Repositorios de Información

Recuperación de información /
Una introducción “gentil” a Elasticsearch (2ª parte)

Tarea: Paginación de resultados (solución)

Vamos a descargar, ejecutar y analizar el script **`scan.py`**, así como probarlo con diferentes tipos de consultas.

Por defecto `helpers.scan` no proporciona resultados en ningún orden en particular (p. ej., ordenados por relevancia).

ElasticDump

- *ElasticDump* es una herramienta para importar y exportar data de repositorios *Elasticsearch* usando archivos *ndjson*.
- Por ejemplo, podríamos hacer lo siguiente para generar un volcado con todos los tuits escritos en español y que mencionaran a Michael Jackson:

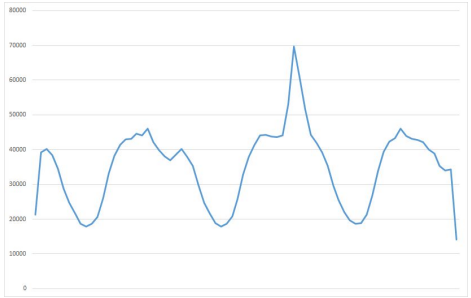
```
elasticdump
--input=https://lectura:abretesesamo@127.0.0.1:9200/tweets-20090624-20090626-en_es-10percent
--output=michaeljackson_en.json
--searchBody "{\"query\":{\"query_string\":{\"query\": \"text:\\\"michael jackson\\\" AND lang:es\"}}}"
```

- **¡Atención!** Para que esto funcione necesitas añadir el rol `monitoring_user` al usuario `lectura` utilizando `bin/elasticsearch-users` **y, además,** establecer esta variable `NODE_TLS_REJECT_UNAUTHORIZED=0` antes de ejecutar *ElasticDump*.

Agregaciones

- Las agregaciones en ES nos permiten obtener valores numéricos, estadísticas y otro tipo de datos que pueden ser útiles para propósitos estadísticos o para realizar minería de datos.
- Se agrupan en tres grandes categorías: metric, bucket y pipeline.
- Veamos algunos ejemplos útiles…

Agregaciones



El extraño pico al final del tercer día de datos coincide con la muerte de Michael Jackson.

El parámetro `fixed_interval` admite las siguientes unidades: `ms`, `s`, `m`, `h` y `d`.

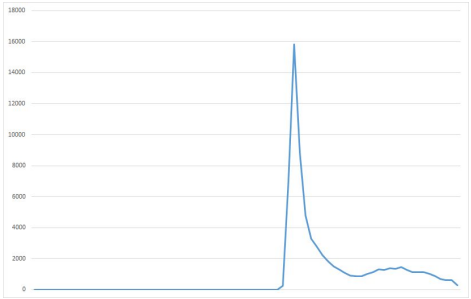
- Por ejemplo, puesto que los tuits tienen marcas de tiempo podemos explotar esa información para generar series temporales y buscar “picos”.
- Para hacer eso (si tuvimos la precaución de usar un tipo “date” en ES—sí, la tuvimos) es tan sencillo como hacer lo siguiente:

```
tweets-20090624-20090626-en_es-10percent/_search
{
    "size":0,
    "aggs":{
        "Tweets per hour":{
            "date_histogram": {
                "field":"created_at",
                "fixed_interval": "1h"
            }
        }
    }
}
```

Agregaciones

- En la anterior consulta ES no se especificó ninguna consulta en el sentido de RI y, en consecuencia, la agregación se calculó sobre toda la colección de documentos.
- Veamos pues si el pico tiene algo que ver con las noticias sobre la muerte de Michael Jackson...

```
tweets-20090624-20090626-en_es-10percent/_search
{
    "size":0,
    "query": {
        "simple_query_string": {
            "query":"\"michael jackson\""
        }
    },
    "aggs":{
        "Tweets per hour":{
            "date_histogram": {
                "field":"created_at",
                "fixed_interval": "1h"
            }
        }
    }
}
```



Parece que sí tiene que ver con las noticias sobre Michael Jackson...

