

## Ejercicio 1

El objetivo del ejercicio 1 es obtener documentos que traten sobre un determinado tema que introducirá el usuario (obviamente sin tener que contener esa palabra necesariamente). Para las pruebas de esta práctica se ha utilizado el alcoholismo (alcoholism) aunque el programa soporta cualquier término.

El script comienza haciendo una serie de consultas utilizando el parámetro gnd (Google normalized distance) para obtener palabras que se almacenarán en un set (se ha utilizado esa estructura de datos para que no se repitan las palabras). Las palabras más relacionadas con la palabra que ha introducido el usuario es lo que contendrá el set, que se imprimirá por pantalla al ejecutar el script.

```
#####
# 1ª Búsqueda: Sacar términos relacionados con la palabra
#####

# Ésta consulta da una lista de palabras relacionadas con alcoholismo
# (o la palabra que pongamos en lugar de alcoholism),
# sin palabras vacías como and o the que no aportan nada
results = es.search(
    index="reddit-mentalhealth",
    body = {
        "size":0,
        "query":{
            "query_string":{
                "default_field":"selftext",
                "query":palabra
            }
        },
        "aggs": {
            "TerminosSignificativos":{
                "significant_terms":{
                    "field":"selftext",
                    "size":10,
                    "gnd":{}
                }
            }
        }
    }
)

# Creamos un set para almacenar los terminos sin que se repitan
vectorTerminos = set([])
```

El resultado (results) está en formato JSON.

Una vez sacados los términos, lo que se hace es simplemente pasarlos a un string (palabras) para poder introducirlos en una consulta que nos devolverá los 25 posts que más de esas palabras coincidan.

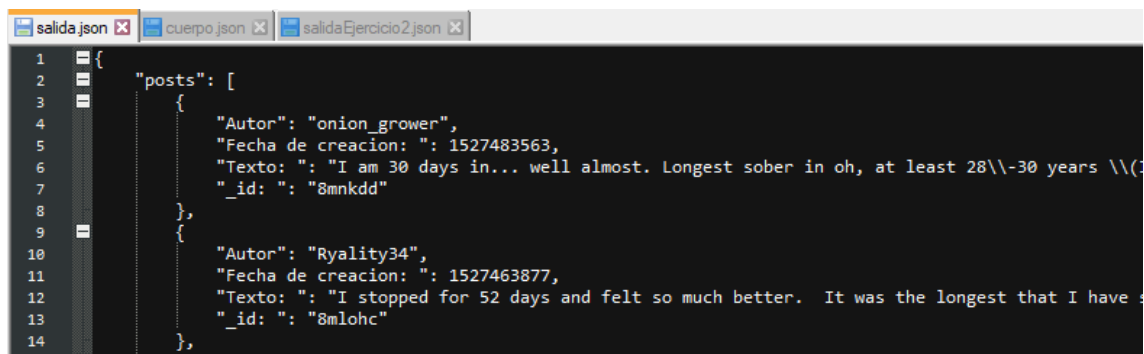
```

results2=es.search(
    index="reddit-mentalhealth",
    body = {
        "size":25, # Obtenemos los 25 primeros resultados
        "query": {
            "match": {
                "selftext": {
                    "query": palabras,
                    "operator": "or"
                }
            }
        }
    }
)

```

Una vez que ya tenemos los términos ya es cuestión de utilizar la librería json para crear el fichero de salida.

### Análisis del fichero de salida y su precisión:



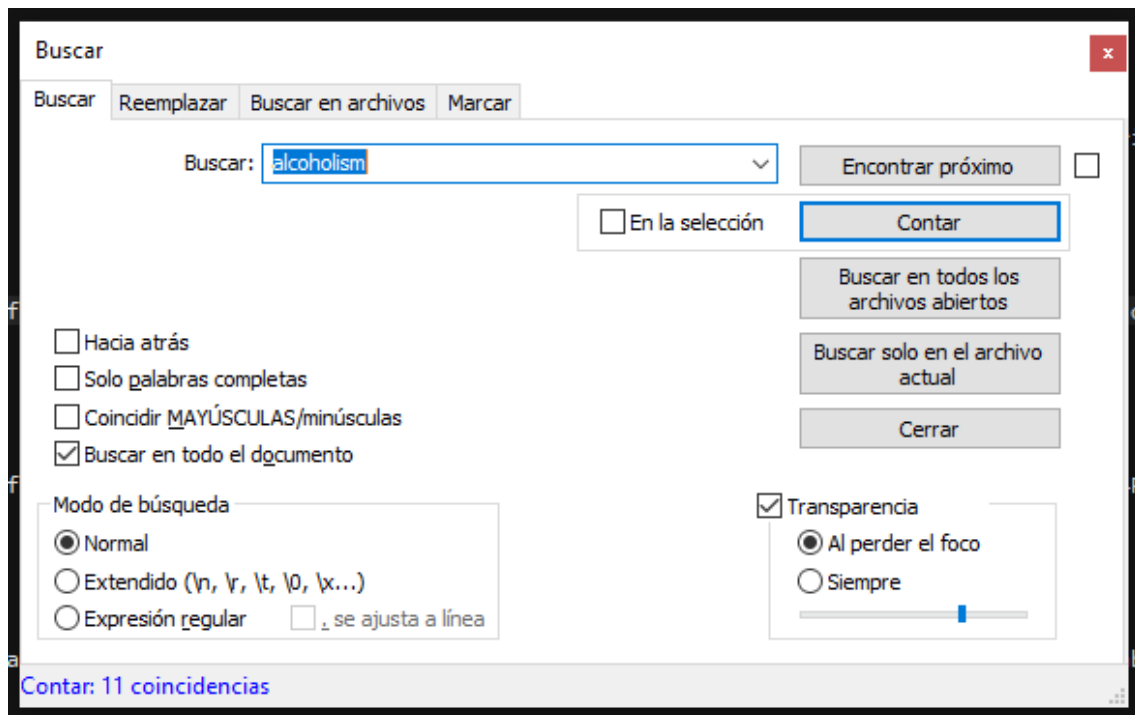
```

1  {
2    "posts": [
3      {
4        "Autor": "onion_grower",
5        "Fecha de creacion": "1527483563",
6        "Texto": "I am 30 days in... well almost. Longest sober in oh, at least 28\\-30 years \\(3",
7        "_id": "8mnkdd"
8      },
9      {
10       "Autor": "Ryality34",
11       "Fecha de creacion": "1527463877",
12       "Texto": "I stopped for 52 days and felt so much better. It was the longest that I have s",
13       "_id": "8mlohC"
14     }
15   ]
16 }

```

Una vez examinados los 25 documentos del fichero de salida, se comprueba que la precisión es del 100% ya que no hay ningún documento que no hable sobre un problema con la bebida.

