipse no es un software demasiado pesado, a comparación de otras herramientas de desarrollo, que incluyen todas las funcionalidades en el paquete principal.
Como ya hemos comentado, Eclipse puede extenderse para el uso de otros lenguajes de programación como C o Python.

Para entornos gráficos, eclipse proporcional un *framework* bastante rico en funcionalidades, pero tan solo utilizable en Java. Si deseamos construir una interfaz gráfica para otro de los lenguajes necesitaremos de la instalación de los módulos propios para cada lenguaje.
Además, una de las herramientas más potentes que ofrece eclipse es el *debugeador*, con el cual podemos realizar el análisis de nuestro código paso a paso, conociendo el valor de las variables en todo momento para encontrar los errores que hayamos cometido o para conocer a fondo como funciona nuestro código en el nivel de ejecución.

5.2 Lenguaje de programación Python

Python se desarrolló a finales de los 80 y su creador fue Guido van Rossum.
La principal característica de Python es que es un lenguaje multiparadigma, con el cuál puedes utilizar una programación orientada a objetos, una programación funcional o una programación imperativa. [39]

Además, de que es un lenguaje que se caracteriza por su fácil legibilidad, su creador intentó que fuera tan fácil de leer que alguien que entendiera el inglés pudiera conocer el funcionamiento del código sin conocer exactamente como este había sido implementado. Entre las características más importantes de Python también se encuentra la resolución dinámica de nombre, ya que enlaza un método y un nombre de variable durante la ejecución del programa.

Principales elementos del lenguaje.

a) Elementos lógicos.

En Python no encontramos lo símbolos que en otros lenguajes de programación se utilizan, ya pueden ser estos *¡, ||, &, &&.* Ya que en este lenguaje se detallan como la palabra inglesa que describen estos símbolos, como *and, or o not.*

b) Tabuladores.

En la gran mayoría de lenguajes de programación, la apertura y cierre de funciones, así como las sentencias *if* y *for*, se realiza mediante los símbolos conocidos como llaves {}
Mientras que en Python se utiliza la tabulación o indexación, para mostrar donde empieza y acaba cada función, o las sentencias ya definidas anteriormente.

c) Variables.

En Python no es necesario definir con anterioridad a su uso las variables que después se van a utilizar, ya que estas pueden ser definidas de forma dinámica, es decir,

se definen en el momento de la ejecución, además, tampoco es necesario especificar a qué tipo pertenece la variable que vamos a utilizar.

Figura 20. Icono Python.
Fuente: https://www.python.org/

5.2.1 Librerías y módulos utilizados

5.2.1.1 NLTK

5.2.1.1.1 Word tokenize

Utilizado para dividir los componentes de un *string*, para de esta forma, separar las palabras que lo componen y poner analizarlas individualmente.
Este módulo ha sido de gran utilidad, ya que gracias a él se pueden buscar comparaciones de una manera sencilla en una lista. Podemos llegar a pensar que también se habría podido separar por palabras una lista de una forma más sencilla, el problema que se presentó y el cual este módulo ha solucionado se mostrará en el siguiente ejemplo.

Ejemplo:
Este coche, que me compré ayer, es muy bonito
.

Podemos observar que la siguiente frase es fácilmente separable por palabras, y se podrían haber utilizado infinidad de métodos. La dificultad viene incluida en las palabras acompañadas de un signo de puntuación, como en este caso son coche, ayer, y bonito.
Al utilizar diferentes módulos incluidos en Python estos no detectaban que la palabra viniera con un signo de puntuación, ya que mostraba que “ayer,”, “bonito.” y “coche,” eran una palabra en sí, por lo tanto, al compararlo con las auténticas palabras incluidas en la base de datos, la salida de la comparación era negativa, siendo esto totalmente falso.
A continuación, se muestra un ejemplo de la utilización de este módulo:

```
listaLeidaTokenizada=word_tokenize(listaLeida,'spanish')
```

Figura 21. Word Tokenize.
Fuente: Elaboración propia.

Como se muestra, necesitamos introducirle dos parámetros de entrada a este método, estos son la lista que deseamos tokenizar y el idioma utilizado en esta lista. Podemos llegar a preguntarnos cuál es la finalidad de incluir el idioma, ya que no debería de ser necesario. Pero sí que lo es, ya que en cada idioma se utilizan diferentes signos de puntuación, así como diferentes reglas ortográficas, y este método las tiene en cuenta.

5.2.1.1.2 Stopwords

Este módulo contiene todas las palabras de parada o también conocidas como palabras vacías, se refiere a aquellas palabras que no reflejan ninguna emoción o sentimiento y, por lo tanto, pueden ser obviadas en la minería de opiniones. Un ejemplo de estas palabras pueden ser los artículos o incluso las preposiciones.
Es de vital importancia en el uso de nuestra herramienta, ya que, incluyendo ese módulo, conseguimos que la herramienta sea más eficaz.
Puede ser utilizado en varios idiomas, la utilización de este módulo en nuestro caso ha sido en español.

Para su correcto uso debemos tener instalada la librería NLTK y además incluir en nuestro código una implementación de esta base de palabras, para finalmente realizar una comparativa del texto leído y buscar si en él se encuentran estas palabras vacías.
El uso de este módulo en nuestro código se ha realizado de la siguiente manera:

```
self.stop_words=set(stopwords.words('spanish'))
```

Figura 22. Stop Words.
Fuente: Elaboración propia.

En este primer paso estamos incluyendo en la variable *stop_words* toda la lista de palabras vacías.

```
listaLeidaStopWords=[w for w in listaLeidaPuntuacion if not w in self.stop_words]
```

Figura 23. Leer Stop Words.
Fuente: Elaboración propia.

Finalmente, se realiza una búsqueda en la lista guardada para filtrar todas las palabras vacías que en ella pudiéramos encontrar.

5.2.1.1.3 SnoballStemmer

Utilizado para lematizar cada una de las palabras encontradas. Este es uno de los módulos más importantes utilizados en el desarrollo de la herramienta, ya que, gracias a él, el diccionario propio creado para su uso ha sido disminuido considerablemente.

El uso de este módulo es el siguiente, tenemos la palabra ‘contento’, que al introducirla en el lematizador obtendríamos “content”. De esta forma, si tenemos las palabras “contento”, “contentos”, “contenta”, “contentas”, tan solo necesitaríamos incluir la palabra contento en nuestro diccionario, ya que al lematizar todas las palabras expuestas obtendríamos una sola variante, que sería “content”, de esta forma disminuimos en una gran cantidad nuestro diccionario.

A continuación, se muestra un ejemplo de la utilización de este código:

```
StemmerEspanol=SnowballStemmer('spanish')
listaStemmerAlegria.append(StemmerEspanol.stem(textoAbuscarAlegria))
```

Figura 24. Stemmer.
Fuente: Elaboración propia.

Con la primera línea de código, estamos creando el lematizador, con esto nos referimos a que vamos a guardar en esta variable el método en cuestión, en el cual debemos indicar cuál es el idioma de las palabras que vamos a lematizar, ya que el módulo necesita conocer cuáles son las reglas ortográficas y gramaticales que se utilizarán en nuestras palabras.

Con la segunda línea de código realizamos varios pasos en uno, podemos observar que utilizamos dos métodos, estos son *stem* y *append*.
El primero se ha utilizado para lematizar la palabra que se encuentra alojada en la variable *textoAbuscarAlegria* y con el segundo introducimos la palabra ya lematizada dentro de la lista *listaStemmerAlegría*.

5.2.1.2 Python Imaging Library (PIL)

La biblioteca de imágenes de Python (PIL) es utilizada para agregar la capacidad de procesamiento de imágenes en Python.
Con ella, somos capaces de leer un amplio rango de tipos de archivos de imagen, así como procesarlos de una forma eficiente.
Esta librería incluye infinidad de módulos tanto para la lectura como para la escritura de imágenes.

Uno de los módulos más potentes es *ImageSequence*, con el cual podemos crear un GIF desde una imagen corriente. Con este módulo podemos darle movimiento a una imagen estática. Pero en la aplicación desarrollada en este trabajo nos vamos a centrar tan solo en dos módulos también muy importantes, ellos son *Image* e *ImageTk*, los cuales se detallarán a continuación.

a) *Image.*

Entre las funciones incluidas en este módulo se encuentra la de cargar imágenes de archivos y crear nuevas imágenes.
Además, también puede ser utilizado para visualizar las imágenes cargadas o creadas e incluso rotarlas.

También se pueden crear miniaturas de las imágenes, que es para lo que ha sido utilizado en esta herramienta. A continuación, se muestra un ejemplo del código utilizado para cargar las imágenes deseadas, así como redimensionarla para ser utilizada como una miniatura o un icono dentro de la aplicación.

```
imagenFondoPrincipal = Image.open("imagenes/fondoAplicacion.png")
imgFondoPrincipal = imagenFondoPrincipal.resize((1000,600), Image.ANTIALIAS)
```

Figura 25. Ejemplo fondo.
Fuente: Elaboración propia.

Como podemos observar, gracias al método *open* cargamos la imagen, para ello debemos introducir el directorio donde se encuentra y la extensión de la imagen en cuestión.
Con el método *resize*, cambiamos el tamaño de la imagen para adaptarla a nuestra aplicación y así crear la miniatura deseada.

b) ImageTk

Este módulo es capaz de modificar los objetos anteriormente creados o leídos con el módulo *Image*. La función que he utilizado con este método es *PhotoImage*, con la cuál he añadido las imágenes de fondo y también se ha incluido las miniaturas anteriormente creadas a los botones diseñados en esta aplicación.

A continuación, se va a mostrar el código utilizado en la herramienta para detallar este módulo:

```
self.photoImgFondoPrincipal =  ImageTk.PhotoImage(imgFondoPrincipal)
iconoFondoPrincipal=ttk.Label(self.raiz,image=self.photoImgFondoPrincipal)
iconoFondoPrincipal.place(x=0,y=0,relwidth=1,relheight=1)
```

Figura 26. Ejemplo ImageTk.
Fuente: Elaboración propia.

En la primera línea de este código, guardamos en la variable *photoImgFondoPrincipal* la imagen anteriormente leída, y de esta forma leerla como una cadena de bits.
Con la segunda línea creamos una etiqueta común y añadimos la imagen a esta etiqueta. De esta forma, ya tenemos una miniatura añadida a una etiqueta, pero esta no es la finalidad del código.

La principal finalidad es añadir una imagen al fondo de nuestra herramienta, por lo que utilizamos *place*, para ubicar la etiqueta anteriormente creada al fondo de la aplicación. De esta forma podemos seguir añadiendo elementos encima de nuestra imagen de fondo y simular el comportamiento de una aplicación real.

Pero como he detallado anteriormente, esta no ha sido la única finalidad del método. La segunda finalidad para lo que ha sido utiliado este método ha sido el generar botones con miniaturas. Vamos a detallar el código utilizado en este caso.

```
photoImgBotonComenzar =  ImageTk.PhotoImage(imgBotonComenzar)
botonEspanol=ttk.Button(self.raiz,image=photoImgBotonComenzar,command=lambda:metodoEntrada(self))
botonEspanol.place(y=480,x=150)
```

Figura 27. Miniatura botón.
Fuente: Elaboración propia.

Como anteriormente se ha detallado, en la primera línea guardamos en la variable que después utilizaremos la imagen creada con anterioridad.
Con la segunda línea de este código de ejemplo, creamos el botón a utilizar e incluimos el evento que este abrirá al ser presionado.

Se muestra el aspecto final de botón creado.

Figura 28. Botón creado.
Fuente: Elaboración propia

5.2.1.3 Matplotlib

Es la librería que incorpora Python para la generación de gráficas en 2D. Entre sus muchas funciones podemos encontrar la creación de histogramas, gráficos de barras, diagramas de dispersión, espectros de potencia, gráficos, etc.
Estamos familiarizamos con el estilo de estos gráficos, ya que son muy similares a los que podemos crear con Matlab, pero esta librería nos facilita mucho la implementación de estas, ya que solo es necesarias unas pocas líneas para la generación completa del gráfico.

a) *Pyplot (PLT)*
Pyplot es un módulo incluido en la librería *Matplotlib*.

Gracias a este módulo los estilos de los gráficos creados se asemejan mucho a los utilizados en Matlab, como se ha detallado anteriormente. Con esto se quiere decir, que cada gráfico es creado como una figura, en la que podemos editar todos los parámetros que en ella se encuentra, ya sea tamaño y estilo de los ejes, tamaño y estilo de las barras o incluso la utilización de un zoom dentro del gráfico. Además, añade otras funciones muy interesantes, como puede ser el guardado del gráfico en un archivo *JPEG*.

A continuación, se muestra el uso que se le ha dado a este módulo dentro de nuestra herramienta:

```
grupos=6
datos=[self.totalPorcentajeAlegria/100,self.totalPorcentajeAsco/100,self.totalPorcentajeIra/100,
       self.totalPorcentajeMiedo/100,self.totalPorcentajeSorpresa/100,self.totalPorcentajeTristeza/100]
figuras, barras = plt.subplots()
index = np.arange(grupos)
anchuraBarra = 0.35
opacidad = 0.8
muestraColor = {'ecolor': '0.8'}
rects1 = barras.bar(index, datos, anchuraBarra,alpha=opacidad, color='b', error_kw=muestraColor)
barras.set_xlabel("Emociones")
barras.set_ylabel('Porcentaje Obtenido')
barras.set_title('Resultado de las emociones obtenidas')
barras.set_xticks(index + anchuraBarra / 2)
barras.set_xticklabels(('Alegría', 'Asco', 'Ira', 'Miedo', 'Sorpresa','Tristeza'))
barras.legend()

figuras.tight_layout()
plt.show()
```

Figura 29. Ejemplo gráfico.
Fuente: Elaboración propia

Como se puede observar, se ha detallado la cantidad de grupos que encontraremos en el gráfico (que en este caso serán las diferentes emociones encontradas), se ha utilizado el método *subplots* para dibujar cada una de estas barras, indicando además el color, anchura y opacidad del color utilizado. Se incluye además el nombre de cada uno de los ejes, así como la leyenda del gráfico, y finalmente se utiliza el método *show* para generar el gráfico completo.
Un ejemplo del gráfico creado con el anterior código es el siguiente:

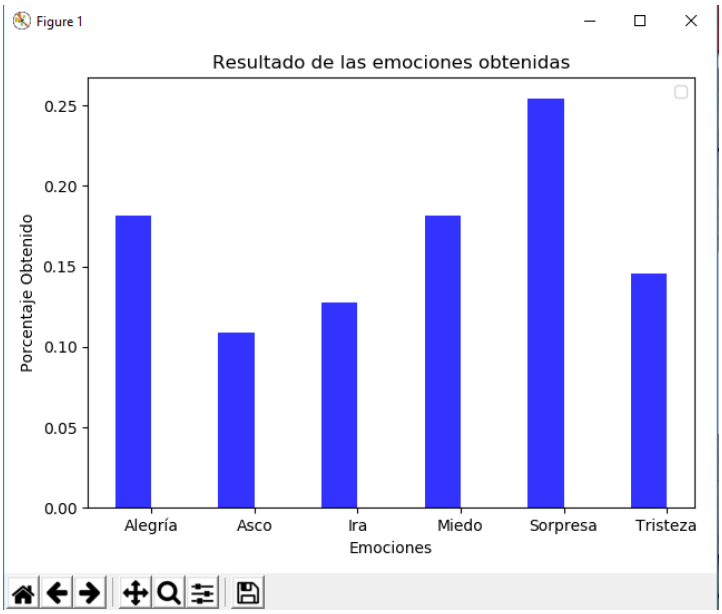


Figura 30. Gráfico de barras.
Fuente: Elaboración propia.

Se observa el enorme parecido al gráfico generado con los ya utilizados en *Matlab*, además se puede comprobar que se han añadido distintas funciones al gráfico de forma automática.

Estas funciones son la de zoom, guardado, modificación del tamaño del gráfico, configuración a tiempo real de los distintos valores asignados a los ejes, además de movimiento del gráfico dentro de la ventana.

5.2.1.4 Tweepy

Para el correcto uso de la API de Twitter, sería necesario un primer paso que es el de crear una cuenta de desarrollador en su plataforma. Twitter es una empresa muy extensa y debe conocer que desarrolladores van a hacer uso de sus servidores.

En este apartado nos centraremos en el uso de la API en sí, suponiendo que ya somos desarrolladores y que conocemos nuestras credenciales para su uso.
Una vez registrados como desarrolladores, debemos crear un proyecto para el cual vamos a utilizar las credenciales de acceso, este ha sido titulado “Trabajo Fin de Grado”.
Debemos contestar varias preguntas que Twitter nos hace antes de facilitarnos nuestras credenciales, pero esto se detallará con detenimiento en el manual.