Agregaciones

- Otra cuestión que podríamos tratar de resolver es cuántos usuarios diferentes hay en esta colección.
- Para ello solo habría que hacer lo siguiente:

```
tweets-20090624-20090626-en_es-10percent/_search
{
    "size":0,
    "aggs":{
        "Different users":{
            "cardinality": {
                "field":"user_id_str"
            }
        }
    }
}
```

Al hacer esto nos aparece un mensaje de error un tanto críptico:

```
Text fields are not optimised for operations that require per-document field data like aggregations and sorting, so these operations are disabled by default. Please use a keyword field instead. Alternatively, set fielddata=true on [user_id_str] in order to load field data by uninverting the inverted index. Note that this can use significant memory.
```

Básicamente, ES nos está diciendo que las agregaciones y los campos de texto no se llevan muy bien…

En este caso no es un gran problema porque, en realidad, el campo `user_id_str` es solo un identificador aunque le hayamos asignado el tipo texto y, en consecuencia, hay una “solución”...

Agregaciones

- Otra cuestión que podríamos tratar de resolver es cuántos usuarios diferentes hay en esta colección **(segundo intento).**
- Para ello solo habría que hacer lo siguiente:

```
tweets-20090624-20090626-en_es-10percent/_search
{
    "size":0,
    "aggs":{
        "Different users":{
            "cardinality": {
                "field":"user_id_str.keyword"
            }
        }
    }
}
```

Así pues, en nuestra colección hay 976.579 usuarios y, puesto que han publicado 2,7 millones de tuits, parece que cada usuario publica un número bastante reducido de tuits (menos de 3 tuits por usuario en promedio).

¿O no...?

```
"aggregations": {
  "Tweets per user": {
    "doc_count_error_upper_bound": 0,
    "sum_other_doc_count": 2650931,
    "buckets": [
      {
        "key": "19148579",
        "doc_count": 819
      },
      {
        "key": "15160529",
        "doc_count": 797
      },
      {
        "key": "20112904",
        "doc_count": 704
      },
      {
        "key": "31710845",
        "doc_count": 672
      },
      {
        "key": "19083154",
        "doc_count": 581
      },
      {
        "key": "16711892",
        "doc_count": 449
      },
      {
        "key": "27383238",
        "doc_count": 445
      },
      {
        "key": "44678658",
        "doc_count": 416
      },
```


En Twitter (como en la mayoría de plataformas online) la mayor parte del contenido es producido por un número reducido de usuarios.

Agregaciones

- Para saber cuántos tuits ha publicado cada usuario solo necesitamos hacer lo siguiente—pero ten en cuenta que esto es un poco "ñapa":

```
tweets-20090624-20090626-en_es-10percent/_search
{
    "size":0,
    "aggs":{
        "Tweets per user":{
            "terms": {
                "field":"user_id_str.keyword"
            }
        }
    }
}
```

Agregaciones

- ¿Qué sucede si calculamos una agregación `terms` sobre el campo `text`?

```
tweets-20090624-20090626-en_es-10percent/_search
{
    "size":0,
    "aggs":{
        "Tweets per tweet?":{
            "terms": {
                "field":"text.keyword"
            }
        }
    }
}
```

Agregaciones

```
"aggregations": {
    "Tweets per tweet?": {
      "doc_count_error_upper_bound": 0,
      "sum_other_doc_count": 2649386,
      "buckets": [
        {
          "key": "Get 400 followers a day using http://www.tweeterfollow.com",
          "doc_count": 1078
        },
        {
          "key": "Show support for democracy in Iran add green overlay to your Twitter avatar with
                 1-click - http://helpiranelection.com/",
          "doc_count": 839
        },
        {
          "key": "Show support for democracy in Iran add green ribbon to your Twitter avatar with
                 1-click - http://helpiranelection.com/",
          "doc_count": 771
        },
        {
          "key": "RIP Michael Jackson",
          "doc_count": 701
        },
        {
          "key": "My Daily Twittascope - http://twittascope.com/my2.php?sign=Leo",
          "doc_count": 640
        }
```

Podemos observar dos cosas:

Por un lado, hay tuits que aparecen repetidos literalmente cientos de veces. Podría ser spam o quizás *copypasta*.

Por otro lado, vemos que hay campos de texto donde utilizar el subcampo `keyword` para tratarlos como meras secuencias de caracteres es totalmente inútil. Eso puede tener sentido para nombres de usuario, direcciones de correo y similares pero no para fragmentos largos de texto libre.

En esos casos necesitamos analizar el texto para poder trabajar con los términos que lo constituyen.

¿Qué era lo que decía aquel mensaje de error tan extraño?