Cabe destacar que hay bastantes documentos que no contienen la palabra que introdujo el usuario (alcoholism). De hecho, sólo hay 11 veces que aparece la palabra en el documento:



Para obtener la precisión del 100% de documentos relevantes, se han utilizado los siguientes valores:

- Número de palabras que se obtienen en la primera extracción de términos relacionados con el alcoholismo: 10

```
#####
# 1ª Búsqueda: Sacar términos relacionados con la palabra
#####

# Ésta consulta da una lista de palabras relacionadas con alcoholismo
# (o la palabra que pongamos en lugar de alcoholism),
# sin palabras vacías como and o the que no aportan nada
results = es.search(
    index="reddit-mentalhealth",
    body = {
        "size":0,
        "query":{
            "query_string":{
                "default_field":"selftext",
                "query":palabra
            }
        },
        "aggs": {
            "TerminosSignificativos":{
                "significant_terms":{
                    "field":"selftext",
                    "size":10,
                    "gnd":{}
                }
            }
        }
    }
)
```

(En las siguientes dos expansiones también se utiliza 10. Al final no salen 30 porque obviamente se repiten términos y acaban saliendo unos 20 distintos para el término alcoholism).

Si se aumenta el número de términos a 100 por ejemplo empiezan a salir palabras que no tienen realmente mucho que ver con el alcoholismo, y por tanto los posts que se recuperan no tratan sobre el alcoholismo en su totalidad.

- Número de documentos que se retornan: 25

```
results2=es.search(  
    index="reddit-mentalhealth",  
    body = {  
        "size":25, # Obtenemos los 25 primeros resultados  
        "query": {  
            "match": {  
                "selftext": {  
                    "query": palabras,  
                    "operator": "or"  
                }  
            }  
        }  
    }  
)
```

Se ha probado a retornar 100 documentos pero ya había alguno que no trataba de alguien que está hablando del alcoholismo, así que se dejó en 25 para lograr una precisión del 100% de documentos relevantes

## Ejercicio 2

Las consultas MLT (More like this) por debajo funcionan de la siguiente manera:

- Se les pasa como parámetros una serie de documentos (ya sea por ejemplo en forma de un `_id` de un documento que se encuentra en el índice, en este caso `reddit-mentalhealth`, o en forma de texto plano)
- Obtienen de ese documento o serie de documentos una serie de palabras representativas de esos documentos.
- A partir de esa serie de palabras, hacen una consulta que devuelve los documentos que más de esas palabras coincidan.

Por tanto, MLT lo que hace por debajo es una expansión de términos, pero en lugar de hacerla a partir de un término como en el ejercicio 1, lo hace a partir de una serie de documentos.

Para emular la expansión de términos del ejercicio 1 con MLT se puede coger uno o varios documentos obtenidos en la salida del ejercicio 1 (hablan sobre el alcoholismo, o la palabra que introduzca el usuario) y pasárselos como entrada a la consulta MLT para que saque otra serie de documentos relacionados.

Los términos que la consulta MLT ha considerado representativos no aparecen porque es un proceso interno, lo que nos devuelve la consulta MLT son los documentos relacionados.

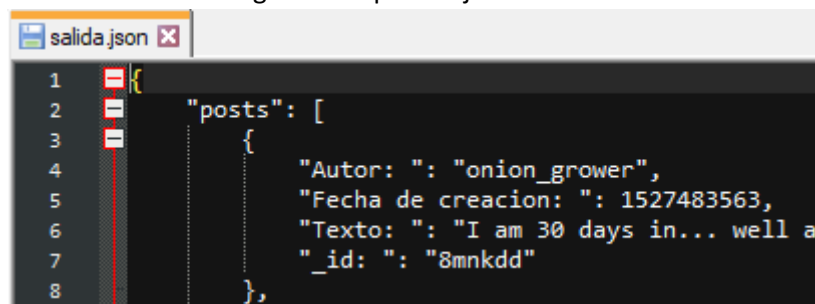
No hay problema porque el objetivo del ejercicio no es saber qué términos cree la consulta MLT que son representativos de los documentos de entrada, sino simplemente obtener los documentos de salida.

Por tanto, los pasos a seguir en este ejercicio son los siguientes:

- Obtener del archivo JSON generado por el ejercicio 1 los `_id` de los documentos (Para ello hay que modificar ligeramente el ejercicio 1 para que también añada los `_id` de cada documento, el objetivo es poder hacer el ejercicio 1 y el 2 para cualquier término, no solo para alcoholismo.)

```
salida['posts'].append({
  'Autor: ':i["_source"].get("author"),
  'Fecha de creacion: ':i["_source"].get("created_utc"),
  'Texto: ':i["_source"].get("selftext"),
  '_id: ':i["_source"].get("id")
})
```

Muestra del archivo generado por el ejercicio 1:



```
1  {
2    "posts": [
3      {
4        "Autor: ": "onion_grower",
5        "Fecha de creacion: ": 1527483563,
6        "Texto: ": "I am 30 days in... well a
7        "_id: ": "8mnkdd"
8      },
```

- Pasarle los `_id` a la consulta MLT, un pequeño ejemplo:

```
1 {
2   "query": {
3     "more_like_this" : {
4       "like" : [
5         {
6           "_index" : "reddit-mentalhealth",
7           "_id" : "8jlc4"
8         }
9       ],
10      "min_term_freq" : 1,
11      "max_query_terms" : 12
12    }
13  }
14 }
```

Esa consulta devuelve documentos parecidos al "8jlc4"

En el script lo que se hace es leer los id's del fichero generado por el ejercicio1 mediante la librería json:

```
# Leer el archivo json que ha generado el ejercicio 1
# y obtener un vector con todos los id de los documentos

vectorId = set([])

with open('salida.json') as file:
    data = json.load(file)
    for documento in data['posts']:
        vectorId.add(documento['_id: '])
```

Una vez ya tenemos los id's de los documentos, hay que crear el cuerpo de una consulta parecida a la que hay más arriba de ejemplo en cerebro. El cuerpo es en json:

```
# Ya tenemos un vector con los id's de los documentos que generó
# el ejercicio 1

# Ahora hacemos una consulta MLT que obtenga documentos parecidos
# a esos pasándole el id

# Para ello tenemos que crear el cuerpo de la consulta (en JSON)
body={'query':{'more_like_this':{'like':[]}}}
body['size']=15 # Obtener los 15 documentos más relevantes
for i in range(len(vectorId)):
    body.get('query').get('more_like_this').get('like').append({
        '_index': "reddit-mentalhealth",
        '_id': vectorId.pop()
    })

# Guardar el archivo con formato json (para verlo y comprobar que está bien)
with open('cuerpo.json', 'w') as file:
    json.dump(body, file, indent=4)
```

```
1  {
2    "query": {
3      "more_like_this": {
4        "like": [
5          {
6            "_index": "reddit-mentalhealth",
7            "_id": "8gfv5"
8          },
9          {
10           "_index": "reddit-mentalhealth",
11           "_id": "8mloh"
12         },
13         {
14           "_index": "reddit-mentalhealth",
15           "_id": "8j1jmi"
16         },
17         ...
18       ],
19       "size": 15
20     }
21   }
22 }
```

(Se le añade el parámetro size al final para que sólo retorne los 15 primeros archivos)

