Informe del sistema de Recuperación de Información

# Introducción

El presente informe detalla la implementación de un sistema de recuperación de información trabajado sobre el Corpus de Google. Se hizo una implementación modular de cada parte requerida para mayor rapidez de la visualización de resultados; así como una implementación de interfaz para su interacción mas adecuada.

# Implementación

La implementación se puede encontrar en <https://github.com/TheReverseWasp/TBD-Boolean-IR_Model> junto con este informe y los resultados obtenidos. Esta misma se encuentra dividida de la siguiente manera:

|  |  |
| --- | --- |
| **Carpeta** | **Descripción** |
| Aplicación | Carpeta donde se encuentra la implementación de la interfaz final junto con su funcionalidad. |
| Datos | Carpeta donde se encuentra el Corpus asi como las carpetas numeradas, estas no se subieron al Github por motivos de protección a los datos utilizados, sin embargo si se desea replicar, se puede utilizar la estructura del archivo de organización de carpeta anidado en esta carpeta. |
| Preprocesamiento | En esta carpeta se encuentran scripts de preprocesamiento. Para poder correr la versión de interfaz se debe compilar hasta el archivo numerado 04. El resto de los scripts son opcionales con uso final para la visualización de los resultados en terminal. |
| Resultados | Carpeta donde se almacenan los resultados de la implementación con interfaz. |