Una vez registrados como desarrollador y creado el proyecto se nos facilitarán las credenciales de conexión.
Las credenciales tendrán esta forma, y es el primer paso necesario para la conexión y obtención de Tweets, funcionalidad que utilizaremos de esta API.
En el apartado *Credentials* de la Web de desarrolladores de Twitter aparecerán nuestras credenciales en esta forma:

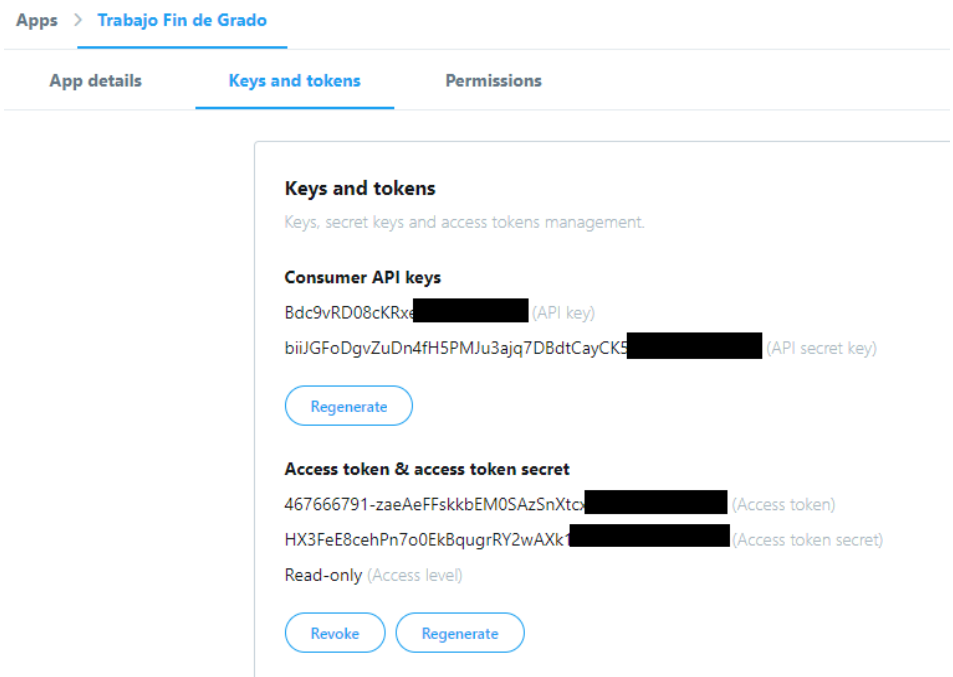


Figura 31. Credenciales Twitter.
Fuente: https://developer.twitter.com/

Por seguridad no he expuesto la totalidad de mis claves de acceso, ya que se podría dar un mal uso de estas.

Una vez que conocemos nuestras credenciales, necesitamos incluirlas estas como una variable en nuestro código, de la siguiente forma:

```
consumer_key = 'Bdc9vRD08cKRxe[redacted]
consumer_secret = 'biiJGFoDgvZuDn4fH5PMJu3ajq7DBdtCay[redacted]
access_key = '467666791-zaeAeFFskkbEM0SAzSnXtcx4C[redacted]
access_secret = 'HX3FeE8cehPn7o0EkBqugrRY2wAXk1TU7Bc[redacted]
```

Figura 32. Ejemplo de uso de las credenciales.
Fuente: Elaboración propia.

Guardadas correctamente las credenciales en las variables especificadas para ellas, necesitamos autentificarnos con estas en la API de Twitter, para ello, están los métodos *OAuthHandler* y *set_access_token*.

Estos métodos debemos implementarlos como se muestra a continuación:

```
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_key, access_secret)
```

Figura 33. Autentificación twitter.
Fuente: Elaboración propia.

Realizados estos pasos, si no nos aparece ningún error, estaremos autentificados correctamente como desarrollador en la API y seremos capaces de realizar cualquier función de las que esta ofrece.
Vamos a detallar las más importantes, centrándonos en la de obtención de Tweets que ha sido la utilizada en esta herramienta.

a) Filtrado de Tweets en tiempo real.

Como su propio nombre indica, con esta funcionalidad podemos visualizar e incluso guardar Tweets a tiempo real. Esta es una funcionalidad muy potente ya que permite que obtengamos información al momento de que algo suceda.
Con ella seremos capaces de detallar que están expresando las personas conectadas a Twitter en este mismo instante.

En un primer momento, esta funcionalidad era la que se iba a incluir en el desarrollo de la herramienta, pero presenta un gran hándicap y es que al leer a tiempo real los Tweets, con el paso del tiempo, las variables donde se almacenan los Tweets ya leídos y los que se están leyendo al momento es inmensa, la herramienta era incapaz de procesar todos estos datos.

Por lo que finalmente se desechó está función, quedando prevista para futuras actualizaciones de la herramienta como mejora de esta.

b) Búsqueda de Tweets.

Con esta funcionalidad podemos obtener Tweets ya guardados de diferentes formas. Podemos realizar una búsqueda por Hashtag, por usuario o incluso Tweets entre dos fechas previamente seleccionadas.

Esta función ofrece infinidad de posibilidades por lo que, después de experimentar varios problemas con la funcionalidad a tiempo real, se optó por incluir esta en el desarrollo de la herramienta.

Existen 3 tipos de API que podemos utilizar, Standard, Enterprise y Premium.
La diferencia entre ellas es el tiempo entre el que permite la búsqueda, empezando por desde 30 segundos de su publicación hasta 7 días en la API estándar hasta los Tweets publicados en 2006 como es el caso de la API Premium.
Todas ellas proporcionan un acceso a las búsquedas de baja latencia y total fidelidad. En todas las búsquedas los tweets se sirven en desde el Tweet más anterior al más reciente.
En la siguiente imagen podemos ver las principales diferencias entre los tres tipos.

Tweets	Standard (free)	Premium	Enterprise
Publish and engage	✓		
Search Tweets: 7-days	✓		
Search Tweets: 30-days		✓	✓
Search Tweets: Full-archive		✓	✓
Filter Tweets	✓		✓
Sample Tweets	✓		✓
Batch Tweets			✓
Direct Messages	✓		
Account and users	✓	✓	✓
Metrics			✓
Ads API	✓		
Publisher tools and SDKs	✓		

Figura 34. Comparativa API Twitter.
Fuente: https://developer.twitter.com

En esta herramienta se ha utilizado la API estándar, ya que cubre todas las necesidades específicas para el desarrollo.
Como podemos observar, con esta API podemos realizar búsquedas en Tweets publicados desde el momento que realizas la búsqueda hasta los 7 días anteriores a esta.

A continuación, se van a detallar los parámetros con los que podemos realizar las búsquedas y con los que podemos especificar un rango concreto de búsqueda:

- q = con el identificaremos donde se va a realizar la búsqueda, si deseamos que se realice en un usuario concreto o en un hashtag en general.

- geocode = podemos introducir latitud y longitud específicas para realizar la búsqueda en Tweets que hayan sido enviados desde esa población en concreto.

- lang = con este parámetro filtramos Tweets publicados en un lenguaje concreto.

- count = con este parámetro seleccionaremos el número de tweets que queremos que se nos muestre o que necesitamos guardar.

- until = gracias a este parámetro podemos realizar búsquedas entre dos fechas dadas. El formato de búsqueda debe ser AAAA-MM-DD

Se muestra a continuación un ejemplo de esta implementación y de los parámetros usados en el desarrollo de la herramienta:

```
hashtag="#"+self.cuadroHashtag.get()
twitter_api1 = tweepy.API(auth, parser=tweepy.parsers.JSONParser())
search_results = twitter_api1.search(q=hashtag,count=50,lang="es")
statuses = search_results['statuses']
```

Figura 35. Búsqueda hashtag.
Fuente: Elaboración propia.

Se observa que se van a realizar búsquedas por Hashtag introducido por el usuario.
En un primer momento es necesario obtener el hashtag introducido, que lo guardaremos en la variable “hashtag”.
Después realizaremos la conexión con Twitter y obtendremos los datos en formato JSON, para esto necesitamos incluir en nuestra búsqueda los parámetros preferidos, en este caso, se ha utilizado el parámetro *q* para realizar búsqueda en Hashtag, el parámetro *count* para especificar que queremos obtener un total de 50 Tweets y el parámetro *lang* para especificar que tan solo queremos obtener Tweets en español.

5.2.1.5 Url Request

Con esta librería seremos capaces de obtener información de una URL específica, ya sea para simplemente visualizar que contiene o para extraer el texto que se incluye en ella.
En esta herramienta esta librería ha sido utilizada simplemente para extraer el texto que se encontraba en una *URL* introducida por teclado por el usuario, insertada esta como un método de entrada.
El código en el que se ha implementado esta librería es el siguiente:

```
response=urllib.request.urlopen(direccionWebArreglada)
html=response.read()
```

Figura 36. Búsqueda en web.
Fuente: Elaboración propia.

Como podemos observar, la *URL* que ha introducido el usuario por teclado se encuentra guardada en la variable *direccionWebArreglada*, por lo tanto, realizamos un *urlopen* hacia esa dirección.
Finalmente, con el método *read* vamos a guardar en la variable *html* todo el texto encontrado en esa dirección.

5.2.1.6 Web Browser

Con esta librería se permite la visualización de archivos web. Para ello, tan solo utilizaremos el módulo *open*, para realizar la apertura de nuestro navegador predeterminado hacia la dirección web que le indiquemos.

Esta librería, permite presentarle al código cuales son los navegadores instalados en nuestro ordenador personal y dependientemente de la web a visualizar se elegiría uno u otro. Dado que esta herramienta se ha desarrollado con el propósito de utilización de un usuario final independiente del desarrollador, no se incluirá está función, y solo se habilitará la apertura del navegador que el usuario final haya establecido como predeterminado.

Para esto, como se ha indicado anteriormente, se utilizará el método open, además de incluir *new_tab* para indicar que la apertura se realice en una nueva ventana.

```
webbrowser.open_new_tab(direccionWeb)
```

Figura 37. Apertura del navegador.
Fuente: Elaboración propia.

Se puede observar, que es necesario pasarle como parámetro la variable donde se encuentra almacenada la dirección web que queremos mostrar en un navegador web.

5.2.1.7 Beautiful Soup

Es una librería de Python con la cual podemos extraer texto y datos en general de una página web en *HTML* o *XML*.
Aunque podríamos extraer el *HTML* de una página web sin la necesidad de incluir esta librería, es de vital importancia incluirla, ya que realizará una extracción del texto de una manera más limpia y más detallada que si obviamos su uso.

Dada la finalidad de la herramienta en desarrollo, la eficiencia con la cual se lee el texto introducido por sus múltiples métodos de entrada, es sumamente importante. Ya que, nos encontramos ante la necesidad de realizar operaciones muy complejas en el menor tiempo posible, al ser los recursos limitados.

Necesitamos conectar con la *URL*, extraer el texto que en ella se encuentra, almacenarlo y finalmente obtener las emociones. Dado la complejidad de esto, si el texto principal que obtenemos viene cargado de errores o de datos inservibles estamos ralentizando la herramienta de una manera muy poco eficaz por lo que, al utilizar esta librería, seremos capaces de obtener un texto limpio el cual almacenaremos también de una manera más limpia y de donde obtendremos las emociones encontradas de forma más eficaz.

A continuación, se muestra un ejemplo del código utilizado para el uso de esta librería, la cual tan solo contiene un módulo principal, también llamado *Beautiful Soup*:

```
soup=BeautifulSoup(html,"html5lib")
```

Figura 38. Beautiful Soup.
Fuente: Elaboración propia.

Como podemos observar en la línea introducida, vamos a guardar en la variable *soup* el texto obtenido con anterioridad, pero de una forma más limpia, obviando imágenes y todos los demás caracteres utilizados en el desarrollo de la web.

Es necesario introducirle dos valores de entrada, estos son, en primer lugar, el texto obtenido de la web en una primera forma, y el segundo valor necesario es la librería de Python que realizará la limpieza de nuestro código. Con esto estamos refiriéndonos a la estructura que sigue el texto que hemos introducido en nuestra variable de entrada, y como no puede ser de otra forma, hemos utilizado *html5lib*, el cual es el que utilizan la gran mayoría de *URLs* y de navegadores utilizados.

6 DESARROLLO DE LA HERRAMIENTA

6.1 Generación del léxico

Como se ha comentado en apartados anteriores, uno de los objetivos principales de esta herramienta era la generación de un léxico propio con el que obtener unos resultados parecidos a los obtenidos con los principales léxicos. De igual forma, también se propone que este léxico propio tengo un tamaño mucho menor que los demás, para de esta forma, que sea más eficaz.

La generación del léxico propio se ha realizado de forma manual, para esto se han seguido los siguientes pasos:

1- En un primer momento, se crea la primera versión del diccionario añadiendo a este las palabras conocidas que en un ámbito general y en un uso común del lenguaje conocemos que muestran una emoción, siendo estas las principales y más fáciles de reconocer.

2- Se realiza una búsqueda exhaustiva de palabras que reflejan cada una de las emociones, para esto se utilizan las webs detalladas en la bibliografía.
Se anotan todas estas palabras y se añaden a las ya indicadas en el punto 1.

3- Tras realizar una búsqueda de palabras que reflejan emociones en español, también se realiza una búsqueda de estas en inglés, detallando después su traducción para así general un diccionario desde diferentes perspectivas.

4- De todas las palabras anotadas que reflejan al menos una emoción, se realiza una búsqueda de sinónimos de estas, ya que con ellas también se refleja la emoción de la palabra principal que habíamos encontrado.

5- Se comprueban las palabras que pueden no solo pertenecer a una emoción, si no que pueden trasmitir diferentes emociones. Estas son añadidas a todas las emociones correspondientes, generando así la cuarta versión del diccionario.

6- Tras añadir todas las palabras que reflejan varias emociones, se realiza un escaneo del diccionario en búsqueda de palabras repetidas, eliminando todas las que se encuentran.

7- Finalmente, se incluyen al diccionario palabras usadas comúnmente en redes sociales o en textos de un carácter menos culto, añadiendo así también la posibilidad de análisis de textos escritos coloquialmente.

Tras realizar todos estos pasos, se genera un léxico propio, realizado totalmente de forma manual y comprobando en todos los pasos que efectivamente al añadir una palabra a este, esta nos transmite una emoción en concreta o varias, para así asegurarnos que el diccionario será efectivo en el desarrollo de la herramienta.

El lexicón propio está compuesto por:

241 palabras de alegría.
60 palabras de asco.
143 palabras de ira.
175 palabras de miedo.
105 palabras de sorpresa.
461 palabras de tristeza.

En total, el lexicón está compuesto por 1185 palabras.

6.2 Métodos de entrada

Los diferentes métodos de entrada utilizados en el desarrollo de la herramienta pretendían ser una de las dudas más importantes para el correcto desarrollo de esta. Siendo esto los primeros puntos de inflexión y razonamiento a la hora de la planificación.

En un primer momento, la idea era tan solo incluir como método de entrada la lectura de documentos y la introducción manual del texto por el usuario. Ya que, estos eran los de máxima utilidad y con los cuales se podía verificar correctamente el uso de la herramienta. Tras la implementación de ambos métodos, se llegó a pensar en incluir alguno más, sobretodo algo que pudiera conectarnos realmente con la finalidad de la herramienta, como puede ser el extraer la emoción de una red social o incluso de textos de opinión que podíamos encontrar en diferentes webs.

Llegados a este punto, se implementó la búsqueda de emociones en tweets, abriendo mucho más el rango de uso de la herramienta y ampliando el número de usuarios por los que podía ser utilizada. Además, esto añade un plus de curiosidad en la aplicación ya que podemos extraer las emociones o sentimientos que mostramos en nuestros tweets ya escritos.

Además, también se añadió la búsqueda en webs, con la cual tan solo es necesario escribir la URL donde queremos que se realice la búsqueda de emociones. Esto añade un alto grado de curiosidad en la aplicación, ya que podemos verificar que nos intentan transmitir diferentes periódicos en sus columnas de opinión, así como conocer cuáles son los sentimientos que transmiten páginas que en un principio deberían ser neutras.

Debido a la vital importancia de cada uno de los métodos y a la gran utilidad que le añade cada uno de ellos a la herramienta, estos han sido los elegidos como métodos de entrada en el desarrollo de la herramienta, siendo cada uno de ellos de vital importancia, por lo que se detallaran a continuación tanto su uso como su implementación final.

6.2.1 Lectura de documentos

El botón habilitado en la herramienta para realizar esta funcionalidad es el siguiente:

Figura 39. Icono documentos
Fuente: www.freepik.com

Si dejamos el ratón sobe el icono, nos aparecerá el siguiente mensaje:

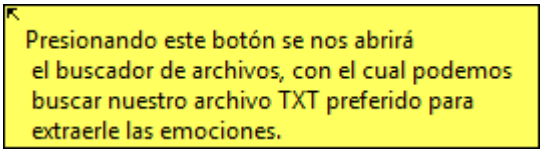


Figura 40. Globo documentos.
Fuente: Elaboración propia.

Este método de entrada es de vital importancia debido a su amplio uso y con el cual podemos probar realmente nuestra aplicación.

Para la implementación de este método de entrada debemos implementar una gran cantidad de funciones primero, ya que no solo necesitamos leer un documento, si no también buscarlo en el disco duro de la máquina donde se está ejecutando la herramienta.