Agregaciones

- Si usar el subcampo `keyword` no es lo que necesitamos para trabajar con el texto de los tuits está claro que tenemos que activar **`fielddata`**.
- Para ello hacemos lo siguiente:

```
PUT tweets-20090624-20090626-en_es-10percent/_mapping
{
  "properties": {
    "text": {
      "type": "text",
      "fielddata": true
    }
  }
}
```

El mensaje de error era el siguiente:

```
Text fields are not optimised for operations that require per-document field data like aggregations and sorting, so these operations are disabled by default. Please use a keyword field instead. Alternatively, set fielddata=true on [user_id_str] in order to load field data by uninverting the inverted index. Note that this can use significant memory.
```

Agregaciones

- Si ahora enviamos esta consulta:

```
tweets-20090624-20090626-en_es-10percent/_search
{
  "size": 0,
  "aggs": {
    "Most frequent terms": {
      "terms": {
        "field": "text"
      }
    }
  }
}
```

Agregaciones

```
"aggregations": {
  "Most frequent terms": {
    "doc_count_error_upper_bound": 0,
    "sum_other_doc_count": 29567338,
    "buckets": [
      {
        "key": "the",
        "doc_count": 728107
      },
      {
        "key": "http",
        "doc_count": 697483
      },
      {
        "key": "to",
        "doc_count": 665045
      },
      {
        "key": "i",
        "doc_count": 540763
      },
      {
        "key": "a",
        "doc_count": 507960
      }
```

No es nada sorprendente que obtengamos una lista de palabras vacías en inglés (después de todo ese idioma domina la colección), como tampoco es sorprendente que aparezca una palabra tan habitual en Twitter como `http`.

¿Qué pasa si buscamos los términos más frecuentes en aquellos documentos escritos en español?

Agregaciones

- Términos más frecuentes para los tuits con `lang:es`

```
tweets-20090624-20090626-en_es-10percent/_search
{
  "size": 0,
  "query": {
      "query_string":{
          "query":"lang:es"
      }
  },
  "aggs": {
    "Most frequent terms": {
      "terms": {
        "field": "text"
      }
    }
  }
}
```

Agregaciones


```
"aggregations": {
  "Most frequent terms": {
    "doc_count_error_upper_bound": 0,
    "sum_other_doc_count": 935051,
    "buckets": [
      {
        "key": "de",
        "doc_count": 32912
      },
      {
        "key": "http",
        "doc_count": 22505
      },
      {
        "key": "que",
        "doc_count": 22160
      },
      {
        "key": "la",
        "doc_count": 21800
      },
      {
        "key": "a",
        "doc_count": 20719
      },
```

Agregaciones

- ¿Y si buscáramos los términos más frecuentes en los tuits escritos en inglés y que mencionan a Michael Jackson?

```
tweets-20090624-20090626-en_es-10percent/_search
{
  "size": 0,
  "query": {
      "query_string":{
          "query":"text:\"michael jackson\" AND lang:en"
      }
  },
  "aggs": {
    "Most frequent terms": {
      "terms": {
        "field": "text"
      }
    }
  }
}
```

Agregaciones

```
"aggregations": {
  "Most frequent terms": {
    "doc_count_error_upper_bound": 0,
    "sum_other_doc_count": 669907,
    "buckets": [
      {
        "key": "jackson",
        "doc_count": 68979
      },
      {
        "key": "michael",
        "doc_count": 68979
      },
      {
        "key": "the",
        "doc_count": 19265
      },
      {
        "key": "http",
        "doc_count": 15912
      },
      {
        "key": "to",
        "doc_count": 14896
      },
```

De nuevo el resultado no debería sorprendernos…

Por un lado, como era de esperar, encontramos `michael` y `jackson` y, por otro, hay palabras muy frecuentes en inglés.

Lo que sucede es que, en realidad, no queremos encontrar los términos más frecuentes en esos tuits. Lo que queremos es encontrar términos sobrerrepresentados en los tuits que mencionan a Michael Jackson en comparación con el resto de tuits.

Elasticsearch tiene la agregación perfecta para hacer eso.