## Preprocesamiento

Para la parte del preprocesamiento se por hacer una implementación modular, esta tiene como objetivo separar por tareas la limpieza, corrección y separación de los datos. La organización de los scripts dentro de la carpeta de preprocesamiento esta organizada de la siguiente manera:

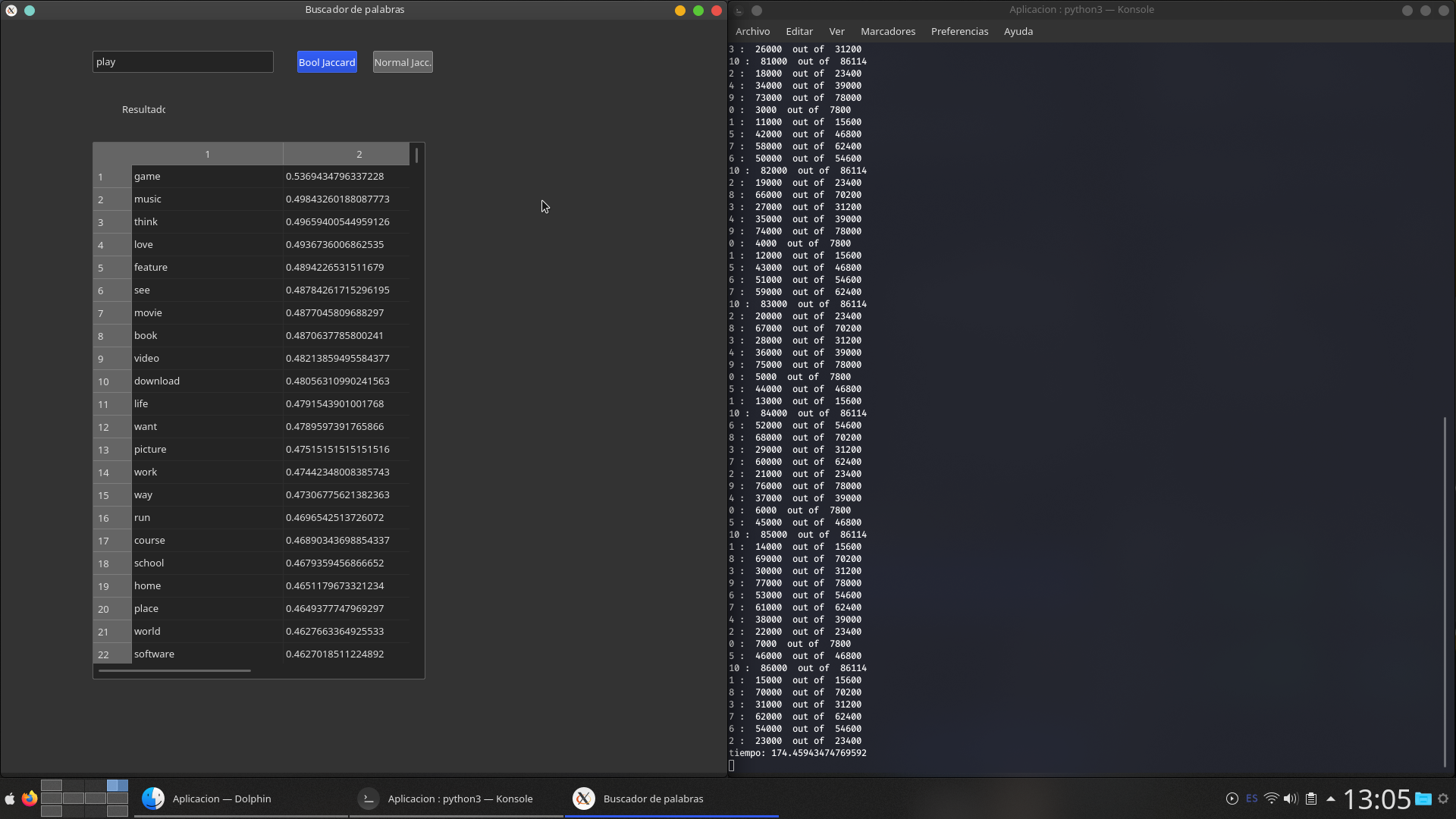
|  |  |  |
| --- | --- | --- |
| Script | Descripción | Tiempo de ejecución |
| 00\_dic\_get,py,py | Obtiene un diccionario a partir de la carpeta de datos sin stop-words. | Minutos - horas |
| 01\_clean\_data\_01.py | Limpia los datos de símbolos. | Minutos - horas |
| 01\_clean\_data\_02.py | Limpia los datos de stop words. | Minutos-horas |
| 02\_lemma\_separator.py | Deriva las palabras del paso anterior en su palabra lemmatizada. | Días - semanas |
| 03\_en\_only.py | Filtra solo los pares de palabra en ingles del paso anterior. | Horas - días |
| 04\_words\_data\_s.py | Separa los archivos planos en jsons; 1/palabra. Dentro de cada uno se encuentra la palabra relacionada con su calificación y su numero de ocurrencias | Minutos-horas |
| 05\_words\_simmil.py (opcional) | Ejecuta la similaridad de todos contra todos y lo guarda en una carpeta adicional en la carpeta Datos. | Semanas-Meses |
| Complementos.py | Archivo de funciones utilizadas en la generalidad de los scripts. | ------------------------------- |
| all\_dist\_word.py  gen\_filenames.py  search\_concurrency.py  tester.py | Archivos de ejecución en terminal. No son utilizados en esta implementación. | ------------------------------- |