Por lo que, en primer lugar, debemos implementar un *filedialog*, esto se refiere a la ventana de búsqueda de documentos a la cual estamos familiarizados en Windows. Con esta ventana de búsqueda podemos movernos libremente por nuestros archivos, así como cambiar libremente de carpetas o tipos de archivos, además, podemos incluir el nombre de archivo en la barra de búsqueda para una búsqueda más efectiva.

En este *filedialog* se ha incluido la opción de que tan solo sea posible seleccionar ficheros de texto (*txt*), ya que son los únicos que la herramienta va a leer y, por lo tanto, obviamos una búsqueda más completa.

Finalmente, y una vez seleccionado el archivo preferido, la herramienta leerá una a una las palabras encontradas en este archivo, las almacenará en una lista para después, en un siguiente paso, realizar las comprobaciones para la búsqueda de emociones existentes.

6.2.2 Búsqueda de Tweets

El botón habilitado en la herramienta para proporcionar una búsqueda de Tweets es el siguiente:

Figura 41. Icono Twitter.
Fuente: www.freepik.com

Si dejamos el ratón sobre él, aparecerá el siguiente mensaje:

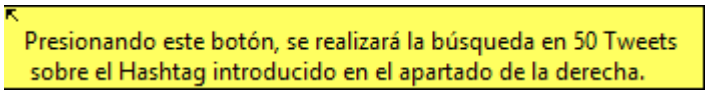


Figura 42. Globo Twitter.
Fuente: Elaboración propia.

Para realizar la búsqueda correctamente en Twitter, es necesario que el usuario introduzca un Hashtag en el apartado correspondiente habilitado para ello, de no ser así, nos aparecerá la siguiente alerta:

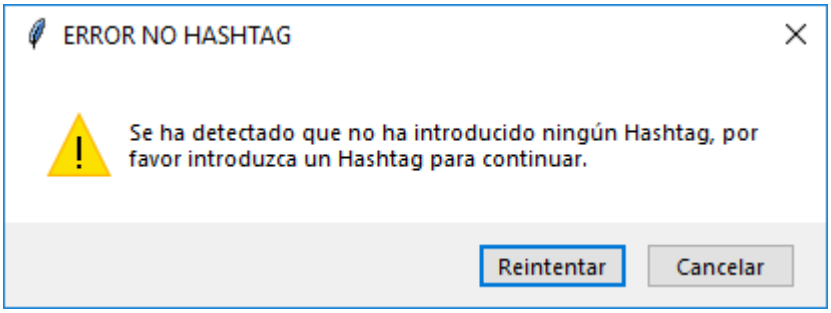


Figura 43. Error no hashtag
Fuente: Elaboración propia.

Si el usuario ha introducido un Hashtag correctamente en el apartado correspondiente, aparecerá el siguiente mensaje al presionar el botón:

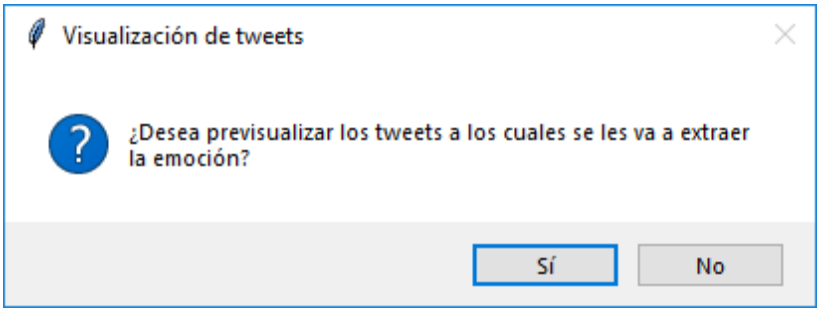


Figura 44. Previsualizar tweets.
Fuente: Elaboración propia.

Al mostrarnos este mensaje, la herramienta espera que presionemos el botón “Sí” o “No”. Si presionamos “Sí” se nos abrirá nuestro navegador predeterminado, mostrándonos los Tweets a los cuales la herramienta les va a extraer la emoción.
Si presionamos “No” la herramienta seguirá su ejecución sin realizar la apertura del navegador.

Sea cual sea nuestra decisión en esta última alerta, la herramienta nos mostrará el siguiente mensaje:

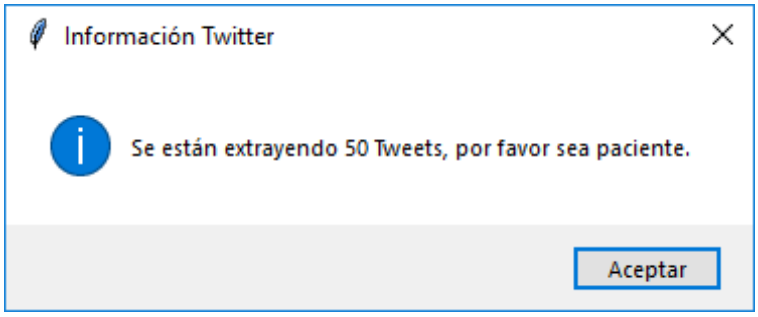


Figura 45. Extracción tweets.
Fuente: Elaboración propia.

Este método de entrada, es el que, para mí, presenta una mayor curiosidad. Es de vital importancia añadirle una funcionalidad a la herramienta que nos conecte al mundo y que mejor que para realizar esta conexión introduzcamos una conexión con Twitter. Día a día se escriben millones de tweets, se utilizan decenas de miles de hashtags en el que tanto personas de nuestro entorno como los famosos más valorados en el mundo muestran sus opiniones y emociones.

Es algo curioso e importante poder saber que emociones predominan en sus tweets, conocer cómo se sienten en cada momento y reflexionar todo lo que mostramos al mundo. Gracias a esta funcionalidad podemos obtener toda esta información, para ello necesitamos implementar varias funciones en nuestro código antes de realizar este análisis.

Principalmente, debemos usar la API con la que Twitter permite a los desarrolladores crear una cuenta en su plataforma para realizar todo tipo de funciones, como, por ejemplo, la extracción de tweets usada en esta herramienta.

La instalación de esta API se detalla en el “Manual de Instalación de Librerías y Módulos” al final de este documento.
En el siguiente apartado detallaremos el uso de esta API, así como, la multitud de funcionalidades que permite y el uso que a estas se le ha dado.

6.2.3 Búsqueda en página web

El botón habilitado en la herramienta para realizar la búsqueda en una página web es el siguiente:

Figura 46. Icono Internet.
Fuente: www.freepik.com

En el cual, si dejamos el ratón sobre él nos aparecerá el siguiente mensaje:

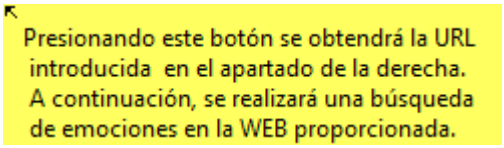


Figura 47. Globo Internet.
Fuente: Elaboración propia.

Para realizar la búsqueda en una página web es necesario que el usuario introduzca su URL preferida en el cuadro indicado para ello.
Existen dos casos en los que nos podemos continuar con el correcto desarrollo de la herramienta:

- No se ha introducido ninguna URL y se presiona el botón de búsqueda en web: En este caso nos aparecerá un mensaje alertando de que no se ha encontrado ninguna URL correctamente escrita en el cuadro de búsqueda y que esta es necesaria para el correcto funcionamiento de la herramienta.

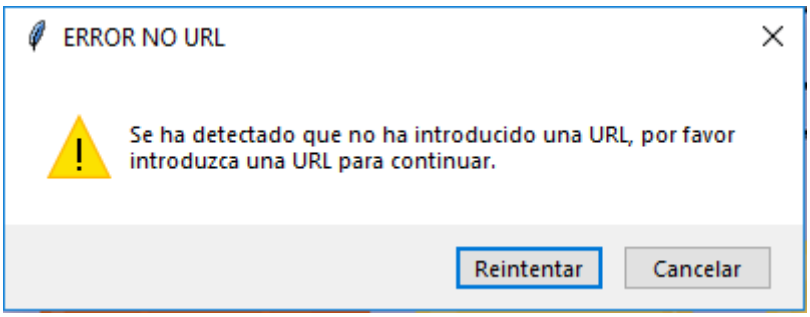


Figura 48. Error no url.
Fuente: Elaboración propia.

- La URL introducida es una URL con certificado y la herramienta no es capaz de procesar estas solicitudes, por lo que aparecerá el siguiente mensaje de error:

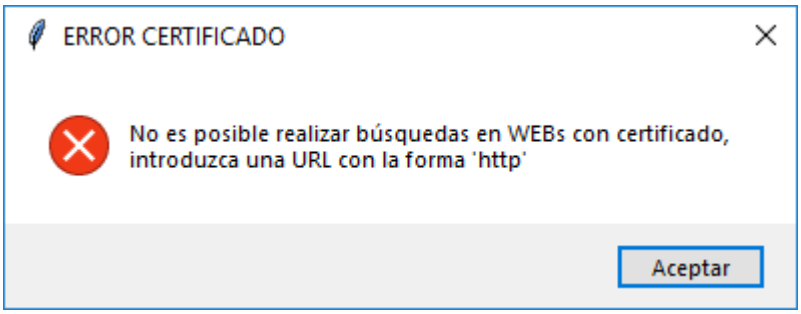


Figura 49. Error certificado.
Fuente. Elaboración propia.

- Tras la búsqueda en la URL introducida, se ha detectado que es una WEB de carácter general en la que se encuentran más de 1500 palabras y, por lo tanto, la herramienta no es capaz de procesar la solicitud.

Sería necesario introducir una URL más específica para poder proseguir con el funcionamiento de la herramienta.

Ejemplo: www.elpais.es

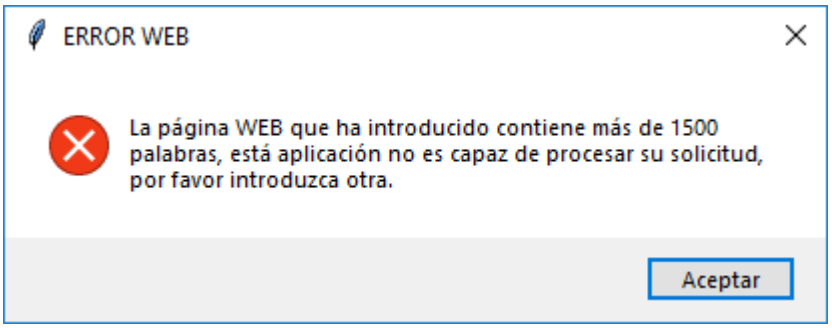


Figura 50. Error web.
Fuente: Elaboración propia.

Si tras presionar el botón de búsqueda, no nos aparecen ninguna de estas alertas, se está realizando un correcto uso de la herramienta.

Ejemplo de uso:

Vamos a realizar una búsqueda en la siguiente *URL*,

https://www.docsity.com/es/universidad/es/universidad-de-jaen/

En la cual se muestran las opiniones sobre la Universidad de Jaén.

Al introducir esta *URL* en el apartado dedicado para ello, nos aparecerá el siguiente mensaje:

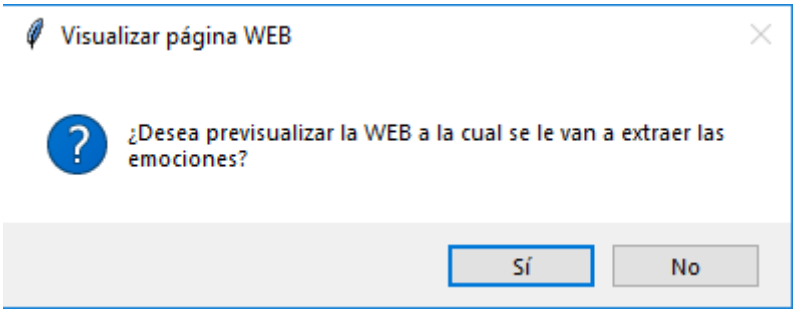


Figura 51. Previsualizar web.

Fuente: Elaboración propia.

En este caso debemos pulsar en el botón “Si” o “No”.

Si deseamos que visualizar en nuestro navegador preferido la *WEB* introducida, debemos presionar “SI”, por el contrario, debemos presionar “No”.

Sea cual sea nuestra decisión, al pulsar cualquier de estos dos botones, la herramienta nos mostrará otro mensaje:

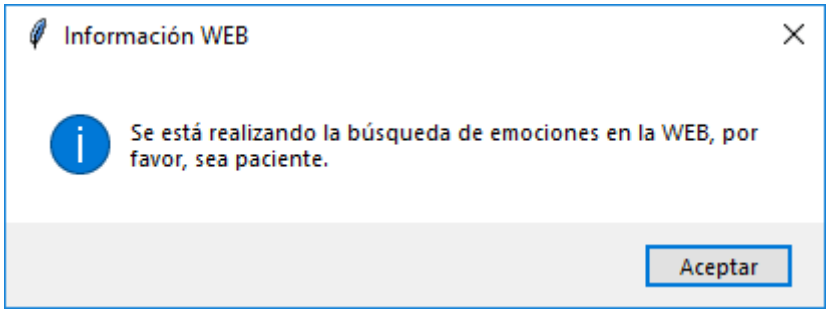


Figura 52. Información web.
Fuente: Elaboración propia.

Debida a la alta necesidad de procesamiento que conlleva esta solicitud, se ha optado por mostrar el mensaje anterior, con el cual se informa al usuario de que la solicitud se está llevando a cabo, pero se necesitan unos segundos para mostrar los resultados, por lo que se pide que “sea paciente”.

6.2.4 Introducción manual del texto

El botón habilitado en la herramienta para lanzar esta funcionalidad es el siguiente:

Figura 53. Icono teclado.
Fuente: www.freepik.com

Si dejamos el ratón sobre él, aparecerá el siguiente mensaje:

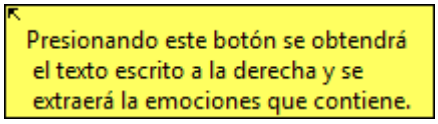


Figura 54. Globo teclado.
Fuente: Elaboración propia.

Si presionamos el botón para la extracción de las emociones y no hay ningún texto escrito en el apartado dedicado para esto, aparecerá la siguiente alerta:

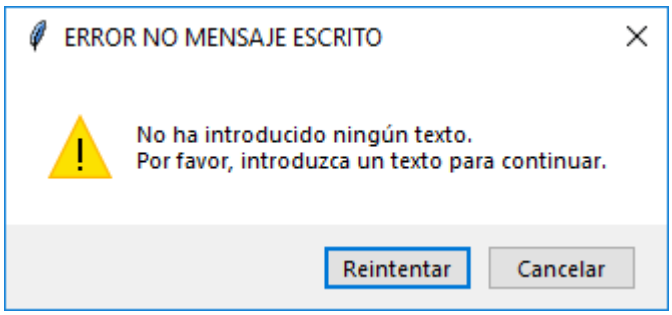


Figura 55. Error no mensaje escrito
Fuente: Elaboración propia.

En su defecto, si el usuario ha escrito un texto correctamente la herramienta seguirá su ejecución hacia la extracción de emociones.

¡Importante!
Aun habiendo introducido correctamente cualquier método de entrada, si en el texto introducido en cualquier de sus variantes no se encuentra ninguna emoción, no se puede proseguir con el uso de la herramienta, por lo que aparecerá el siguiente mensaje de error.

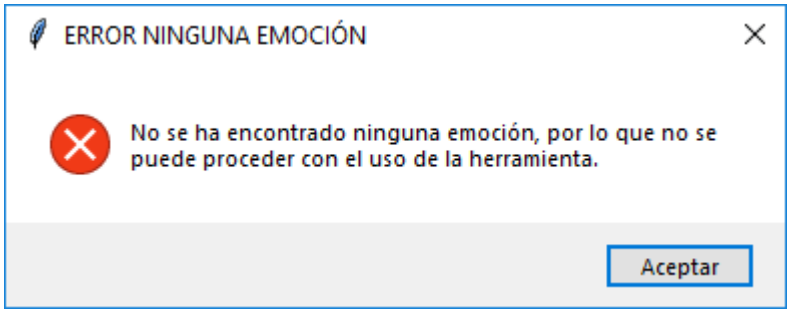


Figura 56. Error ninguna emoción.
Fuente: Elaboración propia.

6.3 Pre-procesamiento del texto.

Una vez realizada la lectura del texto deseado por cualquiera de los métodos de entrada, debemos realizarle un pre-procesamiento a este para dejar solamente la parte importante para su análisis, ya que de no ser así estaríamos volviendo ineficiente la herramienta, debiendo analizar más texto del puramente importante.

a) Tokenización

El primer paso a la hora de editar el texto a que deseamos realizar su análisis es la tokenización. De esta forma vamos a separar todas las palabras por las que están formadas las frases de nuestro texto, así como separar los signos de puntuación que hay a continuación de algunas palabras.

Al realizar este paso, tendremos un texto en el cual podemos diferenciar claramente la posición de cada palabra y de cada signo de puntuación en la lista, dato que posteriormente será de suma importancia.