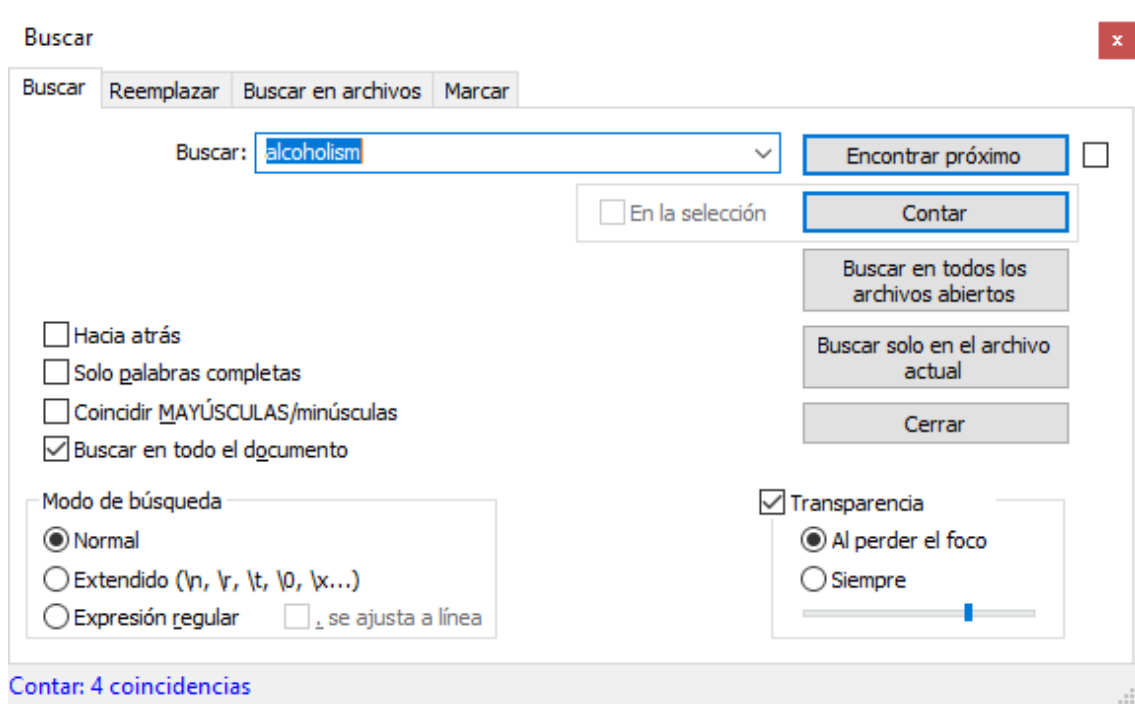
Como se puede ver en el archivo json que ha generado, es una consulta con el mismo formato que la que hay más arriba en cerebro. Ya solo es cuestión de ejecutarla:

```
# Ya tenemos el cuerpo de la consulta, sólo hay que crear la query ahora
# y ejecutarla
results = es.search(
    index="reddit-mentalhealth",
    body = body
)
```

- Una vez ya obtenidos los documentos que genera la consulta MLT hay que almacenarlos en un archivo JSON, lo cual se hace exactamente igual que en primer ejercicio, mediante la librería json se crea un archivo y se van añadiendo los documentos, no he creído conveniente explicar esta parte ya que el objetivo de este ejercicio es entender el funcionamiento de las consultas more like this.

#### Análisis del fichero de salida y su precisión:

Mediante el parámetro “size”: 15 en el cuerpo de la consulta MLT (para que devuelva 15 documentos diferentes) se obtiene una precisión del 100% de documentos relevantes, con un total de 4 coincidencias de la palabra alcoholism:



Cabe destacar que hay documentos que se repiten, pero es porque salen repetidos en el índice de reddit-mentalhealth, pero en realidad son posts distintos porque tienen un atributo `_id` en el índice distinto, lo que me hace pensar que la persona ha copiado y pegado el contenido y publicado varios posts, probablemente en varios subforos, ya que también las fechas son distintas. Eso sí, en cada una de las salidas no se repiten, cada post de cada salida es diferente.

Por ejemplo:

```
{
  "Autor": "throwaway204388",
  "Fecha de creacion": "1526608759",
  "Texto": "You know, I wish I could drink like a normal person. I laughed",
  "_id": "8k9nwz"
}
```

En salida.json (la salida del ejercicio 1)

```
{
  "Autor": "throwaway204388",
  "Fecha de creacion": "1526609935",
  "Texto": "You know, I wish I could drink like a normal person. I laughed at",
  "_id": "8k9s3p"
},
```

En salidaEjercicio2.json

Esto se podría resolver controlando que los selftext sean distintos, pero realmente lo son, sólo cambia que hay algún carácter que cambia de los dos textos, así que sería bastante difícil diferenciar si un post es el mismo o no.

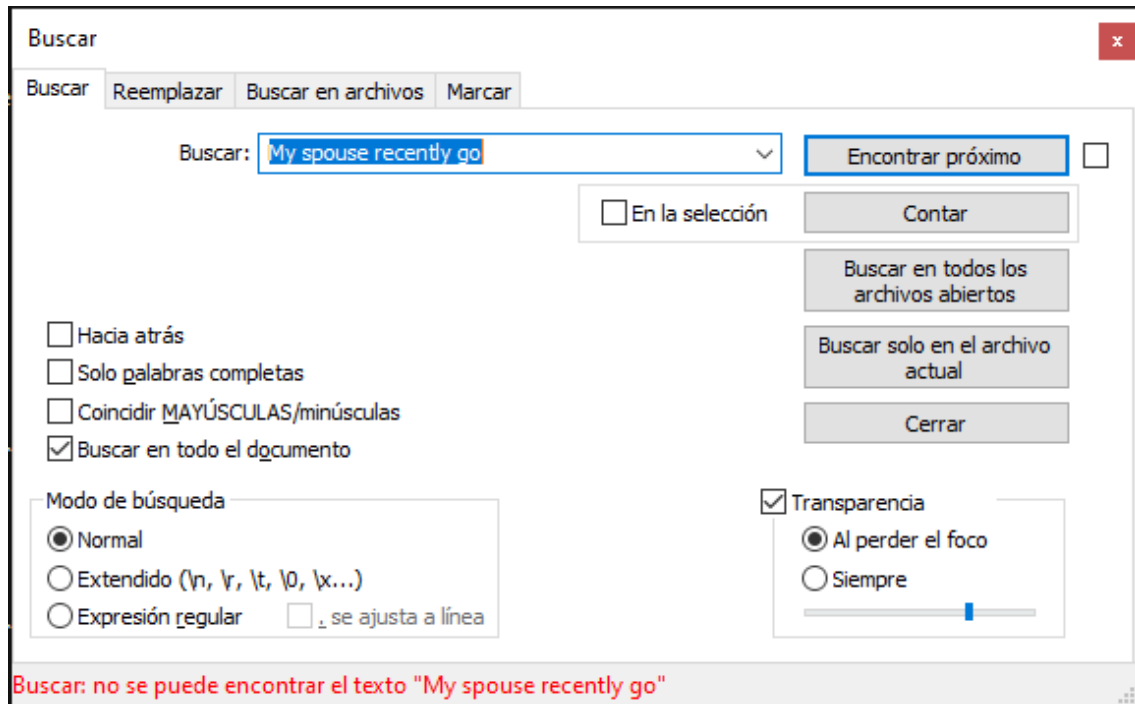
## Conclusión



Por otra parte, la consulta more like this se podría considerar exitosa ya que realmente ha encontrado documentos que hablan sobre problemas con la bebida que no estaban en el primer documento, como por ejemplo el siguiente post:

```
{
  "Autor": "BrucePhoenix",
  "Fecha de creacion": "1525477368",
  "Texto": "My spouse recently go a visa to live in the US and lives with me. That",
  "_id": "8h3n4o"
},
```

No aparece en la salida del ejercicio 1 porque buscando las cuatro primeras palabras, no sale:



Y trata sobre el alcoholismo, habla de un amigo suyo que bebe mucho y tiene problemas.