Agregaciones

- Términos significativos para un subconjunto de documentos definidos por una consulta.

```
tweets-20090624-20090626-en_es-10percent/_search
{
  "size": 0,
  "query": {
      "query_string":{
          "query":"text:\"michael jackson\" AND lang:en"
      }
  },
  "aggs": {
    "Most significant terms": {
      "significant_terms": {
        "field": "text"
      }
    }
  }
}
```

Agregaciones


```
"aggregations": {
  "Most significant terms": {
    "doc_count": 68979,
    "bg_count": 2656499,
    "buckets": [
      {
        "key": "jackson",
        "doc_count": 68979,
        "score": "29.351663543713727",
        "bg_count": 87524
      },
      {
        "key": "michael",
        "doc_count": 68979,
        "score": "27.755902187679283",
        "bg_count": 92381
      },
      {
        "key": "rip",
        "doc_count": 8951,
        "score": "1.6374636134887495",
        "bg_count": 25312
      },
```


Como era de esperar, aparecen tanto `michael` como `jackson` pero también encontramos otros términos como `rip`, `died`, `dead`, `r.i.p.`, `pop` y `cardiac`, además de `farrah` y `fawcett` porque ese mismo día también falleció la actriz Farrah Fawcett (famosa por la serie “Los Ángeles de Charlie”).

Agregaciones

- La agregación de términos significativos utiliza una serie de medidas estadísticas con diferentes propiedades que dan lugar a listas de términos ligeramente diferentes.
- Las principales son: **`jlh`** (predeterminada), `mutual_information`, `chi_square` y **`gnd`**—que es bastante insensible a palabras vacías.
- Además, el número de términos que retorna la agregación se puede especificar con el parámetro `size`.

Agregaciones

- Por otro lado, aunque en los ejemplos anteriores tanto la consulta como la agregación operaban sobre el campo `text` podemos **ejecutar la consulta sobre un campo** y solicitar los **términos significativos sobre otro diferente.**
- Por ejemplo, podríamos buscar usuarios que fueron “significativos” en relación con la historia de Michael Jackson (es decir, tuitearon mucho más sobre ese tema que sobre otros temas).

```
tweets-20090624-20090626-en_es-10percent/_search
{
  "size": 0,
  "query": {
      "query_string":{
          "query":"text:\"michael jackson\" AND lang:en"
      }
  },
  "aggs": {
    "Most significant users": {
      "significant_terms": {
        "field": "user_id_str.keyword",
        "size":100
      }
    }
  }
}
```

El dataset no incluye nombres de usuario, solo sus identificadores; sin embargo, es sencillo conseguir el nombre de usuario para cada id:

```
50804931 @jacksonmichael5
42729435 @ShawnaBrickley
41650343 @sheep_3
50992894 @inmemoryofmj09
41653847 @50_FAGGOTS
41641292 @jcfosters
50852531 @Jackson_Dead
46482766 @carlocivicsi
50987016 @_R_I_P_MJ
49768232 @nolly_pavey
```

...

Agregaciones

- Por último, podemos calcular **varias agregaciones** en **una única consulta**. Las agregaciones pueden estar al mismo nivel—*siblings*—o una agregación puede trabajar sobre la salida de la anterior—*parent*. En este ejemplo vamos a **calcular 10 "trending topics" por hora**.

```
tweets-20090624-20090626-en_es-10percent/_search
{
  "size": 0,
  "query": {
    "query_string": {
      "query": "lang:en"
    }
  },
  "aggs": {
    "Trending topics per hour": {
      "date_histogram": {
        "field": "created_at",
        "fixed_interval": "1h"
      },
      "aggs": {
        "Trending topics": {
          "significant_terms": {
            "field": "text",
            "size": 10,
            "gnd": {}
          }
        }
      }
    }
  }
}
```

Agregaciones


```
"key_as_string": "Thu Jun 25 17:00:00 +0000 2009",
"key": 1245949200000,
"doc_count": 42421,
"Trending topics": {
  "doc_count": 42421,
  "bg_count": 2656499,
  "buckets": [
    {
      "key": "farrah",
      "doc_count": 1764,
      "score": 0.5302056769448703,
      "bg_count": 17686
    },
    {
      "key": "62",
      "doc_count": 310,
      "score": 0.5295827710527616,
      "bg_count": 1158
    },
    {
      "key": "fawcett",
      "doc_count": 1248,
      "score": 0.5286044915961765,
      "bg_count": 10525
    },
```

```
"key_as_string": "Thu Jun 25 22:00:00 +0000 2009",
"key": 1245967200000,
"doc_count": 66957,
"Trending topics": {
  "doc_count": 66957,
  "bg_count": 2656499,
  "buckets": [
    {
      "key": "jackson",
      "doc_count": 18092,
      "score": 0.6516170528577754,
      "bg_count": 87524
    },
    {
      "key": "michael",
      "doc_count": 17946,
      "score": 0.6407134752498718,
      "bg_count": 92381
    },
    {
      "key": "dead",
      "doc_count": 8155,
      "score": 0.640123149934731,
      "bg_count": 23691
    },
```