La tokenización genera una lista con todos los elementos que componen la frase. De esta forma se pueden tratar cada uno de ellos individualmente como una posición en el *array*. Esto será de gran utilidad en la finalidad de la herramienta.

b) Isalpha

Una vez realizada la tokenización, conocemos cada una de las palabras y signos de puntuación que componen nuestro texto.
Son estos últimos los que nos importan llegado a este punto, ya que para el análisis de sentimientos los signos de puntuación no tienen ninguna importancia, por lo que se utilizará el módulo *Isalplha* incluido en la librería *NLTK* para realizar la limpieza de los signos de puntuación en nuestro texto.

c) Stop Words

Las palabras de parada (o *Stop Words* en su término en inglés) es un concepto ampliamente conocido en el mundo del análisis del lenguaje.
También son conocidas como palabras vacías, esto se refiere a que son palabras que podemos obviar en el análisis de un texto, ya que estas no muestran ningún tipo de opinión ni sentimiento.

En la lengua castellana, podemos encontrar estas palabras por ejemplo en los artículos o en las preposiciones. Debido a que no son de utilidad para el análisis, no es eficiente que estas palabras ocupen espacio en la base de datos, por lo que sería necesario realizar una limpieza de todas estas palabras.

Para Python existe un módulo en el que se incluyen todas estas palabras, por lo que después de la tokenización, el siguiente paso de pre-procesamiento va a ser encontrar cuales de estas palabras vacías se encuentran en el texto que el usuario ha introducido.
Para ello debemos tener implementado el módulo *StopWords* en nuestro código, tal y como se detalló en apartados anteriores.

Una vez conocidas las palabras de parada en español por nuestro código, debemos buscar en qué posición de nuestra lista leída se encuentran y después generar una nueva lista en las que estas no aparezcan.

En el siguiente código se muestra como se ha realizado este pre-procesamiento con el código implementado en Python:

```
listaLeidaTokenizada=word_tokenize(listaLeida,'spanish')
listaLeidaPuntuacion= [w for w in listaLeidaTokenizada if w.isalpha()]
listaLeidaStopWords=[w for w in listaLeidaPuntuacion if not w in self.stop_words]
```

Figura 57. Pre-procesamiento
Fuente: Elaboración propia.

Para realizar una prueba, vamos a introducir el texto, y visualizaremos cual ha sido la salida para cada una de estas líneas:

Texto de prueba: Me compré este coche, a mi padre le gusta mucho.

```
['me', 'compré', 'este', 'coche', ',', 'a', 'mi', 'padre', 'le', 'gustaba', 'mucho', '.']
['me', 'compré', 'este', 'coche', 'a', 'mi', 'padre', 'le', 'gustaba', 'mucho']
['compré', 'coche', 'padre', 'gustaba']
```

Figura 58. Salida pre-procesamiento.
Fuente: Elaboración propia.

Con tan sólo una pequeña frase hemos podido determinar el aumento de eficiencia que supone este pre-procesamiento a nuestra herramienta.

En un primer paso, gracias a la tokenización, tendríamos 12 elementos que analizar. Tras eliminar los signos de puntuación, conseguimos que nos queden 10 elementos en el análisis. Finalmente, eliminando las palabras vacías, el resultado del texto que tendremos que analizar es de tan solo 4 elementos.

6.4 Método de búsqueda de las emociones en el texto.

Se ha desarrollado un método complejo de búsqueda para encontrar las emociones que contiene el texto obtenido tras la utilización del método de entrada de este por parte del usuario.

Como se ha detallado en apartados anteriores de este trabajo, se ha generado un diccionario propio para cada una de las seis emociones principales.
Por lo tanto, debemos buscar en el texto leído cada una de las palabras para comprobar si esta se encuentra en algún diccionario y por lo tanto pertenece a alguna emoción.
En un primer momento se optó por realizar una búsqueda comparando palabra a palabra cada una de estas y verificando si se encontraba ligada a alguna emoción.
Tras unas primeras pruebas, se puedo verificar que esta no era una solución totalmente correcta, ya que para ello necesitaríamos un diccionario muy extenso en el que se detallaran todas las derivaciones que por naturaleza se incluyen en el léxico español.

Se va a exponer un ejemplo para detallar esto:
Realizando una búsqueda palabra a palabra, si en un texto nos encontramos con las palabras; contento, contenta y contentos, necesitaríamos incluir estas tres palabras en nuestros diccionarios para que la herramienta las detectara por separado.
Esto deriva en una ampliación casi infinita del diccionario, teniendo que incluir todas las derivaciones ligadas a género o a si estas se encontraban en plural o en singular.

Para solucionar este problema, se ha utilizado la librería *SnoballStemmer*, como se detalló en apartados anteriores, gracias a ella obtendremos la raíz de cada palabra, y realizaremos la búsqueda de palabras con tan solo obtener la raíz, de esta forma obtendremos la raíz de todos los diccionarios pertenecientes a cada emoción, para después comparar estos con el texto leído.
Así, solucionamos el problema antes detallado, mejorando mucho la eficiencia de la herramienta, y utilizando un diccionario mucho menos extenso que los demás utilizados en la herramienta, siendo esto el primer objetivo de mejoría en el que se propuso la herramienta.

6.5 Comparativas en métodos de búsqueda

Para la búsqueda de palabras de emoción según los otros dos lexicones, *NRC* y *SEL* se ha optado por una búsqueda palabra a palabra, ya que estos lexicones son muy extensos y se puede considerar que incluyen todas las palabras.
Por lo que, se ha podido verificar que se han obtenido resultados parecidos con dos métodos diferentes.

- El primer método es el utilizado con el lexicón propio, con el cual se ha obtenido la raíz de cada palabra y se ha comparado con la raíz de las palabras encontradas en el texto.
- El segundo método es el utilizado con los lexicones *NRC y SEL*, con los cuales se ha comparado palabra a palabra si estas se encontraban en el diccionario para cada emoción.

De esta forma, se ha podido realizar la comparativa de ambos métodos, no solo centrándose en uno y pudiendo verificar la eficiencia que se obtiene con cada uno de ellos.

Además, de lograr el objetivo principal en el desarrollo de esta herramienta, el cual era precisamente ese, obtener buenos resultados utilizando un lexicón mucho menor que los demás lexicones utilizados.

6.6 Desarrollo de la interfaz

Para el desarrollo de la interfaz se ha utilizado la librería *Tkinter* incluida en Python. El módulo *Tkinter* (*Tk Interface*) es la interfaz estándar de *Python*. Ha sido utilizada para desarrollar toda la interfaz de esta herramienta, ya que permite infinidad de funcionalidades, así como todo tipo de herramientas para su correcto uso.

Además, se han utilizado dos extensiones para este módulo, estas han sido *ttk y tix*.

-*Tix*: Es una extensión que se le puede introducir a *Tkinter*, con esta tenemos 40 funcionalidades más, entre las que se encuentran *PanelWindow, NoteBook, ComboBox o Ballon.* Esta última ha sido la utilizada en nuestra herramienta.

-*TTK*: Esta ha sido la extensión principal utilizada en el desarrollo de la herramienta, ya que se incluyen las funciones más comunes las cuales podemos encontrar también en el módulo principal, pero se encuentran modificadas y con un entorno visual más moderno, por lo que la visualización de la interfaz queda mejor. Entre las funcionalidades que se incluyen mejoradas con esta extensión, podemos encontrar *Label, Button y Entry*.
Todas estas han sido utilizadas en el desarrollo de la herramienta, por lo que se mostraran sus características y como debemos implementar para su correcta utilización.

A continuación, se van a detallar los módulos utilizados en el desarrollo de la interfaz:

a) tix.Tk

Con esta función se crea la ventana principal o ventana raíz. Este es el primer paso en el desarrollo de cualquier interfaz, ya que para incluir *widget* primero debemos crear una ventana donde alojar los *widgets* que posteriormente crearemos. En una primera estancia, al crear la ventana esta aparece totalmente vacía, debemos ir detallando que *widgets* incluiremos en ella y en qué zonas de esta se van a incluir. Para ello tenemos 3 opciones de edición:

-*place*: este método es el más sencillo de los tres. Para posicionar los *widgets* en una ventana utilizando este método tan solo debemos indicar los pixeles exactos donde queremos situar el *widget*. No es muy recomendable utilizar este *widget*, ya que con que tan solo se modifique un poco el tamaño de la ventana ya se cambiaría toda la visualización de la herramienta, por lo que tan solo se es conveniente utilizar este método cuando existen pocos *widgets* o no se prevé una alta modificación de la ventana.

-*pack*: con este método de empaquetamiento, los *widgets* se irán incluyendo en la ventana preferida por orden en el que se encuentran en el código. Utilizando los puntos cardinales podremos detallar donde se mostrará cada *widget*. Este método es utilizado cuando no vamos a incluir un alto número de widgets en nuestra herramienta, ya que de ser así dificultaría mucho su edición y el empaquetamiento de la herramienta final a gusto del desarrollador.

-*grid*: este método es el más completo y el más utilizado en un ámbito más general, con el dividiremos la ventana como en una tabla bidimensional, en la que iremos añadiendo los *widgets* detallando a que fila y a que columna pertenecen de esa tabla bidimensional. Se aconseja utilizar este método cuando se vaya a desarrollar una herramienta con un alto número de *widgets*, ya que facilita mucho la inserción de estos en el orden correcto dentro de la ventana.

En el desarrollo de la herramienta se ha utilizado el método *pack* para la ventana de bienvenida, ya que presenta una alta facilidad si tan sólo hay unos pocos *widgets*,

como en esta ventana solo se incluye la imagen de fondo y el botón de comenzar se ha utilizado este método.
Para las demás ventanas, al introducir un alto número de *widgets* se ha utilizado el método *grid*, ya que permite total libertad a la hora de ordenar estos y de su posterior edición.

b) tix.Toplevel

Con este método somos capaces de crear ventanas de un nivel superior a las creadas con *Tk*. Por lo que, en un primer momento crearemos la ventana padre o ventana principal con *Tk*, para después crear las demás ventanas que dependen de esta. Esto se realiza utilizando *Toplevel*, de esta forma podemos ir definiendo la herramienta como un grupo de ventanas padres e hijo, detallando cuales dependen de cada una para que así se pueda determinar el camino de navegación correcto para el cual la herramienta ha sido desarrollado, y que, de esta forma, no se pueda cerrar una ventana padre de la ventana que estamos utilizando en ese momento. De esta forma se asegura el correcto uso de la herramienta.

c) ttk.Label:

Este *widget* es utilizado para mostrar imágenes y texto en nuestra herramienta, además permite la edición de todos sus parámetros, así como el tamaño del texto, o el fondo de la etiqueta.
Se ha utilizado este *widget* con asiduidad en nuestra herramienta, ya que es necesario para mostrar toda la información en texto de la herramienta, así como para incluir las imágenes en ella, para esto principalmente hay que agregar una etiqueta a la herramienta para después modificar esta añadiéndole una imagen como icono.

Se muestra a continuación un código de ejemplo para incluir este *widget* en nuestra herramienta:

```
etiquetaPropio=ttk.Label(ventanaComparar,text="PROPIO",font=("Times",15,BOLD))
```

Figura 59. Etiqueta.
Fuente: Elaboración propia.

Como se muestra en el código utilizado de ejemplo, se ha creado una etiqueta que muestre el texto “PROPIO”, para incluir esta etiqueta en nuestra herramienta es necesario añadirle varios parámetros de entrada, en primer lugar, hay que especificar en la que ventana se añadirá el *widget*, en este caso se incluirá en la *ventanaComparar*, tras esto se incluye el texto que se mostrará en la etiqueta, además de las edición de esta etiqueta, en este caso se incluye el tipo de letra, el tamaño de esta y se añade la función *BOLD* para detallar en negrita el texto introducido.

```
iconoFondo=ttk.Label(self.ventanaEmociones,image=self.photoImgFondo)
```

Figura 60. Etiqueta fondo.
Fuente: Elaboración propia.

Con el anterior código realizamos otra función con el mismo *widget* utilizado anteriormente, en este caso se utiliza la etiqueta para introducir una imagen de fondo,

para ello se declara el *widget* igual que anteriormente, pero en este caso, no se introduce un texto en el, si no que se introducirá una imagen mediante el parámetro *image*.

d) ttk.Button:

Con este *widget* seremos capaces de incluir en nuestra herramienta botones con los cuales ejecutar funciones implementadas en nuestro código. Este *widget*, además, se puede visualizar mediante un texto que indique que función realiza el botón al ser presionado o con una imagen también utilizada para esta función.

La implementación de este *widget* en el desarrollo de cualquier herramienta es de vital importancia, ya que gracias a él podemos definir diferentes caminos de navegación de la herramienta, siendo el usuario el que decida cuál de las funciones incluidas en la herramienta desea utilizar.
A continuación, se incluye un ejemplo de la utilización de este *widget*:

```
botonEspanol=ttk.Button(raiz,image=photoImgBotonComenzar,command=lambda:metodoEntrada(self))
```

Figura 61. Boton Español.
Fuente: Elaboración propia.

En la anterior línea de código se define el *widget*, el cual se llamará *botonEspanol*, como en todos los *widgets* a utilizar en la interfaz es necesario indicar a que ventana pertenece, en este caso pertenece a la ventana *raíz*, a continuación se detalla si el botón mostrará un texto o una imagen, en el ejemplo se observa que este botón se mostrará como una imagen y finalmente debemos incluir a que método de nuestro código llamará el botón una vez que este es pulsado, esto se detalla con el parámetro *command*. En este caso, al pulsar el botón entraremos en la función *metodoEntrada*.

e) tix.Ballon:

Como su propio nombre indica, este *widget* crea un globo para mostrar información sobre otros *widgets*. Este puede ser utilizado a modo de ayuda o para indicar algún tipo de comentario al usuario.

La utilización de este *widget* es opcional en el desarrollo de la herramienta, pero su utilización añade una mejor visualización de las funcionalidades que en ella se encuentra. En el desarrollo de esta herramienta, se ha incluido un globo a todos los botones, mostrando que realiza cada botón, para que el usuario final conozca la funcionalidad del botón sin tener necesidad de pulsarlo.

A continuación, se muestra un ejemplo de utilización de este *widget*:

```
b = tix.Balloon(raiz)
b.bind_widget(botonEspanol, balloonmsg='Presiona este botón para comenzar a utilizar la herramienta')
```

Figura 62. Ejemplo Ballon.
Fuente: Elaboración propia.

Se puede observar que es necesario habilitar esta función e incluirsela a una variable, como se detalla en la primera línea del código, en este caso se encuentra en la variable *b*. En la siguiente línea se muestra el ejemplo de utilización de este globo, es

necesario utilizar el método *bind_widge*t para añadirle el globo a un *widget* creado con anterioridad, en este caso se lo añadiremos al *widget botonEspanol* y el texto que se mostrará al mantener el ratón sobre el será el que se indica en la variable *ballonmsg*.

f) ttk.Entry

Este *widget* permite la visualización de una caja de texto en la herramienta. Esto es utilizada para que el usuario introduzca parámetros de entrada o incluso para visualizar variables incluidas en el código, las cuales necesitan de un procesamiento previo antes de mostrar su valor.

En el desarrollo de esta herramienta ha sido utilizado para estos dos casos ya expuesto, en primer lugar, se ha utilizado para que el usuario introduzca por ejemplo la *URL* a la que quiere introducirle extraerle las emociones o incluso el *hashtag* al que se quiere dirigir para visualizar los *tweets* que hay escritos sobre él.
Además, también se ha utilizado un *Entry* para visualizar la ruta del archivo que elegimos mediante el método de entrada de lectura de archivos.
A continuación, se muestra un ejemplo de utilización de este *widget*:

```
cuadroRuta=ttk.Entry(self.ventanaTipoEntrada,textvariable=self.variableRuta,width=80)
```

Figura 63. Ejemplo Entry.
Fuente: Elaboración propia.

Como se observa, para la definición de la variable en la que se incluirá este *widget* para su posterior edición y colocación en la interfaz es necesario incluirle como parámetro de entrada, en un primer lugar la ventana donde se incluirá este *widget*, a continuación, se le ha asignado a la *variableRuta* este *widget*, con esto definimos que dentro de este *widget* se mostrará lo que contenga la llamada *variableRuta* y este valor será conocido en tiempo de ejecución.

Finalmente, se pueden añadir funciones de edición al widget, en este caso tan solo se le ha editado el ancho del cuadro, siendo este de 80 puntos como se puede ver.

g) Text

Este *widget* proporciona tanto la visualización de un texto como la entrada por teclado del mismo, es utilizado en las herramientas para incluir en ellas un cuadro de texto en el cual sea posible introducir un texto para su posterior procesamiento. Además, también se puede utilizar para mostrar grandes documentos de texto.

Este *widget* ha sido utilizado en la herramienta para el método de entrada en el que se le pide al usuario que escriba su texto preferido al cual se le extraerán las emociones.
A continuación, se muestra un ejemplo de la utilización de este *widget* en nuestro código:

```
self.cuadroEscrito=Text(self.ventanaTipoEntrada,height="10",width="65")
```

Figura 64. Ejemplo Text.
Fuente: Elaboración propia.