## Ejercicio 3

Se exponen los pasos seguidos para llegar a la solución.

1.- El ejercicio en cuestión es el siguiente:

Se desea obtener una lista exhaustiva de los medicamentos utilizados por los usuarios que aparecen en la colección.

Describir de manera detallada los pasos seguidos para explorar la colección, cómo se ha construido la consulta o consultas que han permitido llegar a una lista potencial y qué recursos externos se han empleado para validar la información obtenida.

¡Atención! No es necesaria la validación automática de la lista de medicamentos obtenida pero sí se valorará positivamente que se explore qué posibilidades de automatización existen al respecto.

Si se usa algún conocimiento experto de partida debe indicarse claramente en la documentación.

2.- Se ha empezado por construir una pequeña consulta en elasticsearch en cerebro donde sacar las claves para luego pasarlo a python usando un script.

The screenshot shows the Elasticsearch DevTools interface. The top bar includes navigation links (overview, nodes, rest, more), a refresh button, a 15sec timer, and the URL http://localhost:9200 [yellow].

The main area is divided into two panels. The left panel, titled 'previous requests', shows a REST client request for the index 'reddit-mentalhealth3/\_search' with a POST method. The request body is a JSON query:

```
1 {
2   "size": 0,
3   "query": {
4     "query_string": {
5       "query": "medicament OR prescrib"
6     }
7   },
8   "aggs": {
9     "Terminos": {
10      "significant_terms": {
11        "field": "selftext",
12        "size": 2000,
13        "end": {}
14      }
15    }
16  },
17  "sort": [
18    {
19      "aggregations.Terminos.buckets.doc_count": "asc"
20    },
21    "_score"
22  ]
23 }
```

The right panel shows the response JSON:

```
{
  "took": 142,
  "timed_out": false,
  "_shards": {
    "total": 1,
    "successful": 1,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": {
      "value": 1154,
      "relation": "eq"
    },
    "max_score": null,
    "hits": [ ]
  },
  "aggregations": {
    "Terminos": {
      "doc_count": 1154,
      "bg_count": 126801,
      "buckets": [
        {
          "key": "prescrib",
          "doc_count": 1065,
          "score": 0.9833482852313268,
          "bg_count": 1065
        },
        {
          "key": "adderal",
          "doc_count": 126,
          "score": 0.6739547316861874,
          "bg_count": 463
        },
        {
          "key": "psychiatrist",
          "doc_count": 174,
          "score": 0.6711029884115303,
          "bg_count": 1104
        },
        {
          "key": "dose",
          "doc_count": 159,
          "score": 0.6687912515629965,
          "bg_count": 919
        },
        {
          "key": "prescript",
          "doc_count": 103,
          "score": 0.6491132427823665,
          "bg_count": 473
        },
        {
          "key": "mg",
          "doc_count": 99,
          "score": 0.6462651396457675,
          "bg_count": 473
        }
      ]
    }
  }
}
```

At the bottom of the interface, there are buttons for 'cURL', 'format', and 'send'.

3.- Se ha buscado en google translate las palabras relacionadas con posible medicación, en español normalmente se recetan o se hace mención al medicamento, se ha construido la consulta en torno a eso

4.- Se ha buscado algún termino y comprobar que efectivamente, es una medicina, a lo largo de la documentación se ha usado Adderall, ya que por orden es de los primeros y sale como

elasticsearch[yellow] X Adderall at DuckDuckGo X Adderall: Uses, Dosage, Side Eff X +

← → ↻ 🏠 <https://www.drugs.com/adderall.html> [📄] [⋮] [📄] [S] [U] [S] [⋮]

**Drugs.com** Know more. Be sure. Search 🔍 [☰]

Treatment Options > ADHD > Adderall Print Share

# Adderall 🔊

**Generic Name:** [amphetamine and dextroamphetamine](#) (am FET a meen and DEX troe am FET a meen)  
**Brand Names:** *Adderall, Adderall XR, Mydayis*

Medically reviewed by [Kaci Durbin, MD](#) Last updated on Apr 25, 2019.

**Overview** Side Effects Dosage Professional Tips Interactions [More](#) ▾

## What is Adderall?

Adderall contains a combination of [amphetamine](#) and [dextroamphetamine](#). Amphetamine and dextroamphetamine are central nervous system stimulants that affect chemicals in the brain and nerves that contribute to hyperactivity and impulse control.

Adderall is used to treat attention deficit hyperactivity disorder (ADHD) and narcolepsy.

Adderall may also be used for purposes not listed in this medication guide.

medicina en varias páginas.

5.- A partir de aquí, se ha construido una consulta en python, ayudada por un compañero de grupo, donde obtener los diferentes términos que aparecen en la consulta y guardarlos en un fichero. Se ha usado el entorno de PyScript por su facilidad para debuggear.

Código del compañero:

```

# Ésta consulta da una lista de palabras relacionadas con alcoholismo
# (o la palabra que pongamos en lugar de alcoholism),
# sin palabras vacías como and o the que no aportan nada
results = es.search(
    index="reddit-mentalhealth",
    body={
        "size":0,
        "query":{
            "query_string":{
                "default_field":"selftext",
                "query":palabra
            }
        },
        "aggs":{
            "TerminosSignificativos":{
                "significant_terms":{
                    "field":"selftext",
                    "size":1000,
                    "gnd":{}
                }
            }
        }
    }
)

# Objetos json (cada uno tendrá una key con el termino)
terminosSignificativos = results["aggregations"]["TerminosSignificativos"]["buckets"]

# Creamos un vector para almacenar los terminos

```

No se ha guardado una copia de la primera versión de este código, se pone un ejemplo aproximado:

```

# Lanzamos el scaneo
results = es.search(
    index="reddit-mentalhealth3",
    body={
        "size":0,
        "query": {
            "query_string": {
                "default_field": "selftext",
                "query": "medicament OR prescrib"
            }
        },
        "aggs": {
            "Terminos": {
                "significant_terms": {
                    "field": "selftext",
                    "size": 1000,
                    "gnd": {}
                }
            },
            "sort": [
                {
                    "aggregations.Terminos.buckets.doc_count": "desc"
                }
            ]
        }
    }
)

f=open("scan-dump.txt","wb")
for r in range(0,len(results["aggregations"]["Terminos"]["buckets"])):
    f.write(results["aggregations"]["Terminos"]["buckets"][r]["key"].encode("UTF-8")+'\n'.encode("UTF-8"))

f.close()