Indexado con eliminación de palabras vacías, *stemming* y n-gramas de términos

- Durante las clases de teoría hablamos de la eliminación de palabras vacías (que puede ser interesante), de *stemming* (que puede ser útil para aglutinar términos asociados semánticamente a través de su raíz léxica) y de los problemas que plantean las entidades multipalabra (p. ej., *"café au lait"*, *"Unión Europea"* o *"Universidad de Oviedo"*) cuando el texto se "tokeniza" en "términos".
- Hay varias formas de abordar esas cuestiones en *Elasticsearch*. Algunas son bastante rápidas (pero no dan los resultados que nos gustarían) y otras son algo más elaboradas.
- A continuación se ofrecen algunos ejemplos ilustrativos. Lo ideal sería ejecutar los scripts para producir los distintos índices y luego probar las mismas consultas en cada uno de ellos.

¡Atención! En *Elasticsearch* se usa el término *shingle* para referirse a n-gramas de términos y *ngram* para referirse a n-gramas de caracteres extraídos de los términos.

En consecuencia, *"Unión Europea"* es un "shingle" de dos términos, mientras que *"un"* y *"ón"* son n-gramas de dos caracteres o bigramas.

Indexado con eliminación de palabras vacías, *stemming* y n-gramas de términos

- Tenemos los siguientes scripts:
  - **`bulk-indexer05.py`** ilustra la forma más inmediata de hacerlo casi todo (eliminación de palabras vacías y *stemming*, no genera *shingles*). Para ello usa un *parser* predefinido: `english`. Observa cómo el tamaño del índice se reduce considerablemente aun cuando contiene casi el mismo número de documentos que el primer índice que creamos. Lamentablemente, no funciona como nos gustaría (preserva muchas palabras vacías como *"I"*, *"my"*, *"just"*, *"so"*, ...) Este script genera el índice `tweets-20090624-20090626-en_es-10percent-v05`.
  - **`bulk-indexer2.py`** muestra cómo proporcionar una lista de palabras vacías; es más verboso pero nos da la ventaja de la personalización. Se ha escogido esta solución puesto que está claro que *Elasticsearch* no usa la lista de palabras vacías sugerida por Porter. **¡Atención!** Sería mucho mejor proporcionar la lista de palabras vacías mediante una ruta a un fichero concreto en el servidor pero de esta manera el script es autocontenido. Este código genera el índice `tweets-20090624-20090626-en_es-10percent-v2`.
  - **`bulk-indexer3.py`** elimina palabras vacías como el script anterior y también aplica un algoritmo de *stemming*, en concreto Porter2 (se considera que esta versión es mejor que el *stemmer* original de Porter y no es tan agresiva como otros algoritmos). Este script genera el índice `tweets-20090624-20090626-en_es-10percent-v3`.
  - **`bulk-indexer4.py`** hace lo mismo que el script anterior y, además, produce *shingles* de 2 y 3 términos junto con unigramas (no me preguntéis por qué los *shingles* de un solo término son unigramas).

Indexado con eliminación de palabras vacías, *stemming* y n-gramas de términos

- Para cada uno de los índices vamos a probar tres consultas diferentes:

```
{
  "size": 0,
  "query": {
    "match": {
      "text": "the"
    }
  },
  "aggs": {
    "Most significant terms": {
      "significant_terms": {
        "field": "text",
        "size": 25
      }
    }
  }
}
```

En este caso, ninguno de los índices retorna términos asociados ya que el término *“the”* simplemente no existe.

Indexado con eliminación de palabras vacías, *stemming* y n-gramas de términos

- Para cada uno de los índices vamos a probar tres consultas diferentes:

```
{
  "size": 0,
  "query": {
    "match": {
      "text": "my"
    }
  },
  "aggs": {
    "Most significant terms": {
      "significant_terms": {
        "field": "text",
        "size": 25
      }
    }
  }
}
```

Los índices v2, v3 y v4 no devuelven ningún término significativo ya que ninguno contiene el término *“my”*.

El índice v05, que utiliza una lista de palabras vacías mucho más limitada, sí proporciona resultados: *“i”*, *“twittascop”*, *“daili”*, or *“favorit”* (observa que esos términos han sido “estematizados”).