Para la utilización de este *widget*, es necesario detallar a que ventana está asociado, además podemos incluir muchas funciones de edición del mismo, en este caso tan solo se ha modificado el ancho y el alto.

h) Listbox

Con este *widget* seremos capaces de mostrar un listado por pantalla, en él se incluirá la variable *list* que deseamos mostrar, siendo este *widget* totalmente editable e incluso ajustable al tamaño de la lista que le hayamos pasado, de esta forma se consigue una lectura más sencilla de esta.

Este *widget* ha sido utilizado en la herramienta para mostrar las palabras encontradas para cada una de las emociones. Al permitir varios tipos de edición del *widget*, también se ha utilizado para mostrar cual es la lista que más palabras contiene, con tan solo cambiar el color de este *widget*.

A continuación, se muestra un ejemplo de la utilización:

```
self.cuadroAlegria=Listbox(self.ventanaEmociones,width="20",height="10")

self.cuadroAlegria.insert(0+i,listaEncontradaAlegria[i])

self.cuadroAlegria.config(background="PaleTurquoise1")
```

Figura 65. Ejemplo ListBox.
Fuente: Elaboración propia.

En una primera línea vamos a crear el *widget*, además es necesario especificar la ventana donde se va a encontrar, en un segundo momento necesitamos introducirle la lista, por lo que se recorrerá la lista con un *for* añadiendo los valores uno a uno.
Finalmente, si es la lista con más palabras de todas las emociones, se cambiará el fondo con la tercera línea de código, está será del código de color PaleTurquoise1.

i) Scrollbar

Este *widget* implementa una barra que domina un *listbox* o un cuadro de texto. Esta barra puede ser horizontal o vertical, además de permitir muchas otras ediciones. Principalmente siempre es utilizada conjuntamente a un cuadro de texto, añadiéndole esta en los laterales para poder visualizar de manera completa lo que en el cuadro se encuentra sin necesidad de crear un cuadro demasiado grande.

En esta herramienta ha sido utilizado tanto en los *listbox*, como en el cuadro de texto de la entrada manual. En ambos casos, su función es mostrar los elementos que hayan quedado ocultos debido a que el texto que se muestra, o la lista que se incluye es más grande que el tamaño del *widget* donde se muestra.
A continuación, se muestra un ejemplo de la utilización de este *widget*:

```
scrollAlegria=Scrollbar(self.ventanaEmociones,command=self.cuadroAlegria.yview)
```

Figura 66. Ejemplo Scroll
Fuente: Elaboración propia.

Se puede ver que se llamará *scrollAlegria* el nuevo *widget* creado, este está asociado a la *ventanaEmociones*, y lo que es más importante, se incluye en el eje y del cuadro alegría, esto se refiere a que será utilizado para mostrar todos los elementos que no han sido posible visualizar en el eje vertical.

j) MessageBox

Este *widget* permite agregar cuadros de mensaje a nuestra herramienta. Estos son utilizados para informar al usuario de una información importante, una alerta sobre el uso de la herramienta o incluso realizarle una pregunta sobre una decisión importante en el desarrollo de la herramienta.

Existen varios tipos de estos mensajes, los cuales se detallan a continuación:

-*showinfo*: crean una ventana mostrando información al usuario. Estos son utilizados para mostrar información de utilidad o ayuda para el correcto uso de la herramienta.

-*showwarning*: con esta ventana se muestra una alerta acerca de la función ejecutada con anterioridad, son utilizadas para advertir al usuario de que no se está haciendo un uso correcto de la herramienta o que de los parámetros introducidos no siguen el canon establecido por el desarrollador.

-*showerror*: con esta ventana se le informa al usuario de que ha ocurrido un error en la petición, esto puede ser debido a que esta función no está implementada o que no se puede procesar su solicitud.

-*askyesno*: al implementar este módulo se le solicita al usuario que decida entre “Si” y “No”, tras pulsar el botón preferido por el usuario, tras esta decisión la herramienta tiene funciones específicas para cada una de las decisiones, siendo estas utilizadas según lo elegido por el usuario.

-*askokcancel*: con esta ventana preguntamos al usuario si quiere seguir con el desarrollo de la herramienta, o por el contrario prefiere pulsar “Cancel” para volver al estado anterior.

En el desarrollo de la herramienta han sido utilizadas todos los tipos de estas ventanas, cada una de ellas para su propósito específico, ya sea para hacer partícipe al usuario del camino de navegación por el que quiere transcurrir o para informar de las funciones que se incluyen en la herramienta.

6.7 Pruebas del desarrollo del software

Para el correcto desarrollo del software este ha sido dividido en pequeños módulos para así afrontar pequeñas partes de la herramienta y finalmente incluirlos todos en la herramienta final, siguiendo así el lema de la programación “Divide y vencerás”.

Al encontrarse con módulos específicos que solo realizan una función, las pruebas de este son mucho más efectivas pudiendo de estar forma comprobar y mejorar todo aquello que no es del todo correcto.

En un primer momento, se creó una pequeña herramienta que era capaz de leer dos listas de palabras y comprobar si existían palabras en común entre ellas.
Tras implementar de forma correcta esta parte, se pasó a incluir el método de entrada por búsqueda en un fichero, siendo este el que se le pasaría al módulo anterior como una lista de palabras.

Además, se generó los diferentes diccionarios encontrados para cada emoción, introduciendo estos también por lectura de un archivo TXT a la herramienta.
Se comprobó que se podía encontrar las emociones incluidas en cada diccionario si el archivo leído mediante el método de entrada las incluía.

Al verificar esto, pudimos detallar que porcentaje de palabras que reflejaban una emoción específica se encontraba en el texto leído, añadiendo así la funcionalidad de mostrar el porcentaje, e identificar con otro color la emoción que más porcentaje había obtenido.

Una vez incluida esta funcionalidad importante, se crearon los diccionarios para los lexicones *NRC* y *SEL*. Añadiendo así una comparativa de los resultados obtenidos con el lexicón propio y con el de los demás.
Tras verificar que se obtenían buenos resultados, se pensó que se podía incluir cuales eran las palabras encontradas y que emoción reflejaban para cada uno de ellos, así se podría verificar de una forma visual que diferencias había entre ellos y a que emoción categorizan cada uno el texto.

Tras añadir las que serían las funcionalidades más importantes en el desarrollo de la herramienta, se empezaron a incluir funcionalidades opcionales las cuales añaden un extra de valor en la herramienta, así como un rango más amplio de funcionamiento.
Se añadieron 3 métodos de entrada más, estos serían; búsqueda de emociones en una dirección web, búsqueda de emoción en *tweets* y entrada manual.
Además de esto, se incluyó la generación de un diagrama de barras para visualizar los porcentajes obtenidos para cada emoción, tanto para el apartado en el que determinamos la emoción principal según el lexicón propio como donde comparamos con los demás lexicones.

Finalmente, se realizó el desarrollo de toda la interfaz donde se implementaría todas las funcionalidades detalladas anteriormente.
Una vez, realizada toda esta implementación, se realizaron todo tipo de pruebas para verificar que por ningún camino de navegación aparece ningún error en ejecución.

<u>Pruebas de los métodos de entrada.</u>

a) Búsqueda de un archivo de texto.

Para comenzar se eligió el método de entrada mediante búsqueda de un archivo de texto en los documentos del usuario. Se verifica que efectivamente se abre la ventana del explorador de archivos al pulsar el botón indicado para ello, además, se verifica que

se lee correctamente el archivo y que se guarda correctamente todas las palabras encontradas en una lista para su posterior procesamiento.

b) Búsqueda en una dirección web.

Para realizar las pruebas de este método de entrada se introdujeron varias direcciones webs a modo de ejemplo, estas se incluían de todas las formas posibles, simulando así un uso común de la herramienta. Para esto se introdujo la dirección web de la universidad de Jaén en todas sus variantes, siendo estas:

www.ujaen.es / ujaen.es / http://ujaen.es / http://www.ujaen.es / https://ujaen.es / https://www.ujaen.es

De esta forma se pudo verificar que para incluir la búsqueda en direcciones web del formato *https* era necesario incluir los certificados de estas, por lo que para no aumentar la complejidad de la herramienta se optó por incluir un mensaje de advertencia en la herramienta y en caso de que aun así se incluyera una dirección web con este formato fuera imposible proseguir con el correcto uso de la herramienta hasta que no se modificara este parámetro.
Para los demás casos, se pudo comprobar que se realizaban las búsquedas correctamente.

Tras esto, se pudo verificar que, si la página contenía más de 1500 palabras, el desarrollo de la herramienta no era correcto y en algunas de las ejecuciones la herramienta colapsaba, por lo que se incluyó una función que leyera la cantidad de palabras para en caso de que se sobrepasara este valor se le indicara al usuario que no sería posible procesar su solicitud, por lo que sería necesario que especificara una dirección web diferente a la que había introducido.

Finalmente, se verificó que si se deja en blanco el espacio para incluir una dirección web y se pulsa el botón con la finalidad de que se realice la búsqueda, la herramienta mostrará un cuadro de texto en el que se pide que por favor se rellené el campo indicado ya que de no ser así no se podrá proseguir con el correcto funcionamiento de la herramienta.

c) Búsqueda en tweets.

Para comprobar si este método de entrada funciona correctamente se ha probado a introducir *hashtags* comunes y no tan comunes, comprobando que efectivamente al pulsar el botón habilitado para ello, se realiza una búsqueda de 50 *tweets*, obviando imágenes y el texto se introduce en la lista creada para ello.
También se verifica que, si el usuario deja en blanco el apartado para escribir el *hashtag* preferido y pulsar el botón para realizar una búsqueda en *tweets*, aparecerá una alerta indicando que no se puede procesar su solicitud ya que no ha introducido ningún *hashtag* y se le pide que por favor introduzca uno válido para poder acceder a él y extraerle las emociones.

Además, con esto también se verifica que la API de *Twitter* está funcionando con normalidad y que las credenciales aún son válidas, ya que de no ser así no se podría ejecutar los módulos incluidos en esta.

d) Búsqueda mediante entrada manual del texto.

Para realizar las comprobaciones pertinentes hacia este método de entrada se ha introducido textos a modo de ejemplo en el cuadro de texto habilitado para ello.
Los textos de ejemplo han ido desde una o dos palabras hasta textos de un gran tamaño, verificando que se podían leer textos amplios sin ningún problema. Además, también se ha verificado que si se introduce un texto en el que no se encuentra ninguna emoción la herramienta no muestra ningún error, tan solo avisa con un cuadro de alerta indicando que el texto introducido no contiene ninguna emoción y pide que por favor se introduzca un texto nuevo.

Además, también se ha verificado que, si se deja el cuadro de texto en blanco y se presiona el botón para extraer la emoción al texto, la herramienta también muestra un mensaje indicando de que no existe ningún texto al que extraerle la emoción, por lo que pide que por favor se escriba uno correctamente.

Pruebas de la búsqueda de palabras que reflejan alguna emoción.

Para verificar si efectivamente las palabras del texto introducido mediante alguno de los cuatro métodos de entrada desarrollados se estaban leyendo correctamente y se estaban incluyendo en la lista de la emoción correspondiente se han hecho muchas pruebas.
En un primer momento, se generó un documento en el que solo se incluía una frase contemplando todas las emociones, para así comprobar que el método de búsqueda que se había desarrollado era eficaz.

Tras unos resultados positivos, se creó un archivo de texto en el que se iban incluyendo todas las emociones menos una, para comprobar que no aparecía ningún error si alguna de las listas de las emociones se encontraba vacía, ya que tras esto se calculaban los porcentajes y demás funcionalidades.
Efectivamente, todo era satisfactorio, ya que para esto se habían añadido excepciones por si alguna de las listas era nula que introdujera un valor 0 en los porcentajes y pudiera proseguir la herramienta con su ejecución.

Por lo tanto, se introdujeron las listas leídas por cualquiera de los métodos de entrada, para verificar que en todas ellas el formato de la lista de palabras creada era el correcto y que se podían comparar con las listas de cada emoción de forma correcta.
Se verificó, por tanto, que efectivamente no había problema con ningún método de entrada y que con todos ellos la búsqueda de las emociones era correcta, teniendo así la parte fundamental de la herramienta correctamente desarrollada.

Pruebas de generación de gráficos.

Para estas pruebas se han generado diferentes textos que mostraban desde tan solo una emoción, hasta incluir textos con todas las emociones.
Con esto se ha conseguido verificar que en todas las opciones posibles que permite la herramienta, el gráfico se generaba de forma correcta y no aparecía ningún error si alguna de las emociones no se encontraba presente.
Para esto se han tenido que ir haciendo pequeños cambios en el código, ya que, en un primer momento, si alguna de las emociones era nula, el gráfico no se generaba como

debida, de esta forma, se iban incluyendo alertas y código suplementario para cubrir estos errores y que así, no hubiera ningún problema al leer cualquier tipo de texto, se encontraran en este todas las emociones, solo unas pocas o incluso ninguna.

Pruebas de comparativa entre diccionarios.

Para realizar las pruebas en las que podemos verificar que el uso de todos los diccionarios está siendo correcto y que la salida mostrada por la herramienta es la que realmente debería de aparecer, se han realizado muchas ejecuciones de ejemplo.
En un primer momento, se generaron los diccionarios para cada lexicón y tras eso se generó un documento mostrando cada una de las emociones, para así, después de este ser leído, realizar la comparativa para cada diccionario.

Tras esto, se generó un documento en el que se incluían todas las emociones, para realizar la comparativa de palabras y mostrar que todos los diccionarios de las emociones estaban siendo leídos y comparados de una forma correcta.
Tras obtener muy buenos resultados, se optó por pasar a la siguiente funcionalidad, la cual era extraer los porcentajes obtenidos para lexicón incluido en la herramienta.
Llegado a este paso se encontraron diferentes tipos de problemas, los principales eran debidos a listas nulas, generadas por textos en los que no se encontraba una emoción concreta y todas las operaciones aparecían nulas, por lo que, era necesario ir incluyendo para estos casos, los valores de las variables que debían obtener si se entraba a este pequeño fragmento de código.

Debugeando cada ejecución y añadiendo esté código para cada uno de las ejecuciones que no eran correctas se consiguió que, fueran cuales fueran los resultados de emociones obtenidas, siempre se mostrará unos porcentajes correctos y que la visualización de la comparativa fuera efectiva. De esta manera, se podía verificar que la herramienta no iba a encontrar ningún error sea cual sea el texto de entrada que le introduzca el usuario.

Pruebas finales.

Para las pruebas finales se han ido creando documentos de texto incluyendo todas las posibilidades posibles que podrían ocurrir en el uso normal de la herramienta.

Estas son:

- Que el documento leído no incluya ninguna emoción, para esto se ha creado una alerta que avise al usuario de que no se ha encontrado ninguna emoción y por lo tanto no se puede proseguir con el correcto funcionamiento de la herramienta, por lo tanto, si quiere proseguir con el uso es necesario que introduzca un texto diferente.

- Que el documento leído tan solo incluya una emoción de las seis posibles, en este caso se comprueba que para las demás emociones no se le introducen valores nulos, si no que se utiliza una lista con valor cero para que todas las funcionalidades que se incluyen después puedan ser ejecutadas con normalidad, siendo por ejemplo estas el cálculo de porcentajes o la generación de los gráficos.

- Que el documento leído incluya dos o más emociones, siendo estas generadas correctamente, incluyendo además las listas vacías para las demás emociones y realizando de esta manera toda la ejecución de manera correcta.

- Que en el documento se incluyan todas las emociones, en este caso se tiene que verificar que cada una de las emociones se incluya en la lista indicada para ello y que no aparezca en una lista que no es la suya, además es necesario verificar que la cantidad de palabras incluidas en cada lista es la correcta y que la generación de gráficos y de porcentajes arrojan los valores correctamente calculados según el total de palabras encontradas.

Finalmente, tras realizar todas estas pruebas, se puede concluir que la herramienta será efectiva en cualquier camino de navegación ya que se ha estudiado cada uno ellos. Además, se ha previsto que algunas de estos caminos de navegación se realicen de forma incorrecta, para que la herramienta avise al usuario de que esa no es la forma correcta de uso y se corrija a tiempo antes de que la herramienta no se capaz de procesar algún error introducido.

6.8 Problemas encontrados en el desarrollo

En un primer momento, el desconocimiento del lenguaje *Python*, hacía que la herramienta implementará la dificultad con la que ya se había caracterizado.

Tras obtener un buen conocimiento de este lenguaje y comenzar la planificación de cómo se iba a desarrollar la herramienta, nos encontramos con otro gran reto, este no era otro que obtener un amplio conocimiento sobre la minería de opiniones que era necesario para desarrollar de una forma correcta y eficaz la herramienta.
Por lo que, tras una intensa búsqueda de información se obtuvo finalmente una idea de los métodos principales de minería de opiniones y como estos podían beneficiarnos en el desarrollo de la herramienta.

Una vez conocido el método de minería de opiniones que se iba a utilizar, nos encontramos con los sentimientos, que, aunque en un primer momento se puede llegar a pensar que son de fácil entendimiento, estos necesitan de una gran comprensión para identificarlos de una forma correcta y así, poder generar un léxico propio de forma manual, incluyendo esto una gran dificultad, pero a la vez añadiendo esto un alto valor a la herramienta.