```

6.- Tras abrir el resultado, se vio que aparecen las cantidades de la medicina, 10mg, 1g, 100mg... Como tiene fácil solución y solo esos términos tienen número, se añadió una sentencia if que comprueba si hay algún número en el término para que no se añadiese.

```

# Lanzamos el scaneo
results = es.search(
    index="reddit-mentalhealth3",
    body={
        "size":0,
        "query": {
            "query_string": {
                "default_field": "selftext",
                "query": "medicament OR prescrib"
            }
        },
        "aggs": {
            "Terminos": {
                "significant_terms": {
                    "field": "selftext",
                    "size": 1000,
                    "min_count": 1
                }
            }
        },
        "sort": [
            {
                "aggregations.Terminos.buckets.doc_count": "desc"
            }
        ]
    }
)

f=open("scan-dump.txt","wb")
for r in range(0,len(results["aggregations"]["Terminos"]["buckets"])):
    if(any(char.isdigit() for char in tmp)!=True):
        f.write(results["aggregations"]["Terminos"]["buckets"][n]["key"].encode("UTF-8")+'\n'.encode("UTF-8"))
f.close()

```

7.- Esta lista fue ordenada alfabéticamente y se fue comprobando diversos elementos para ver que algunos salían como medicinas (paracetamol, adderal, amoxicillin) y otros no (area, april,

1	abat
2	abdomen
3	abdomin
4	abilifi
5	abnorm
6	abrupt
7	ach
8	acid
9	add
10	adder
11	adderal
12	adderral
13	addit
14	adhd
15	administ
16	advers
17	advil
18	advis
19	affect
20	alev
21	allerg
22	allergi
23	allergist
24	alprazolam
25	altern
26	ambien
27	amitriptylin
28	amoxicillin
29	amphetamin
30	amsterdam
31	anem
32	anemia
33	anti
34	antiacid
35	antibiot
36	antidepress
37	antifung
38	antihistamin
39	antipsychot
40	apo
41	appetit

asian).

8.- En este punto se le pidió ayuda al profesor Daniel Gayo, quien amablemente me respondió con varias opciones

1 – Decir cómo se haría y enviar la lista tal cual

2 – Usar la API de wikidata

3 – Cruzar esta lista con otras que ya existen

Dado que había un pequeño ejemplo de uso y todavía se tenían algunos días hasta la entrega, se decidió tratar de usar la API de wikidata con Python

9.- Wikidata es una pagina que contiene elementos o palabras y se clasifican “según lo que son”. Por ejemplo, Paracetamol es un medicamento pero también un grupo musical. Adderal es un medicamento pero también tiene otras características.

paracetamol

**acetaminophen (Paracetamol)**  
common drug for pain and fever

**Paracetamol**  
scientific article published on July 1980

**Paracetamol**  
musical group

**Paracetamol**  
scientific article published on 01 February 1980

**Adderall**

pharmaceutical amphetamine brand

**The Adderall Diaries (Adderall Diaries)**

2015 American crime-thriller film directed by Pamela Rom...

**Adderall abuse on college campuses: a comprehens...**  
scientific article

10.- A partir de aquí se uso el consejo de Daniel Gayo del foro, comprobar que el termino es una instancia de meditación



**Sobre Wikidata**

de DANIEL GAYO AVELLO - 20 de noviembre de 2019, 12:46

Hola,

En el ejercicio 3 una solución elegante sería usar Wikidata.

Podéis consumirlo con facilidad vía REST.

Ejemplo trivial:

<https://www.wikidata.org/w/api.php?action=wbsearchentities&search=xanax&language=en>

La gracia del asunto sería en determinar que el término que descubristeis con Elasticsearch es una instancia de Medication.


Saludos, Dani

11.- Después de investigar, se conoció como trabaja Wikidata:

Wikidata tiene muchos datos y palabras, asigna a cada tipo de dato un identificador único, estos identificadores tienen propiedades, y las propiedades también están identificadas por su id. Ejemplo:

Adderall tiene de id Q935761 <https://www.wikidata.org/wiki/Q935761>

← → ↺ 🏠 <https://www.wikidata.org/wiki/Q935761>

**WIKIDATA**

Item

Discussion

# Adderall

(Q935761)

Tiene diferentes propiedades, una de ella instance of. Esta propiedad tiene el id P31, inicialmente se supo por la redirección del enlace (abajo del todo).

Permanent link

Page information

Concept URI

Cite this page

instance of

Property:P31

has active ingredient

https://www.wikidata.org/wiki/Property:P31

Dentro de esta propiedad, medicación tiene el id Q12140

What links here

Related changes

Special pages

Permanent link

Page information

Concept URI

Cite this page

Statements

instance of

medication

Q12140

0 references

Resumiendo, se obtiene la id de un termino y luego se mira que tenga como propiedad P31 y dentro algún valor sea Q12140.

12.- Se estudió la API de wikidata para ver como forma los datos y de que forma se puede trabajar con ellos, primeramente de forma manual con el ejemplo de Adderal. El proceso manual que se siguió es como sigue:

Para averiguar la id se usa el ejemplo del foro, solo que en lugar de Xana se puso Adderal  
<https://www.wikidata.org/w/api.php?action=wbsearchentities&search=adderal&language=en>



This is the HTML representation of the JSON format. HTML is good for debugging, but is unsuitable for application use.

Specify the *format* parameter to change the output format. To see the non-HTML representation of the JSON format, set *format=json*.

See the [complete documentation](#), or the [API help](#) for more information.

```
{
  "searchinfo": {
    "search": "adderal"
  },
  "search": [
    {
      "repository": "",
      "id": "Q935761",
      "concepturi": "http://www.wikidata.org/entity/Q935761",
      "title": "Q935761",
      "pageid": 886009,
      "url": "http://www.wikidata.org/wiki/Q935761",
      "label": "Adderall",
      "description": "pharmaceutical amphetamine brand",
      "match": {
        "type": "label",
        "language": "en",
        "text": "Adderall"
      }
    },
    {
      "repository": "",
      "id": "Q17097948",
      "concepturi": "http://www.wikidata.org/entity/Q17097948",
      "title": "Q17097948",
      "pageid": 18694341,
      "url": "http://www.wikidata.org/wiki/Q17097948",
      "label": "The Adderall Diaries",
      "description": "2015 American crime-thriller film directed by Pamela Romanowsky",
      "match": {
        "type": "alias",
        "language": "en",
        "text": "Adderall Diaries"
      },
      "aliases": [
        "Adderall Diaries"
      ]
    }
  ]
}
```

Aparecen varias ids, la que nos interesa es la primera, Q935761.

Con esta id, se averigua que constraints tiene, es decir, sus características.

<https://www.wikidata.org/w/api.php?action=wbcheckconstraints&id=Q935761>

This is the HTML representation of the JSON format. HTML is good for debugging, but is unsuitable for application use.

Specify the *format* parameter to change the output format. To see the non-HTML representation of the JSON format, set *format=json*.