La carpeta Aplicación esta dividida de la siguiente manera:

|  |  |
| --- | --- |
| Archivo | Función |
| main.py | Script de interacción con la interfaz donde se ejecuta la funcionalidad. |
| helpers.py | Script de funciones de soporte para los botones de la interfaz. |
| complementos.py | Script de funciones base utilizadas en helpers.py |

## Resultados

Los resultados de esta implementación pueden visualizarse en ´la carpeta de Resultados después de correr la aplicación. Cada calculo demora entre 100 y 200 segundos, la implementación fue acelerada con threads siendo su tiempo de ejecución normal entre 350 a 600 segundos. Los resultados de la búsqueda por modelo booleano se pueden ver en la Imagen 1, donde se ve las palabras mas cercanas a la palabra play con un tiempo de búsqueda de 174 segundos aproximadamente.

Imagen 1: Resultados de la búsqueda de la palabra play con tiempo de 174 segundos.

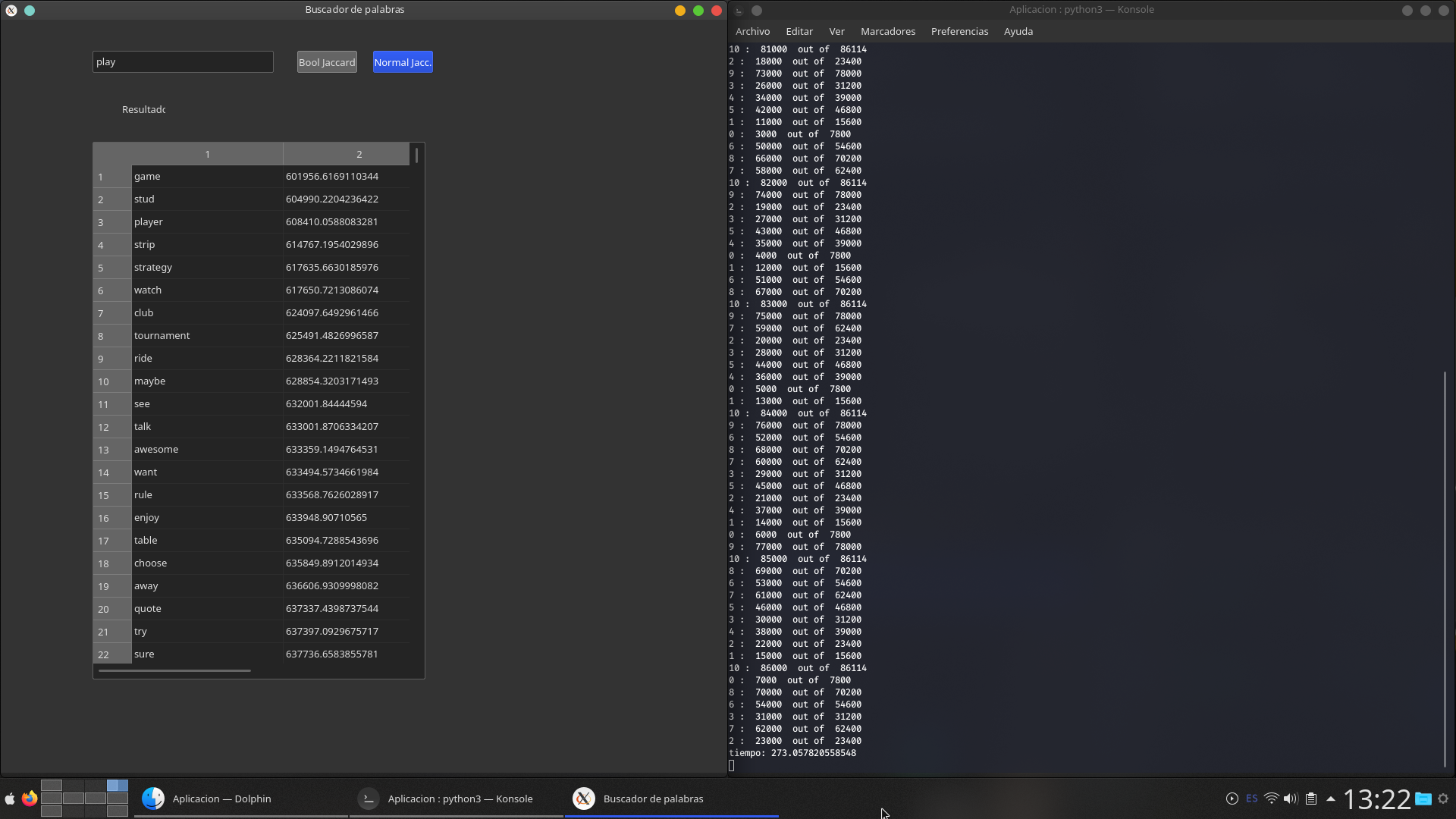
# **Otros agregados**

En nuestra implementación se hizo la búsqueda por características, es decir, utilizando los datos que el modelo booleano no utiliza, recordemos que la característica de cada termino en el json de cada palabra es de:

**palabra: [ocurrencia sumada, Nro de ocurrencias]**

Por lo que se puede aprovechar estos datos haciendo una distancia entre dos palabras, o, en este caso, uno contra todos. Los resultados de esta búsqueda se pueden observar en la Imagen 2, con la palabra play con un tiempo estimado de 273 segundos aproximadamente con resultados similares al de la búsqueda booleana.

Para el caso de esta implementación se agrego una penalización a las palabras que no guardan nada de coincidencia en ambos diccionarios. Así como una aplicación de manera similar al modelo booleano con Jaccard solo considerando las palabras que ambos diccionarios tienen coincidencia.

Imagen 2: Resultados de la búsqueda por distancia común de la palabra play.

# Conclusiones

En el presente informe se describió la implementación del sistema de recuperación de información utilizando un modelo booleano con indice de Jaccard y un modelo de distancias común con aplicables del indice de Jaccard. Se entendió la necesidad de implementar utilizando módulos en búsqueda de un PMV y de la presencia de Information Retrieval hoy en día.