Generado el léxicon, era necesario detallar como se iban a realizar las búsquedas de las emociones incluidas en ese léxico, siendo esto el principal problema en el desarrollo de la herramienta, ya que se ha necesitado constancia y bastante tiempo para desarrollar un método que sea eficaz y que nos permita eliminar las palabras menos relevantes del léxico, consiguiendo el objetivo principal propuesto para la herramienta, el cual era generar un léxico formado por un número mínimo de palabras con el que obtener unos buenos resultados.

Tras finalmente generar de forma correcta una búsqueda efectiva en las emociones, se presenta el último y gran reto, que no era otro que desarrollar una interfaz intuitiva, que incluyera todo lo necesario para la ejecución de la herramienta y que además se vea profesional.

Mucha dedicación y esfuerzo después, quedaba concluida la interfaz, incluyendo además todos los botones como iconos y además textos explicativos sobre estos, para así ampliar el rango de usabilidad de la herramienta y que esta pueda ser utilizada sin necesidad de conocer en ningún momento como ha sido implementada.

7 LÍNEAS FUTURAS

Tras conseguir con éxito el principal objetivo de en el desarrollo de esta herramienta, el cuál era obtener resultados parecidos con un lexicón la mitad de extenso, se propone para líneas futuras mejorar la búsqueda de las emociones, añadiendo funcionalidades como:

Incluir la negación en la búsqueda de emociones: en el desarrollo de esta herramienta no se ha tenido en cuenta que las emociones presentadas en un texto pueden venir precedidas por una negación, que automáticamente detallarían lo contrario que en un primer momento se obtiene. Sería una buena técnica implementar la lectura de la negación y atribuir esto a la siguiente emoción que se lea, para así generar una herramienta mucho más completa.

Búsqueda de ironía: debido a la extensión que esto hubiera involucrado en el desarrollo de la herramienta, no ha sido posible incluir esta función en esta versión, por lo tanto, sería conveniente incluirla en futuras versiones de la herramienta o en estudios posteriores.
Con la búsqueda de ironía se tendría una herramienta mucho más completa, debido a que esta se encuentra en el lenguaje natural y en la forma en la que nos expresamos, por lo que, si esta no se incluye los resultados de la herramienta estarán correctos en su totalidad.

Búsqueda de n-gramas: esta función añadiría complejidad, pero a la misma vez, una inmensa mejora a la herramienta. En el lenguaje, usamos con asiduidad potenciadores que van a acompañados de palabras que reflejan una emoción, por ejemplo, no se muestra la misma emoción de alegría diciendo, es bonito a es muy bonito.
Sería necesario incluir en la nueva versión de la herramienta estos potenciadores, que forman un bigrama con la palabra principal, o incluso n-gramas más grandes, formados por 3 o 4 palabras, como pueden ser las frases hechas que tanto utilizamos en la lengua castellana.

8 CONCLUSIÓN

Tras el desarrollo de esta herramienta, así como este Trabajo de Fin de Grado en general se ha podido verificar la dificultad que existe en el estudio del lenguaje y en especial de la minería de opiniones.

Es necesario un amplio conocimiento, no solo del lenguaje, sino también de la forma con la que nos expresamos y con la que mostramos nuestros sentimientos.

Hemos podido observar a lo largo de este Trabajo de Fin de Grado, que la minería de opiniones es un campo en el que aún queda mucho desarrollo y en el que aún se puede mejorar las herramientas ya creadas.

Se puede concluir, que la minería de opiniones y que específicamente el desarrollo de una herramienta capaz de realizar esta función, es una tarea tediosa en la que se necesita un amplio conocimiento en diversos campos, pero que a la vez ofrece un valor adicional a aquellos que la utilizan, ya que añade un amplio conocimiento sobre cómo nos expresamos y sobre como mostramos nuestras emociones a los demás.

9 REFERENCIAS BIBLIOGRÁFICAS

[1] Introducción al análisis de sentimientos – Meaning Cloud. (online)
https://www.meaningcloud.com/es/blog/introduccion-al-analisis-de-sentimientos-mineria-de-opinion

[2] Análisis de sentimientos y minería de opiniones – Brain Sins (online)
https://www.brainsins.com/es/blog/analisis-del-sentimiento-y-mineria-de-opiniones/99679

[3] Minería de opinión como técnica para el análisis de información en línea – Infotecarios (online)
http://www.infotecarios.com/mineria-opinion-una-tecnica-analisis-informacion-linea/#.XOw_k4gzbIU

[4] Paul Ekman – Wikipedia (online)
https://es.wikipedia.org/wiki/Paul_Ekman

[5] Las 6 emociones básicas: la teoría psicológica de Paul Ekman – Viviendo la salud (online)
https://viviendolasalud.com/cuerpo-y-mente/emociones-basicas

[6] Las emociones básicas Paul Ekman – Psicocode (online)
https://psicocode.com/psicologia/las-emociones-basicas-paul-ekman/

[7] What is Senpy? – Senpy (online)
https://senpy.readthedocs.io/en/latest/senpy.html

[8] Extractor multipalabra – Linguakit (online)
https://linguakit.com/es/extractor-multipalabra

[9] Sentiment Analysis – Bitext (online)
https://api.bitext.com/#/services/textanalysis/sentiment/spa

[10] Tone analyser – IBM Watson (online)
https://tone-analyzer-demo.ng.bluemix.net/

[11] Mohammad, S. M. (2017). Word affect intensities. *arXiv preprint arXiv:1704.08798*.

[12] Mohammad, S. M., & Turney, P. D. (2010, June). Emotions evoked by common words and phrases: Using mechanical turk to create an emotion lexicon. In *Proceedings of the NAACL HLT 2010 workshop on computational approaches to analysis and generation of emotion in text* (pp. 26-34). Association for Computational Linguistics.

[13] Plaza-del-Arco, F. M., Molina-González, M. D., Jiménez-Zafra, S. M., & Martín-Valdivia, M. T. (2018). Lexicon Adaptation for Spanish Emotion Mining. *Procesamiento del Lenguaje Natural*, *61*, 117-124.

[14] Grigori Sidorov – Sidorov (online)
http://www.cic.ipn.mx/~sidorov/

[15] Sinónimos para Asco – Sinonimos Woxikon (online)
https://sinonimos.woxikon.es/es/asco

[16] Sinónimos asco – Word Reference (online)
http://www.wordreference.com/sinonimos/asco

[17] Pseudo-sinónimos asco – Busca Palabra (online)
https://www.buscapalabra.com/sinonimos-y-antonimos.html?palabra=asco&sinonimos=true

[18] Sinónimos Ira – Word Reference (online)
http://www.wordreference.com/sinonimos/ira

[19] Sinónimos para ira – Sinónimos Woxikon (online)
https://sinonimos.woxikon.es/es/ira

[20] Sinónimos para miedo – Sinónimos Woxikon (online)
https://sinonimos.woxikon.es/es/miedo

[21] 120 palabras relacionadas con el miedo – Blog de Jack Moreno (online)
https://jackmoreno.com/2018/08/18/120-palabras-relacionadas-con-el-miedo/

[22] Sinónimos para sorpresa – Sinónimos Woxikon (online)
https://sinonimos.woxikon.es/es/sorpresa

[23] Sinónimos para tristeza – Sinónimos Woxikon (online)
https://sinonimos.woxikon.es/es/tristeza

[24] 120 ejemplos de palabras tristes – Blog de Jack Moreno (online)
https://jackmoreno.com/2017/10/17/120-ejemplos-de-palabras-tristes/

[25] Sinónimos para alegría – Sinónimos Woxikon (online)
https://sinonimos.woxikon.es/es/alegr%C3%ADa

[26] Pseudo-sinónimos de alegría – Busca Palabra (online)
https://www.buscapalabra.com/sinonimos-y-antonimos.html?palabra=alegr%C3%ADa&sinonimos=true

[27] The Platform for Open Innovation and Collaboration – Eclipse Foundation (online)
https://www.eclipse.org/

[28] Python Interface to Tcl/Tk – Docs Python (online)
https://docs.python.org/2/library/tkinter.html

[29] Developer tools and APIs built for innovation and scale – Twitter (online)
https://developer.twitter.com/en/pricing.html

[30] Sphinx Documentation – Docs Python (online
https://docs.python-guide.org/writing/documentation/

[31] Lista en Tcl/Tk – Recursos Python (online)
https://recursospython.com/guias-y-manuales/lista-listbox-en-tkinter/

[32] Perikos, I., & Hatzilygeroudis, I. (2016). Recognizing emotions in text using ensemble of classifiers. *Engineering Applications of Artificial Intelligence*, *51*, 191-201.

[33] Molina-González, M. D., Martínez-Cámara, E., Martín-Valdivia, M. T., & Perea-Ortega, J. M. (2013). Semantic orientation for polarity classification in Spanish reviews. *Expert Systems with Applications*, *40*(18), 7250-7257.

[34] Las Redes Sociales más utilizadas: cifras y estadísticas - IEB SCHOOL (online) https://www.iebschool.com/blog/medios-sociales-mas-utilizadas-redes-sociales/

[35] Las redes sociales ya superan a la prensa escrita como opción para informarse – Trecebits (online)
https://www.trecebits.com/2018/12/11/las-redes-sociales-ya-superan-a-la-prensa-escrita-como-opcion-para-informarse/

[36] Saif M. Mohammad. – Saif M. Mohammad (online)
http://saifmohammad.com/

[37] Lineas de trabajo I+D+I – OTRI Ujaen (online)
http://otri.ujaen.es/ofertaidi/es/grupo-investigacion/tic-209

[38] The Platform for Open Innovation and Collaboration – Eclipse (online)
https://www.eclipse.org/

[39] Python – Wikipedia (online)
https://es.wikipedia.org/wiki/Python

[40] Procesamiento del Lenguaje Natural – Vicomtech (online)
http://www.vicomtech.org/t4/e11/procesamiento-del-lenguaje-natural

[41] Simplificación automática de textos – INICO
http://inico.usal.es/cdjornadas2015/CD%20Jornadas%20INICO/cdjornadas-inico.usal.es/docs/313.pdf

[42] Categorización automática de textos usando PLN – Adulto Supertado (online)
http://www.adultosupertado.org/t5299-categorizacion-automatica-de-textos-usando-pln

10 ANEXOS

10.1 Manual de usuario de la herramienta

Al ejecutar la herramienta nos aparece la ventana de bienvenida:

Figura 67. Pantalla de bienvenida.
Fuente: Elaboración propia.

En ella se puede visualizar el nombre de la herramienta, así como el desarrollador y los escudos tanto de la Universidad de Jaén como de la Escuela Politécnica Superior de Linares. Para comenzar con el uso de la herramienta debemos pulsar el botón comenzar, además si dejamos el ratón sobre él nos mostrará el siguiente mensaje:

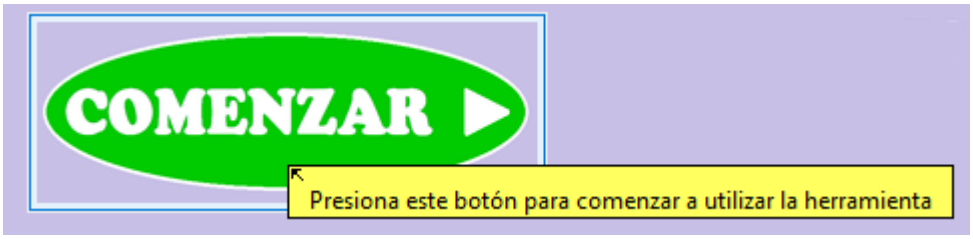


Figura 68. Botón de comienzo.
Fuente: Elaboración propia.

Tras pulsar el botón “COMENZAR”, nos aparecerá la ventana donde tendremos que seleccionar el método de entrada:

Figura 69. Métodos de entrada.
Fuente: Elaboración propia.

En ella podemos visualizar 4 métodos de entrada:

a) Búsqueda de un archivo TXT

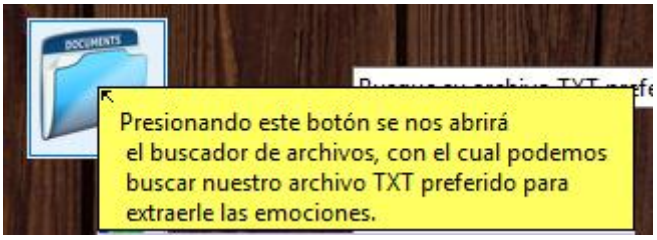


Figura 70. Entrada por archivo.
Fuente: Elaboración propia.

Si presionamos en él nos aparecerá el buscador de archivos de nuestro sistema operativo.

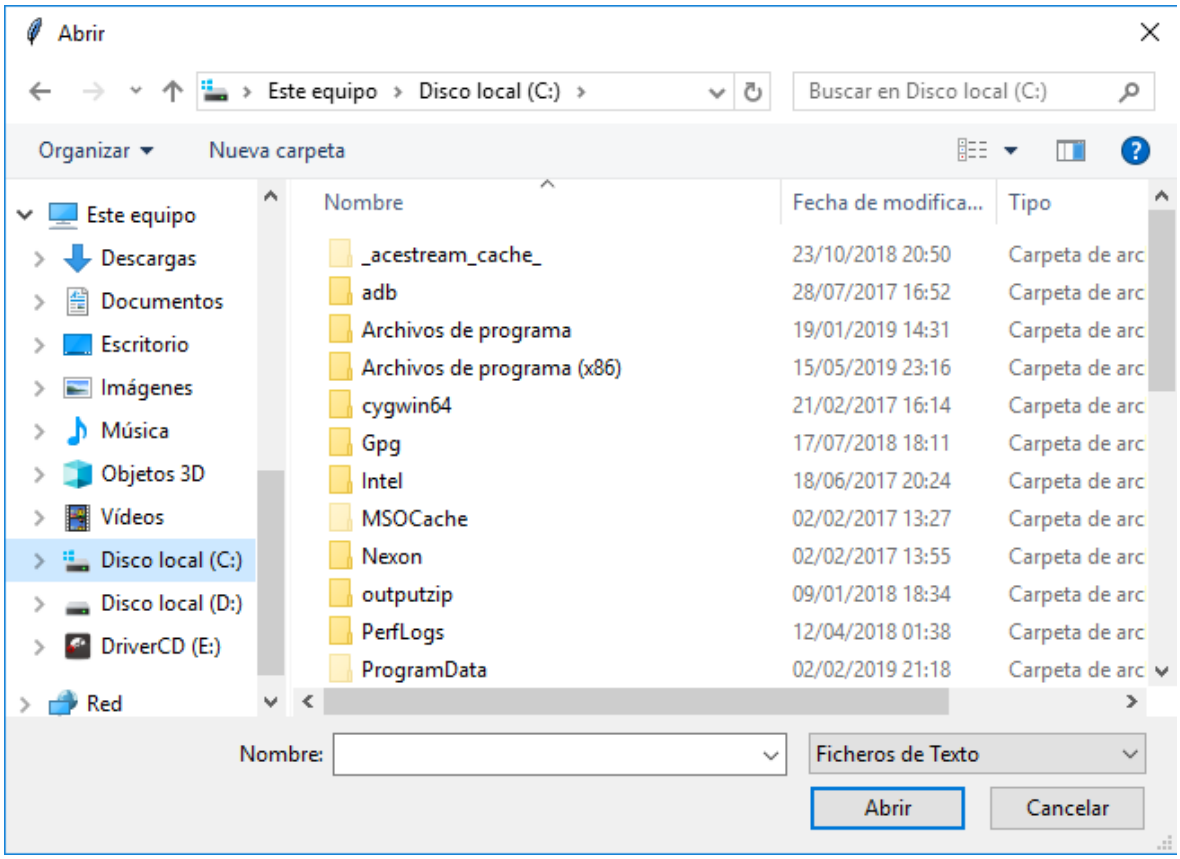


Figura 71. Ejemplo Filedialog.
Fuente: Elaboración propia.

En esta ventana, podemos navegar por nuestros archivos hasta encontrar nuestro TXT preferido.

b) Búsqueda en la Web

Para realizar esta búsqueda, debemos pulsar el botón habilitado para ello. Como se muestra en la siguiente imagen.

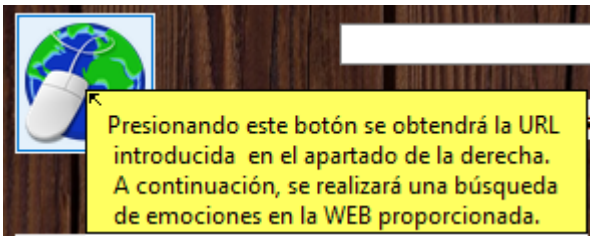


Figura 72. Método entrada búsqueda web.
Fuente: Elaboración propia.