See the [complete documentation](#), or the [API help](#) for more information.

```
{
  "wbcheckconstraints": {
    "Q935761": {
      "claims": {
        "P31": [
          {
            "id": "Q935761$ba8c8151-4ea2-7d53-cc13-04a40e9d36bd",
            "mainsnak": {
              "hash": "767ee58459dc663ef0085e81e75ea3abdbf593c5",
              "results": []
            }
          },
          {
            "id": "Q935761$BA03F2BA-8FED-460E-A63A-6BBD5936B8E8",
            "mainsnak": {
              "hash": "87cdd435c7bb91eadb3355615e99ee224aa44984",
              "results": []
            }
          }
        ]
      }
    }
  }
}
```

Y aquí falla, ya que no se puede saber si es un medicamento o no a través de este tipo de consulta, el hash es único y no hay ningún dato que identifique a todos los medicamentos.

13.- Se siguió investigando y se encontró otra función de la API que si tiene los datos que se necesitan. Wbgetentities

<https://www.wikidata.org/w/api.php?action=wbgetentities&ids=Q935761&languages=en>

This is the HTML representation of the JSON format. HTML is good for debugging, but is unsuitable for application use.

Specify the *format* parameter to change the output format. To see the non-HTML representation of the JSON format, set *format=json*.

See the [complete documentation](#), or the [API help](#) for more information.

```
{
  "entities": {
    "Q935761": {
      "pageid": 886009,
      "ns": 0,
      "title": "Q935761",
      "lastrevid": 1055154748,
      "modified": "2019-11-18T07:01:50Z",
      "type": "item",
      "id": "Q935761",
      "labels": {
        "en": {
          "language": "en",
          "value": "Adderall"
        }
      },
      "descriptions": {
        "en": {
          "language": "en",
          "value": "pharmaceutical amphetamine brand"
        }
      },
      "aliases": {
        "en": [
          {
            "language": "en",
            "value": "amphetamine mixed salts"
          }
        ]
      }
    }
  }
}
```

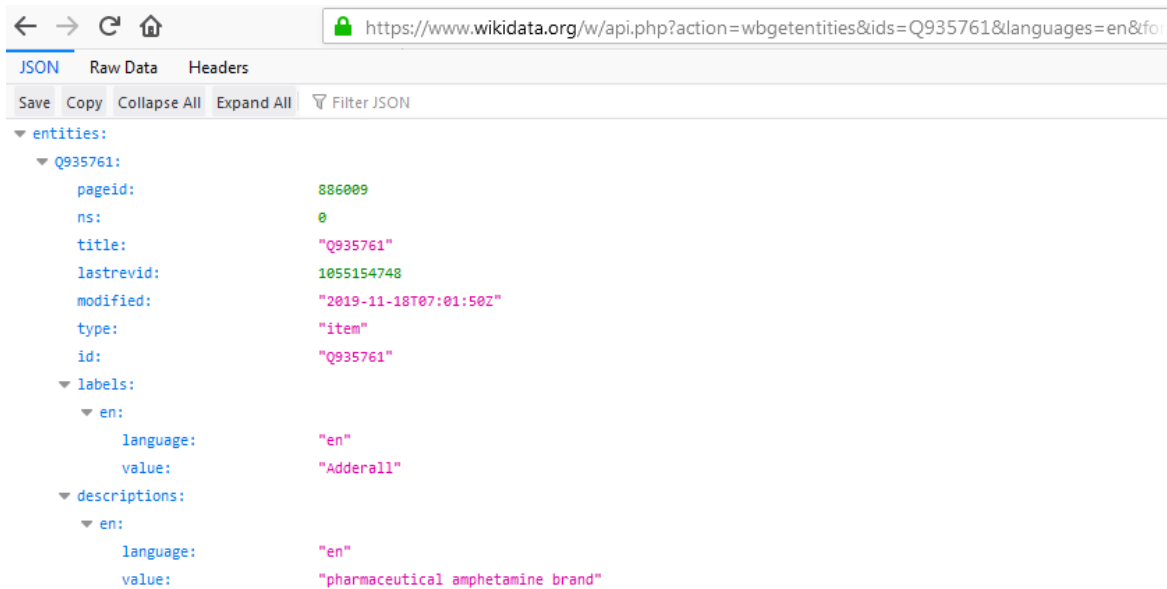
Este es el comienzo, la parte que nos interesa tiene que ver con P31 (propiedad “instance of”). Se busca...

```
"P31": [
  {
    "mainsnak": {
      "snaktype": "value",
      "property": "P31",
      "hash": "87cdd435c7bb91eadb3355615e99ee224aa44984",
      "datavalue": {
        "value": {
          "entity-type": "item",
          "numeric-id": 12140,
          "id": "Q12140"
        },
        "type": "wikibase-entityid"
      },
      "datatype": "wikibase-item"
    },
    "type": "statement",
    "id": "Q935761$BA03F2BA-8FED-460E-A63A-6BBD5936B8E8",
    "rank": "normal"
  },
  {
    "mainsnak": {
      "snaktype": "value",
      "property": "P31",
      "hash": "767ee58459dc663ef0085e81e75ea3abdbf593c5",
      "datavalue": {
        "value": {
          "entity-type": "item",
          "numeric-id": 169336,
          "id": "Q169336"
        },
        "type": "wikibase-entityid"
      },
      "datatype": "wikibase-item"
    },
    "type": "statement",
    "id": "Q935761$BA03F2BA-8FED-460E-A63A-6BBD5936B8E8",
    "rank": "normal"
  }
]
```

Y sale que tiene la id que buscamos, Q12140, y es única porque wikidata no repite ids, es mencionado en su manual.

14.- Entonces el proceso manual cambia a – se tiene un término → se convierte a una id → se mira si tiene P31 y un valor de id Q12140, el paso que falta es automatizarlo con un script.

15.- Se investiga algo más profundamente en como trabajar con JSON en python, también se fija en que la parte superior de cada solicitud tiene un format=json, se obtiene un json con



```
{
  "entities": {
    "Q935761": {
      "pageid": 886009,
      "ns": 0,
      "title": "Q935761",
      "lastrevid": 1055154748,
      "modified": "2019-11-18T07:01:50Z",
      "type": "item",
      "id": "Q935761",
      "labels": {
        "en": {
          "language": "en",
          "value": "Adderall"
        }
      },
      "descriptions": {
        "en": {
          "language": "en",
          "value": "pharmaceutical amphetamine brand"
        }
      }
    }
  }
}
```

exactamente el contenido html

16.- La forma más sencilla de hacer solicitudes a la web es usando la librería requests de python, con un ejemplo sencillo de StackOverflow se pudo adaptar a este caso:

<http://maps.googleapis.com/maps/api/directions/json?origin=Chicago,IL&destination=Los+Angeles,CA&waypoints=Joplin,MO|Oklahoma+City,OK&sensor=false>

It returns a result [in the JSON format](#).

