

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 así 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	Obtiene un diccionario a partir de la carpeta de datos sin stop-	Minutos - horas

	words.	
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 lematizada.	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

Por: Ricardo Manuel Lazo Vásquez (TheReverseWasp - Github) Universidad Católica San Pablo

palabras mas cercanas a la palabra play con un tiempo de búsqueda de 174 segundos aproximadamente.

	1	2
1	game	0.5369434796337228
2	music	0.4984326018808773
3	think	0.49659400544959126
4	love	0.493673600686235
5	feature	0.4894226531511679
6	see	0.48784261715296195
7	movie	0.4877045809688297
8	book	0.4870637785800241
9	video	0.48213859495584377
10	download	0.48056310990241563
11	life	0.4791543901001768
12	want	0.4789597391765866
13	picture	0.47515151515151516
14	work	0.47442348008385743
15	way	0.47306775621382363
16	run	0.4696542513726072
17	course	0.46890343698854337
18	school	0.467935945686652
19	home	0.4651179673321234
20	place	0.4649377747969297
21	world	0.4627663364925533
22	software	0.4627018511224892

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

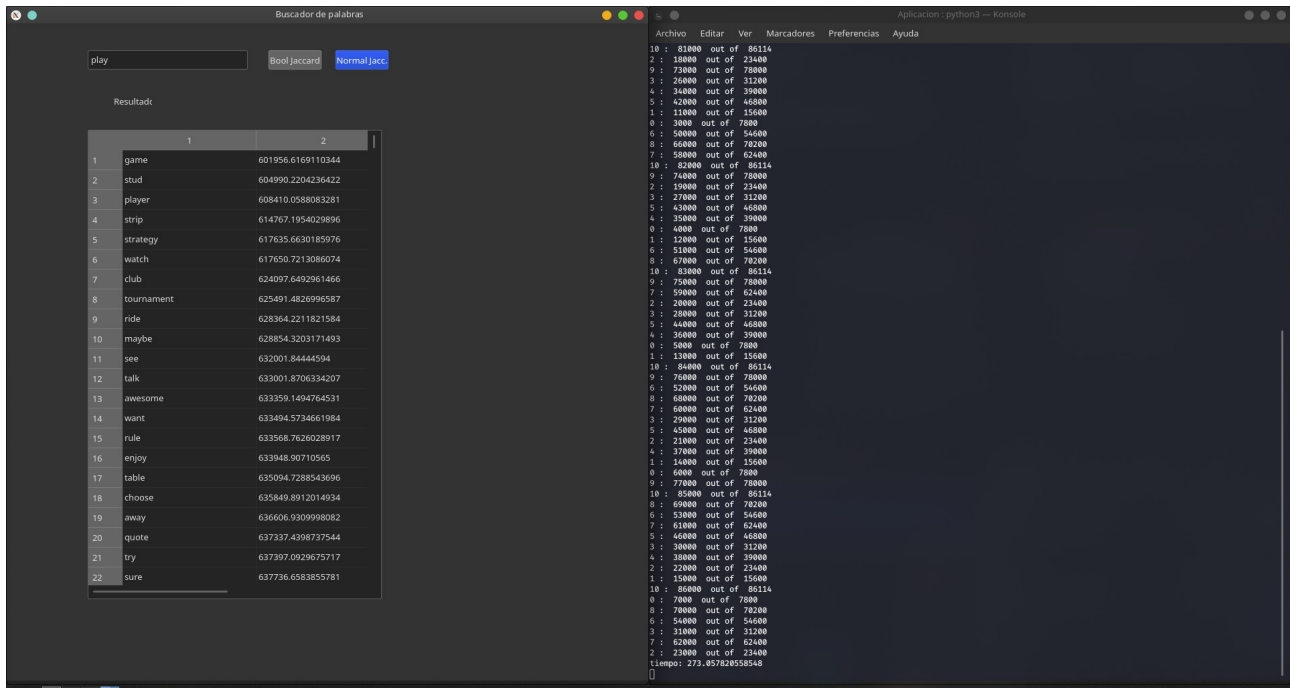
Otros agregados

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.



Buscador de palabras

play Bool jaccard Normal jacc

Resultados

	1	2
1	game	601956.6169110344
2	stud	604990.2204236422
3	player	608410.0588083281
4	strip	614767.1954029896
5	strategy	617635.6630185976
6	watch	617650.7213086074
7	club	624097.6492961466
8	tournament	625491.4826996587
9	ride	628364.2211821584
10	maybe	628854.3203171493
11	see	632001.84444594
12	talk	633001.8706334207
13	awesome	633359.1494764531
14	want	633494.5734661984
15	rule	633568.7626028917
16	enjoy	633948.90710565
17	table	635094.7288543696
18	choose	635849.8912014934
19	away	636606.9309998082
20	quote	637337.4398737544
21	try	637397.0929675717
22	sure	637736.6583855781

Archivo Editar Ver Marcadores Preferencias Ayuda

10 : 81000 out of 86114
2 : 18000 out of 23400
9 : 73000 out of 78000
3 : 20000 out of 31200
4 : 34000 out of 39000
5 : 42000 out of 46800
1 : 11000 out of 15600
0 : 3000 out of 7800
6 : 50000 out of 54600
8 : 66000 out of 78200
7 : 58000 out of 62400
10 : 62000 out of 86114
9 : 74000 out of 78000
2 : 19000 out of 23400
3 : 27000 out of 31200
5 : 43000 out of 46800
4 : 35000 out of 39000
0 : 4000 out of 7800
1 : 12000 out of 15600
6 : 51000 out of 54600
8 : 67000 out of 78200
10 : 83000 out of 86114
9 : 75000 out of 78000
7 : 59000 out of 62400
2 : 20000 out of 23400
3 : 28000 out of 31200
5 : 44000 out of 46800
4 : 36000 out of 39000
0 : 5000 out of 7800
1 : 13000 out of 15600
10 : 84000 out of 86114
9 : 76000 out of 78000
6 : 52000 out of 54600
8 : 68000 out of 78200
7 : 60000 out of 62400
3 : 29000 out of 31200
5 : 45000 out of 46800
2 : 21000 out of 23400
4 : 37000 out of 39000
1 : 14000 out of 15600
0 : 6000 out of 7800
9 : 77000 out of 78000
10 : 85000 out of 86114
8 : 69000 out of 78200
6 : 53000 out of 54600
7 : 61000 out of 62400
5 : 46000 out of 46800
3 : 30000 out of 31200
4 : 38000 out of 39000
2 : 22000 out of 23400
1 : 15000 out of 15600
10 : 86000 out of 86114
0 : 7000 out of 7800
8 : 70000 out of 78200
6 : 54000 out of 54600
3 : 31000 out of 31200
7 : 62000 out of 62400
2 : 23000 out of 23400
Limpio: 273.85782858548
[]

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 índice de Jaccard y un modelo de distancias común con aplicables del índice 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.