Si no introducimos ninguna *URL* o la *URL* introducida es del formato *HTTPS*, nos aparecerá un mensaje de error indicándonos que por favor introduzcamos una *URL* en la forma correcta.

c) Búsqueda en Tweets

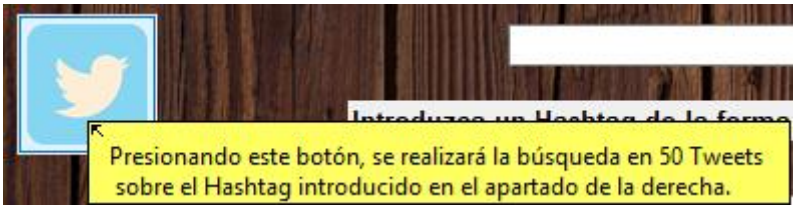


Figura 73. Método entrada búsqueda tweets.
Fuente: Elaboración propia.

Para realizar la búsqueda en 50 tweets, debemos introducir correctamente un *hashtag*, este puede estar formado como habitualmente lo conocemos, esto es #UJAEN.

Si no introducimos ningún *hashtag* y pulsamos el botón, nos aparecerá una alerta indicando que es necesario introducir un *hashtag*.

d) Entrada manual del texto

Para obtener el texto escrito en el cuadro habilitado para ello, debemos pulsar el botón con forma de teclado:

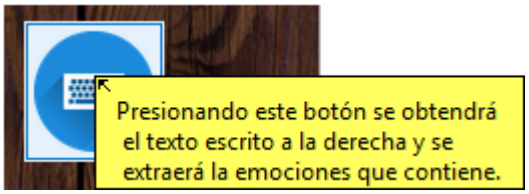


Figura 74. Método entrada teclado.
Fuente: Elaboración propia.

Además, se incluye un botón para borrar de manera inmediata todo lo escrito en el panel habilitado para ello, tan solo debemos presionar el botón siguiente:

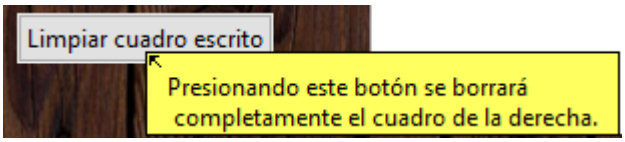


Figura 75. Borrar cuadro escrito.
Fuente: Elaboración propia.

Tras la selección de un método de entrada y la introducción por cualquiera de sus variantes de un texto, nos aparecerá una ventana mostrándonos el texto a modo de previsualización, si estamos de acuerdo con el texto introducido debemos presionar “Utilizar este texto”, de no ser así debemos presionar “Descartar texto” para introducir uno diferente.

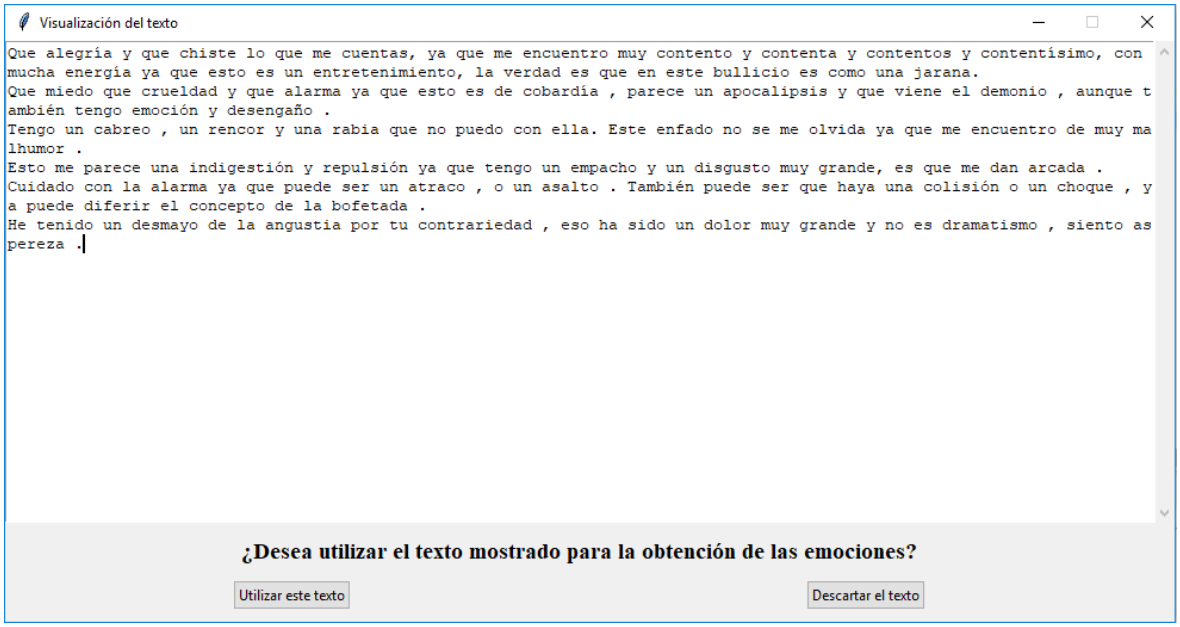


Figura 76. Previsualización del texto.
Fuente: Elaboración propia.

Al presionar “Utilizar este texto”, la herramienta continuará y mostrará la siguiente pantalla, donde se visualizan las palabras encontradas asignadas a cada una de las emociones así como distintos botones que permiten unas acciones nuevas.

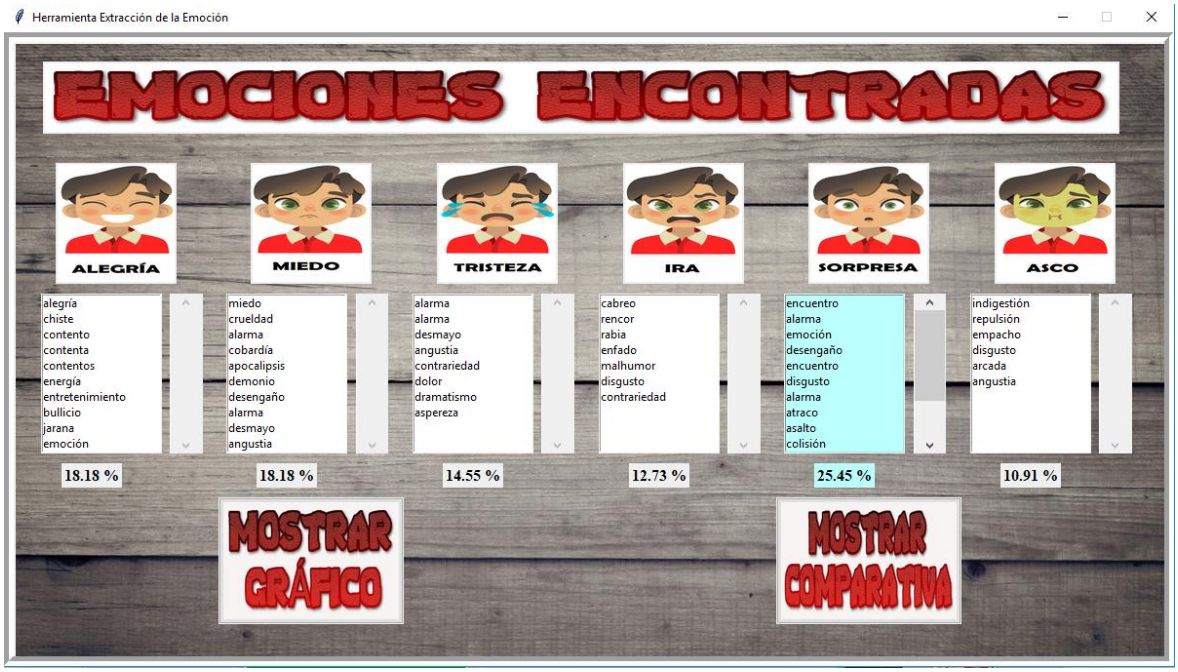


Figura 77. Emociones encontradas.
Fuente: Elaboración propia.

En esta ventana se nos muestra, en un primer lugar un icono definiendo cada una de las emociones principales de Ekman. Tras cada icono podemos encontrar un *Listbox* mostrando todas las palabras encontradas de esa emoción.

Finalmente, debajo de este *Listbox* se muestra el porcentaje obtenido para cada emoción.

Para la emoción principal encontrada, o la que más porcentaje tenga de todas se mostrará en otro color, en este caso en la imagen podemos ver como la emoción sorpresa, tras obtener un 25,45% y ser la mayor, su *Listbox* se muestra en un tono azul claro en el que se determina que ha sido la emoción en la que más palabras se han encontrado.

Además, en esta ventana se incluyen dos nuevos botones:

a) Mostrar gráfico

Figura 78. Icono mostrar gráfico.
Fuente: Elaboración propia.

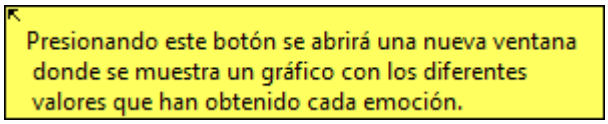


Figura 79. Globo mostrar gráfico.
Fuente: Elaboración propia.

Si presionamos este botón se abrirá una nueva ventana, generando un gráfico de barras en el que se incluyen los valores obtenidos de cada emoción, pudiendo así visualizar de una forma más clara las emociones encontradas en el texto.

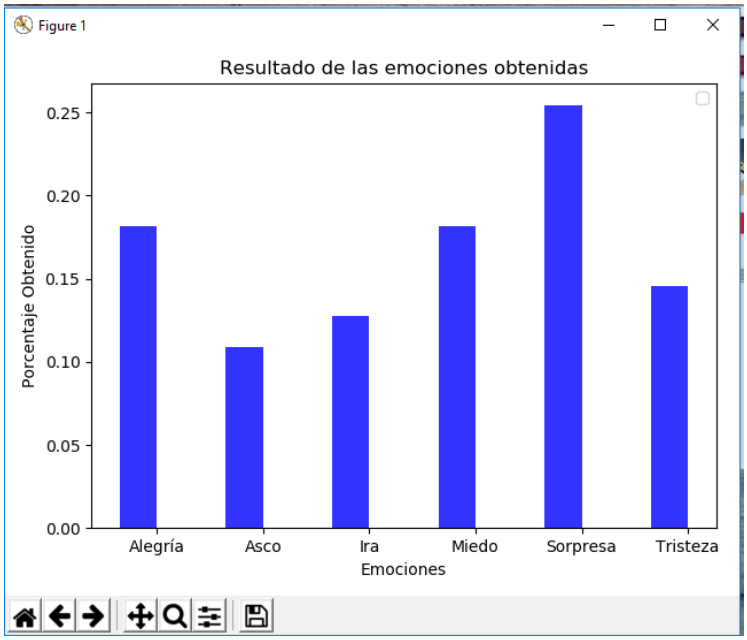


Figura 80. Gráfico generado.
Fuente: Elaboración propia.

En el eje de la x se muestran todas las emociones que se buscaban en el texto, mientras que en el eje de la y se encuentra el rango de porcentajes obtenidos, siendo 1 el valor máximo.

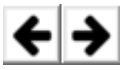
En esta nueva ventana, tenemos además varias opciones como son:

 Permite guardar la figura como un archivo de imagen.

 Permite la edición de los *subplots*.

 Permite realizar zoom al gráfico.

 Permite mover a nuestro gusto el gráfico para su correcta visualización.

 Permite movernos entre gráficas creadas.

 Permite volver a la vista general en caso de que se haya aplicado alguna modificación.

b) Mostrar comparativa

Figura 81. Icono mostrar comparativa.
Fuente: Elaboración propia.

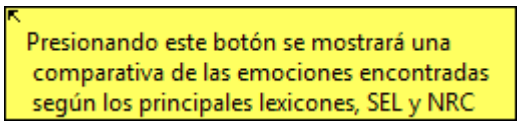


Figura 82. Globo mostrar comparativa.
Fuente: Elaboración propia.

Al presionar este botón, se abrirá una nueva ventana mostrando los porcentajes obtenidos de cada emoción según los principales lexicones.

Figura 83. Comparativa.
Fuente: Elaboración propia.

En esta ventana se muestra una comparativa de los tres lexicones principales, estos son el PROPIO desarrollado para esta herramienta, *NRC* y *SEL*.

Para cada uno de ellos se muestra el porcentaje obtenido para cada emoción, así como también se marca de color verde la emoción que más porcentaje ha obtenido.

Tras esto se muestra un icono mostrando la emoción principal.

Al final de la ventana se muestran dos botones con lo cual podemos acceder a la siguiente parte de la herramienta, uno de ellos es:

Figura 84. Icono mostrar gráfico comparativo.
Fuente: Elaboración propia.

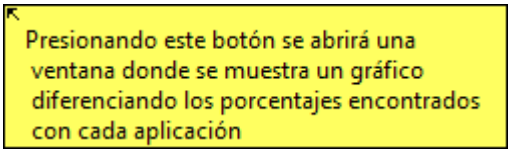


Figura 85. Globo mostrar gráfico comparativo.
Fuente: Elaboración propia.

Al presionar este botón aparecerá una nueva ventana mostrando un gráfico de barras diferenciando los 3 lexicones por colores, como se muestra en la leyenda tenemos el lexicón propio en rojo, el lexicón *SEL* en verde y el lexicón *NRC* en azul.

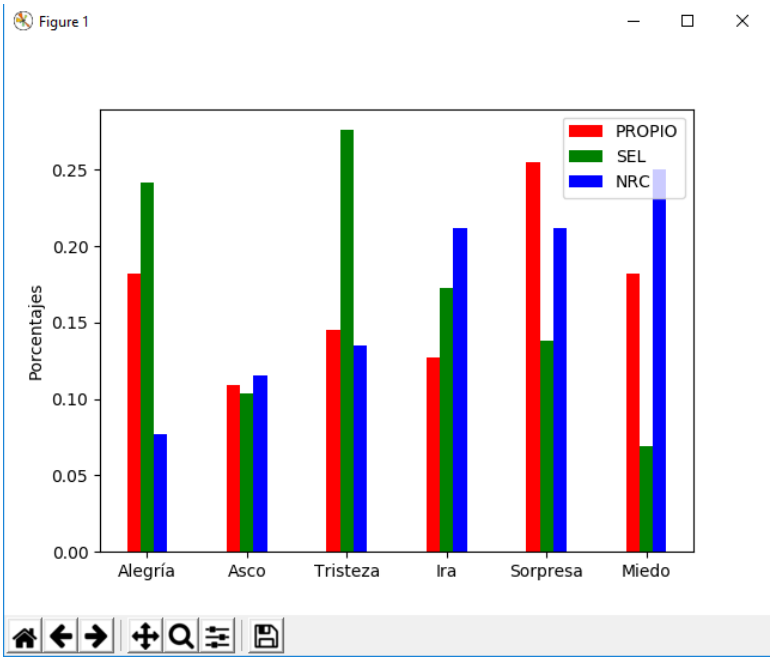


Figura 86. Gráfico generado comparativa.
Fuente: Elaboración propia.

También encontramos el botón “Muestra Palabras”:

Figura 87. Icono mostrar palabra comparativa.
Fuente: Elaboración propia.

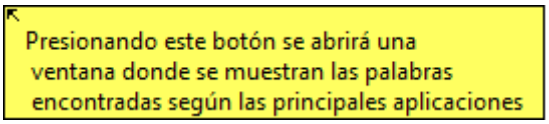


Figura 88. Globo mostrar palabra comparativa.
Fuente: Elaboración propia.

Presionando este botón nos aparecerá la última ventana posible en la herramienta, en esta se encuentran todas las palabras encontradas para cada uno de los diccionarios, así como estos divididos por colores para su correcta visualización.

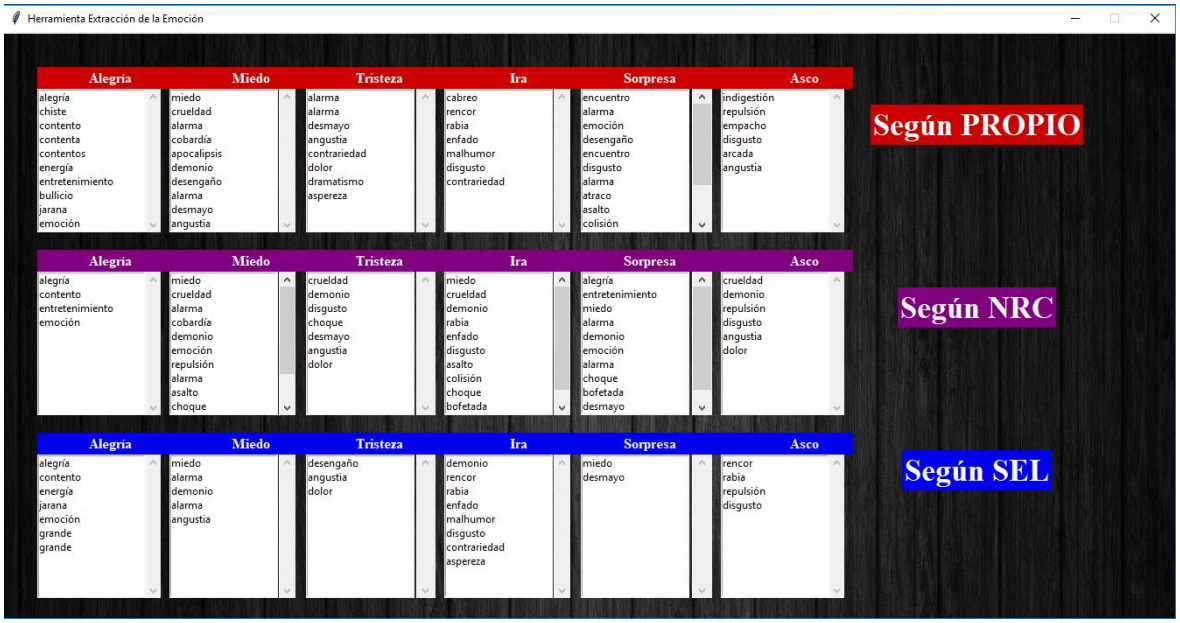


Figura 89. Palabras comparadas.
Fuente: Elaboración propia.