How can I do this in Python? I want to send such a request, receive the result and parse it.

python json

share  
improve this  
question

add a comment

edited Oct 8 '15 at 15:42



[approxiblue](#)

6,066 ● 12 ● 41 ● 50

asked Jun 17 '11 at 13:17



[Arun](#)

2,059 ● 3 ● 13 ● 12

Answers

active

oldest

votes

I recommend using the awesome [requests](#) library:

```
import requests

url = 'http://maps.googleapis.com/maps/api/directions/json'

params = dict(
    origin='Chicago,IL',
    destination='Los+Angeles,CA',
    waypoints='Joplin,MO|Oklahoma+City,OK',
    sensor='false'
)

resp = requests.get(url=url, params=params)
data = resp.json() # Check the JSON Response Content documentation below
```

17.- Se altera el código añadiendo diferentes arrays para la id, las propiedades y la lista de medicinas final. Todo el código esta comentado para facilitar su comprensión

```

claves = []
#creamos un array vacio para las ids de estas clave que maneja wikidata
idsComprobar = []
# Rellenamos con el for el array con las palabras e ignorando las que tienen numeros
#for r in range (0,len(results["aggregations"]["Terminos"]["buckets"])):
for r in range (0,len(results["aggregations"]["Terminos"]["buckets"])):
    tmp = results["aggregations"]["Terminos"]["buckets"][r]["key"]
    if(any(char.isdigit() for char in tmp)!=True):
        claves.append(results["aggregations"]["Terminos"]["buckets"][r]["key"]);

#Se establece la url a la que hacer las solicitudes JSON
url = "https://www.wikidata.org/w/api.php"

for x in range (0,len(claves)):
    #Se establecen los parametros, solo va a cambiar el indice de las claves
    params = dict(
        action = "wbsearchentities",
        search = claves[x],
        language = "en",
        format = "json"
    )
    #Se obtiene la respuesta
    resp = requests.get(url=url, params=params)
    #y el objeto json
    data = resp.json()
    # Para cada posible valor, comprobar si es un medicamento
    for v in range (0,len(data["search"])):
        vid=data["search"][v]["id"]
        #Agrego las ids a un array que tendra todas las ids para comprobar
        #Solo si no estan
        if vid not in idsComprobar:
            idsComprobar.append(vid)

#se almacena en una variable la id de medication y la id de instance of,
#su obtencion ha sido mediante la API, esta documentado.
idmedication = "Q12140"
propiedad = "P31"
#se almacena en un array las ids que son medicamentos
medicamentos = []

for y in range (0,len(idsComprobar)):
    #Se establecen los parametros, solo va a cambiar el indice de las claves

```

...

Recorre los arrays y hace consultas a la api de wikidata explicada anteriormente, al final guarda los datos en el fichero. Como no se sabe si el medicamento se corresponde a la primera id (ej, adderal es Q935761) u otras (adderal también tiene la id de Q17097948), hay que añadir y comprobarlas todas.

18.- Es un proceso bastante lento y puede parecer que este bloqueado, se decidió poner a modo de barra de proceso, una información de cuanto le queda y por donde va:

```

for x in range (0,len(claves)):
    #Se establecen los parametros, solo va a cambiar el indice de las claves
    if(x%25==0):
        print("Current element: "+claves[x]+" id... "+str(x)+" of "+str(len(claves))+"\n")

```

19.- Una vez finalizado el programa, crea un fichero medicamentos.txt con la lista de medicamentos. Se confirmó que el programa funciona.

```

1 Adderall
2 antibiotic
3 cyclosporine
4 meropenem
5 lisdexamfetamine
6 alprazolam
7 bupropion
8 sertraline
9 fluoxetine
10 amoxicillin
11 clonazepam
12 escitalopram
13 Concerta
14 prednisolone
15 metronidazole
16 diphenhydramine
17 doxycycline
18 duloxetine
19 dextroamphetamine
20 gabapentin
21 trazodone
22 atomoxetine
23 hydroxyzine
24 paroxetine
25 clindamycin
26 polyethylene glycol
27 aspirin
28 buspirone
29 levofloxacin
30 ibuprofen
31 cyclobenzaprine
32 lisinopril
33 ciprofloxacin
34 diazepam

```

20.- A partir de aquí se puede obtener más o menos exhaustividad que se desee con cualquier número de consultas, hay que tener en cuenta que a mayor exhaustividad, mayor duración ya que tiene que ver y comprobar más términos.

```

for r in range (0,len(results["aggregations"]["Terminos"]["buckets"])):
    tmp = results["aggregations"]["Terminos"]["buckets"][r]["key"]
    if(any(char.isdigit() for char in tmp)!=True):
        claves.append(results["aggregations"]["Terminos"]["buckets"][r]["key"]);

for r2 in range (0,len(results2["aggregations"]["Terminos"]["buckets"])):
    tmp = results2["aggregations"]["Terminos"]["buckets"][r2]["key"]
    if(any(char.isdigit() for char in tmp)!=True and results2["aggregations"]["Terminos"]["buckets"][r2]["key"] not in claves):
        claves.append(results2["aggregations"]["Terminos"]["buckets"][r2]["key"]);

```

## Ejercicio 4

En este ejercicio se nos pide encontrar factores comórbidos relativos a la ideación suicida y a las conductas autolesivas.

Para ello lo primero que vamos a hacer es encontrar los términos más importantes dentro del conjunto de post de Reddit proporcionado mediante una consulta:

```
es = Elasticsearch()

#numero de resultados maximo que dara la consulta
numero_salidas = 500

results = es.search(
    index="reddit-mentalhealth",
    body = {
        "size": 0,
        "query": {
            "query_string": {
                "default_field": "selftext",
                "query": query
            }
        },
        "aggs": {
            "Title": {
                "significant_terms": {
                    "field": "title",
                    "size": numero_salidas,
                    "gnd": {}
                }
            },
            "Text": {
                "significant_terms": {
                    "field": "selftext",
                    "size": numero_salidas,
                    "gnd": {}
                }
            },
            "Subreddit": {
                "significant_terms": {
                    "field": "subreddit",
                    "size": numero_salidas,
                    "gnd": {}
                }
            }
        }
    }
)

words = []
for j in ["Subreddit", "Text", "Title"]:
    for i in results["aggregations"][j]["buckets"]:
        if i["key"] not in words and i["key"] not in querywords:
```

```
words.append(i["key"])  
print("Obtenidas palabras relacionadas con "+query+": "+str(len(words)))  
return words
```



En esta consulta buscamos los términos dados en el parámetro query ("suicide suicidal \"kill myself\" \"killing myself\" \"end my life\"" en el caso de suicidio y "\"self harm\"" en el caso de conductas autolesivas) y devolvemos los resultados en una lista donde quitamos los términos usados, ya que por definición los factores comórbidos no pueden ser la propia enfermedad.

A continuación necesitamos validar los resultados con un conocimiento experto, para ello utilizaremos la herramienta Publish or Perish ofrecida en <https://harzing.com/resources/publish-or-perish>.

Una vez obtenida la herramienta hacemos las consultas de “suicide comorbidity” y “self harm comorbidity” en Google Scholar obteniendo así dos archivos Json que usaremos más adelante en el script

Harzing's Publish or Perish (Windows GUI Edition) 7.15.2643.7260

File Edit Search View Help

My searches Trash

Search terms: suicide comorbidity Source: Google Sch... Papers: 994 Cites: 184167 Cites/ye...: 5580.82 h: 219 g: 381 hl: 123 hl: 3.73 acc10: 423 Search date: 13/12/2019 Cache date: 13/12/2019 Last...: 0

Google Scholar search

How to search with Google Scholar

Authors: Years: 0 - 0 Search

Publication name: ISSN: Search Direct

Title Clear All

Keywords: suicide comorbidity Revert

New

Results

Publication years: 1986-2019

Citation years: 33 (1986-2019)

Papers: 994

Citations: 184167

Cites/year: 5580.82

Cites/paper: 185.28

Authors/paper: 3.43

h-index: 219

g-index: 381

hl: 123

hl: 3.73

Papers with ACC >= 1,2,5,10,20: 957,926,732,423,183

Copy Results

Save Results

Cites	Per year	Rank	Authors	Title	Year	Publication	Publisher	Type
974	37.46	1	MM Henriksson, H.L.	Mental disorders and comorbidity...	1993	American journal of ...	researchgate.net	PDF
2704	135.20	2	RC Kessler, G Borg...	Prevalence of and risk factors for L...	1999	Archives of general ...	jamanetwork.com	
341	11.76	3	J Johnson, MM Wei...	Panic disorder, comorbidity, and s...	1990	Archives of General ...	jamanetwork.com	
309	13.43	4	K Suominen, M He...	Mental disorders and comorbidity...	1996	Acta Psychiatrica ...	Wiley Online Library	
609	46.85	5	MK Nock, RC Kessler	Prevalence of and risk factors for s...	2006	Journal of abnormal psych...	psycnet.apa.org	
213	10.14	6	U Wunderlich, T Br...	Comorbidity patterns in adolesce...	1998	European archives of psyc...	Springer	
305	21.79	7	J Sareen, T Houalah...	Anxiety disorders associated with ...	2005	The Journal of nervous ...	journals.lww.com	
292	9.42	8	M Shafii, J Steltz-L...	Comorbidity of mental disorders L...	1988	Journal of Affective ...	Elsevier	
179	14.92	9	A McGirr, J Paris, A ...	Risk factors for suicide completio...	2007	The Journal of clinical ...	psycnet.apa.org	
255	15.94	10	K Hawton, K Houst...	Comorbidity of axis I and axis II di...	2003	American Journal of ...	Am Psychiatric Assoc	
149	12.42	11	SJ Wang, KD Juang...	Psychiatric comorbidity and suicid...	2007	Neurology	AAN Enterprises	
182	7.28	12	T Bronisch, HU Wit...	Suicidal ideation and suicide atte...	1994	European Archives of Psyc...	Springer	
236	18.15	13	G Borges, J Angst, ...	A risk index for 12-month suicide ...	2006	Psychological ...	cambridge.org	
290	26.36	14	G Borges, J Angst, ...	Risk factors for the incidence and ...	2008	Journal of affective ...	Elsevier	

Harzing's Publish or Perish (Windows GUI Edition) 7.15.2643.7260

File Edit Search View Help

My searches Trash

Search terms: self harm comorbidity Source: Google Sch... Papers: 992 Cites: 102277 Cites/ye...: 3409.23 h: 157 g: 268 hl: 90 hl: 3.00 acc10: 315 Search date: 13/12/2019 Cache date: 13/12/2019 Last...: 0

Google Scholar search

How to search with Google Scholar

Authors: Years: 0 - 0 Search

Publication name: ISSN: Search Direct

Title Clear All

Keywords: self harm comorbidity Revert

New

Results

Publication years: 1989-2019

Citation years: 30 (1989-2019)

Papers: 992

Citations: 102277

Cites/year: 3409.23

Cites/paper: 103.10

Authors/paper: 3.43

h-index: 157

g-index: 268

hl: 90

hl: 3.00

Papers with ACC >= 1,2,5,10,20: 922,841,592,315,131

Copy Results

Save Results

Cites	Per year	Rank	Authors	Title	Year	Publication	Publisher	Type
192	8.35	1	SL Welch, CG Fairb...	Impulsivity or comorbidity in buli...	1996	The British Journal of Psyc...	cambridge.org	
47	6.71	2	S Moor, M Crowe, ...	Effects of comorbidity and early a...	2012	Journal of affective disord...	Elsevier	
85	7.73	3	Y Kaminer, OG Buk...	Adolescent substance abuse: Psyc...	2008	Journal of abnormal psych...	books.google.com	BOOK
609	46.85	4	MK Nock, RC Kessler	Prevalence of and risk factors for s...	2006	Journal of abnormal psych...	psycnet.apa.org	
205	68.33	5	A Keski-Rahkonen, ...	Epidemiology of eating disorders ...	2016	Current opinion in psychiat...	journals.lww.com	
56	14.00	6	BJ Turner, KL Dixon...	Non-suicidal self-injury with and ...	2015	Psychiatry ...	Elsevier	
106	8.15	7	KD Wu, LA Clark, D...	Relations between obsessive-com...	2006	Journal of Anxiety Disorders	Elsevier	
65	5.42	8	S Fischer, D Grange	Comorbidity and high-risk behavi...	2007	International Journal of Ea...	Wiley Online Library	
134	14.89	9	F Andersohn, R Sch...	Use of antiepileptic drugs in epile...	2010	Neurology	AAN Enterprises	
72	7.20	10	EJ Kilbane, NS Gok...	A review of panic and suicide in bi...	2009	Journal of affective ...	Elsevier	
22	3.67	11	S Apfelbaum, P Re...	Comorbidity between bipolar dis...	2013	Actas espanolas de ...	europemc.org	
526	29.22	12	C Haw, K Hawton, ...	Psychiatric and personality disord...	2001	The British Journal of ...	cambridge.org	
84	7.64	13	A Favaro, P Santon...	Self-injurious behavior and attem...	2008	Journal of Affective ...	Elsevier	
58	4.83	14	B McCormick, N BL...	Relationship of sex to symptom se...	2007	Comprehensive ...	Elsevier	

Obtenemos los títulos de los artículos obtenidos en los Json mediante el script:

```
words = []
#Es necesario especificar el encoding para que no de error
with open(name, encoding='utf-8-sig') as json_file:
    data = json.load(json_file)
    for word in data:
        words.append(word['title'])
print("Obtenidas palabras del json "+name+": "+str(len(words)))
return words
```

Y una vez hecho esto solo hay que comparar las dos listas de palabras para obtener así los términos que hacen referencia a las comorbilidades

```
wordsElasticSearch = readElasticsearch(query)
wordsJson = readJson(json)
finalWords = []

for we in wordsElasticSearch:
    for wj in wordsJson:
        #separamos las palabras de los titulos y comprobamos que no se
        #hayan añadido ya
        if we in wj.split() and we not in finalWords:
            finalWords.append(we)
return finalWords
```

Nota:

Para realizar la consulta con elasticsearch correctamente es necesario configurarlo anteriormente, para ello se ha optado en meter el siguiente código en el script, aunque también se podría hacer desde cerebro:

```
es = Elasticsearch()
mapping = {
    "properties": {
        "author": {
            "type": "text",
            "fielddata": "true"
        },
        "selftext": {
            "type": "text",
            "fielddata": "true"
        },
        "title": {
            "type": "text",
            "fielddata": "true"
        },
        "subreddit": {
            "type": "text",
            "fielddata": "true"
        }
    }
}
```

```
    }  
  }  
}  
  
response = es.indices.put_mapping(  
    index="reddit-mentalhealth",  
    body=mapping,  
    ignore=400 # ignore 400 already exists code  
)
```