10.2 Manual de instalación de herramienta de extracción de emociones

10.2.1 Características técnicas del equipo.

Se detallan a continuación las características técnicas del equipo con el cual se ha desarrollado la herramienta.

Marca: MSI
Modelo: CX62 6QD
Sistema Operativo: Windows 10 64 bits.
Procesador: Intel Core i7-6700HQ
Memoria RAM: 8 GB DDR4
Gráficos: Nvidia GeForce 940m

10.2.2 Instalación de los módulos necesarios en Python

Para la instalación de todas las librerías de *Python* es necesario utilizar PIP (Paquete de instalación PIP).
Con esta utilidad podemos realizar la instalación o desinstalación de los diferentes módulos en *Python* tan solo lanzando una línea de comandos.
Como en esta herramienta se ha utilizado *Python* 3.7.1, esta utilidad ya viene incluida con el paquete principal.
Para todos los casos que mostraremos a continuación es necesario que nos situemos en el directorio donde se encuentra instalado *Python*.
Para ello sería necesario escribir el siguiente comando en la consola, en mi caso sería:

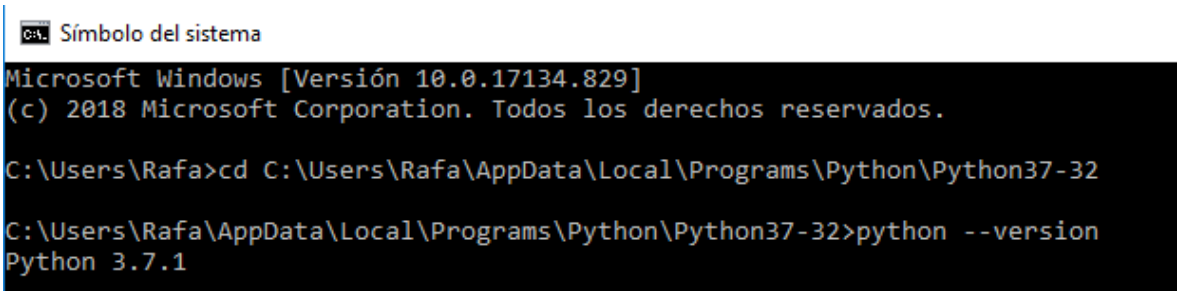


Figura 90. Directorio Python.
Fuente: Elaboración propia.

Ya nos encontramos situados en el directorio donde está instalado *Python*, ahora es necesario ir hasta la carpeta *Scripts*, para esto lanzamos el siguiente comando:

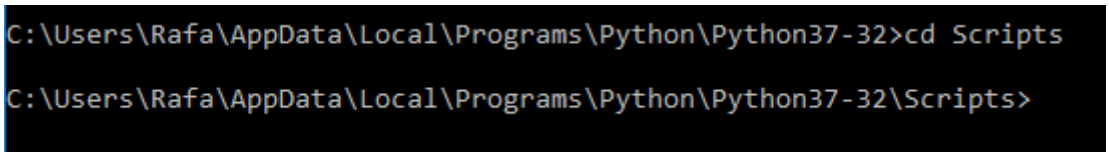


Figura 91. Directorio Scripts.
Fuente:Elaboración propia.

Ya nos encontramos en el directorio correcto para realizar las diferentes instalaciones con PIP, por lo que se detallaran en los siguientes apartados los comandos necesarios para instalar los módulos utilizados en esta herramienta.

10.1.1 Instalación de NLTK

Para realizar la instalación de la librería *NLTK* es necesario lanzar el siguiente comando:

Pip install nltk

Comenzará la descarga de todo el paquete *NLTK* y se procederá a la instalación de este de forma automática. Finalmente, cuando se complete aparecerá el mensaje *Sucesfully installed* como se muestra en la siguiente imagen.

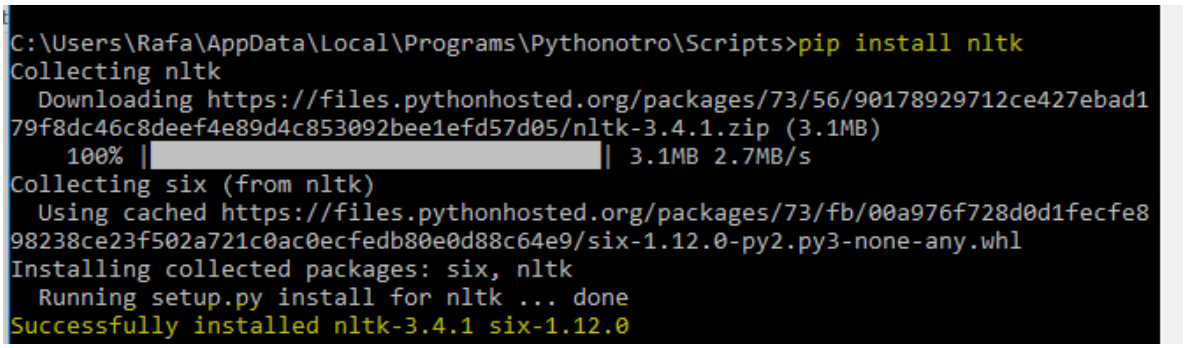


Figura 92. Instalación NLTK.
Fuente: Elaboración propia.

Una vez instalada la librería *NLTK*, podemos hacer uso de los siguientes módulos:

a) *Word Tokenize*

Para la utilización, tan solo necesitamos importarla en nuestro código con tan solo incluir la siguiente línea:

```
from nltk.tokenize import word_tokenize
```

Figura 93. Importar Word tokenize.
Fuente: Elaboración propia.

b) StopWords

Para la utilización de este módulo, tan solo necesitamos importarlo en nuestro código con la siguiente línea:

```
from nltk.corpus import stopwords
```

Figura 94. Importar Stop Words.
Fuente: Elaboración propia.

c) SnoballStemmer

Para la utilización de este módulo, tan solo necesitamos importarlo en nuestro código con la siguiente línea:

```
from nltk.stem import SnowballStemmer
```

Figura 95. Importar Snowball Stemmer.
Fuente: Elaboración propia.

d) BeautifulSoup

Para la utilización de este módulo, tan solo necesitamos importarlo en nuestro código con la siguiente línea:

```
from bs4 import BeautifulSoup
```

Figura 96. Importar Beautiful Soup.
Fuente: Elaboración propia.

10.1.2 Python Imaging Library (PIL)

Para la correcta instalación de la librería es necesario lanzar el siguiente comando:

Pip install Pillow

Comenzará la descarga de todo el paquete PIL donde se incluyen todos los módulos necesarios para la creación y edición de imágenes.

Se realizará la instalación automática de la librería, una vez que se complete, aparecererá el mensaje:

Sucesfully installed Pillow, como se muestra en la siguiente imagen.

```
C:\Users\Rafa\AppData\Local\Programs\Pythonotro\Scripts>pip install pillow
Collecting pillow
  Using cached https://files.pythonhosted.org/packages/81/1a/6b2971adc1bca55b9a53
ed1efa372acff7e8b9913982a396f3fa046efaf8/Pillow-6.0.0.tar.gz
Installing collected packages: pillow
  Running setup.py install for pillow ... done
Successfully installed Pillow-6.0.0
```

Figura 97. Instalación Pillow
Fuente: Elaboración propia.

Una vez instalada la librería PIL podemos hacer uso de los siguientes módulos:

a) Image

Para la utilización de este módulo, tan solo es necesario incluir la siguiente línea en nuestro código:

```
from PIL import Image
```

Figura 98. Importar Image.
Fuente: Elaboración propia.

b) ImageTk

Para la utilización de este módulo, tan solo es necesario incluir la siguiente línea en nuestro código:

```
from PIL import ImageTk
```

Figura 99. Importar ImageTK.
Fuente: Elaboración propia.

10.1.3 Matplotlib

Para la correcta instalación de la librería es necesario lanzar la siguiente línea:

Python –m pip install –U matplotlib

Comenzará la descarga de todo el paquete *Matplotlib* donde se incluyen todos los módulos necesarios para la creación y edición de gráficos

Se realizará la instalación automática de la librería, una vez que se complete, aparecerá el mensaje:

Sucesfully installed Matplotlib, como se muestra en la siguiente imagen:

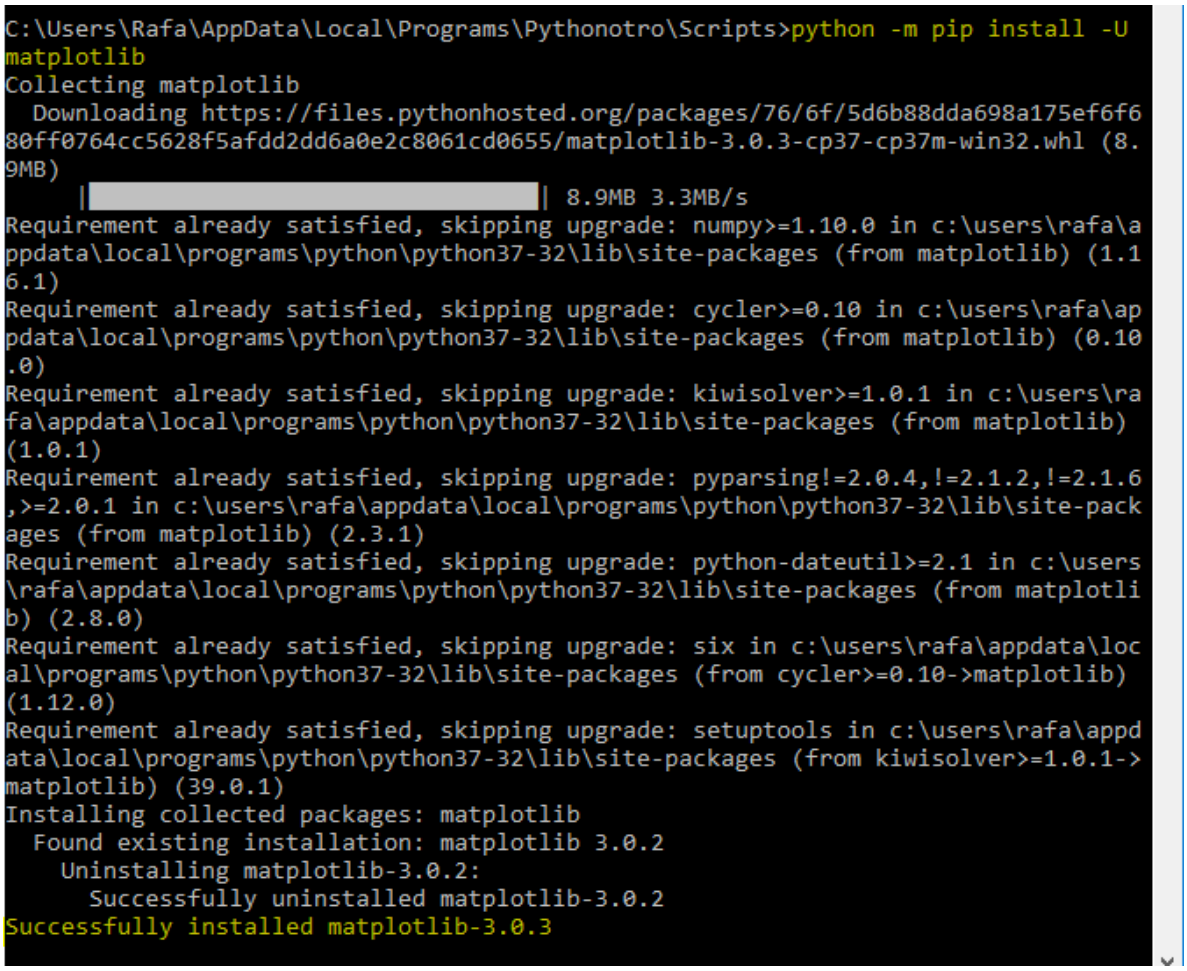


Figura 100. Instalación Matplotlib.
Fuente: Elaboración propia.

Una vez instalada la librería, podemos hacer uso del siguiente módulo:

a) Pyplot

Para la utilización de este módulo tan solo necesitamos incluir la siguiente línea en nuestro código:

```
import matplotlib.pyplot as plt
```

Figura 101. Importar Pyplot.

Fuente: Elaboración propia.

b) Numpy

Para la utilización de este módulo tan solo necesitamos incluir la siguiente línea en nuestro código:

```
import numpy as np
```

Figura 102. Importar Numpy.
Fuente: Elaboración propia.

10.1.4 Tweepy

Para la correcta instalación de la librería es necesario lanzar la siguiente línea de código:

Pip install tweepy

Comenzará la descarga y la instalación automática de toda la librería *Tweepy*, que finalizará con un *Sucesfully Installed*.

```
C:\Users\Rafa\AppData\Local\Programs\Pythonotro\Scripts>pip install tweepy
Collecting tweepy
  Using cached https://files.pythonhosted.org/packages/d5/5f/daac4b4e9b30d7d2a6fd
d16a880ff79f27918fe388e4dfc1983dec3a9876/tweepy-3.7.0-py2.py3-none-any.whl
Requirement already satisfied: six>=1.10.0 in c:\users\rafa\appdata\local\program
s\pythonotro\lib\site-packages (from tweepy) (1.12.0)
Collecting requests-oauthlib>=0.7.0 (from tweepy)
  Using cached https://files.pythonhosted.org/packages/c2/e2/9fd03d55ffb70fe51f58
7f20bcf407a6927eb121de86928b34d162f0b1ac/requests_oauthlib-1.2.0-py2.py3-none-any
.whl
Collecting PySocks>=1.5.7 (from tweepy)
  Downloading https://files.pythonhosted.org/packages/cd/18/102cc70347486e75235a2
9a6543f002cf758042189cb063ec25334993e36/PySocks-1.7.0-py3-none-any.whl
Collecting requests>=2.11.1 (from tweepy)
  Using cached https://files.pythonhosted.org/packages/7d/e3/20f3d364d6c8e5d2353c
72a67778eb189176f08e873c9900e10c0287b84b/requests-2.21.0-py2.py3-none-any.whl
Collecting oauthlib>=3.0.0 (from requests-oauthlib>=0.7.0->tweepy)
  Using cached https://files.pythonhosted.org/packages/16/95/699466b05b72b94a41f6
62dc9edf87fda4289e3602ecd42d27fcaddf7b56/oauthlib-3.0.1-py2.py3-none-any.whl
Collecting idna<2.9,>=2.5 (from requests>=2.11.1->tweepy)
  Using cached https://files.pythonhosted.org/packages/14/2c/cd551d81dbe15200be1c
f41cd03869a46fe7226e7450af7a6545bfc474c9/idna-2.8-py2.py3-none-any.whl
Collecting certifi>=2017.4.17 (from requests>=2.11.1->tweepy)
  Downloading https://files.pythonhosted.org/packages/60/75/f692a584e85b7eaba0e03
827b3d51f45f571c2e793dd731e598828d380aa/certifi-2019.3.9-py2.py3-none-any.whl (15
8kB)
    100% |████████████████████████████████| 163kB 3.3MB/s
Collecting chardet<3.1.0,>=3.0.2 (from requests>=2.11.1->tweepy)
  Using cached https://files.pythonhosted.org/packages/bc/a9/01ffebfb562e4274b648
7b4bb1ddec7ca55ec7510b22e4c51f14098443b8/chardet-3.0.4-py2.py3-none-any.whl
Collecting urllib3<1.25,>=1.21.1 (from requests>=2.11.1->tweepy)
  Downloading https://files.pythonhosted.org/packages/01/11/525b02e4acc0c747de8b6
ccdab376331597c569c42ea66ab0a1dbd36eca2/urllib3-1.24.3-py2.py3-none-any.whl (118k
B)
    100% |████████████████████████████████| 122kB 3.3MB/s
Installing collected packages: oauthlib, idna, certifi, chardet, urllib3, request
s, requests-oauthlib, PySocks, tweepy
Successfully installed PySocks-1.7.0 certifi-2019.3.9 chardet-3.0.4 idna-2.8 oaut
hlib-3.0.1 requests-2.21.0 requests-oauthlib-1.2.0 tweepy-3.7.0 urllib3-1.24.3
```

Figura 103. Instalación Tweepy.
Fuente: Elaboración propia.

Para utilizar el módulo *Tweepy* tan solo necesitamos incluir la siguiente línea en nuestro código:

```
import tweepy
```

Figura 104. Importar Tweepy.
Fuente: Elaboración propia.