Indexado con eliminación de palabras vacías, *stemming* y n-gramas de términos

- Para cada uno de los índices vamos a probar tres consultas diferentes:

```
{
  "size": 0,
  "query": {
    "match": {
      "text": "iran"
    }
  },
  "aggs": {
    "Most significant terms": {
      "significant_terms": {
        "field": "text",
        "size": 25
      }
    }
  }
}
```

Indexado con eliminación de palabras vacías, *stemming* y n-gramas de términos

```
{
  "size": 0,
  "query": {
    "match": {
      "text": "iran"
    }
  },
  "aggs": {
    "Most significant terms": {
      "significant_terms": {
        "field": "text",
        "size": 25
      }
    }
  }
}
```

Observa cómo se aplicó *stemming* en los índices v05, v3 y v4.

Y observa también cómo los *shingles* del índice v4 muestran la posición de las palabras vacías con guiones bajos _.


```
{
  "key": "iran",
  "doc_count": 17628,
  "score": "144.05366462446108",
  "bg_count": 17628
},
{
  "key": "iranelect",
  "doc_count": 5613,
  "score": "18.25237400003504",
  "bg_count": 13960
},
{
  "key": "helpiranelection.com",
  "doc_count": 1722,
  "score": "13.711097603387076",
  "bg_count": 1767
},
{
  "key": "democraci",
  "doc_count": 1922,
  "score": "13.258215811980474",
  "bg_count": 2274
},
{
  "key": "avatar",
  "doc_count": 1796,
  "score": "8.239501365237315",
  "bg_count": 3182
},
{
  "key": "neda",
  "doc_count": 2118,
  "score": "8.01760371090251",
  "bg_count": 4536
},
```

```
{
  "key": "iran",
  "doc_count": 18037,
  "score": "140.7644841159838",
  "bg_count": 18037
},
{
  "key": "iranelection",
  "doc_count": 5804,
  "score": "18.834819030240595",
  "bg_count": 13821
},
{
  "key": "helpiranelection",
  "doc_count": 1725,
  "score": "13.10265176784291",
  "bg_count": 1772
},
{
  "key": "democracy",
  "doc_count": 1928,
  "score": "12.65665622878028",
  "bg_count": 2289
},
{
  "key": "neda",
  "doc_count": 2202,
  "score": "8.264778571838232",
  "bg_count": 4544
},
{
  "key": "avatar",
  "doc_count": 1757,
  "score": "8.11068139153152",
  "bg_count": 2956
},
```

```
{
  "key": "iran",
  "doc_count": 17586,
  "score": "144.40009098146254",
  "bg_count": 17586
},
{
  "key": "iranelect",
  "doc_count": 5606,
  "score": "18.294309445550322",
  "bg_count": 13960
},
{
  "key": "helpiranelection.com",
  "doc_count": 1722,
  "score": "13.776901089578802",
  "bg_count": 1767
},
{
  "key": "democraci",
  "doc_count": 1922,
  "score": "13.321880673980118",
  "bg_count": 2274
},
{
  "key": "avatar",
  "doc_count": 1796,
  "score": "8.27914846610782",
  "bg_count": 3182
},
{
  "key": "neda",
  "doc_count": 2109,
  "score": "7.99814085600801",
  "bg_count": 4530
},
```

```
{
  "key": "iran",
  "doc_count": 17586,
  "score": "144.40009098146254",
  "bg_count": 17586
},
{
  "key": "_ iran",
  "doc_count": 9729,
  "score": "79.88561839864944",
  "bg_count": 9729
},
{
  "key": "iran _",
  "doc_count": 3662,
  "score": "30.068982894013182",
  "bg_count": 3662
},
{
  "key": "_ iran _",
  "doc_count": 2450,
  "score": "20.117151308119144",
  "bg_count": 2450
},
{
  "key": "iranelect",
  "doc_count": 5606,
  "score": "18.294309445550322",
  "bg_count": 13960
},
{
  "key": "democraci _ iran",
  "doc_count": 1782,
  "score": "14.63214842084421",
  "bg_count": 1782
},
```

Indexado con eliminación de palabras vacías, *stemming* y n-gramas de términos

- ¿Qué opción es mejor? Depende... No hay una fórmula milagrosa.
- A veces resulta útil eliminar las palabras vacías y otras es una mala idea (p.ej., una gran cantidad de pronombres en primera persona puede ser señal de que la persona está verbalizando problemas personales).
- A veces puede ser muy interesante utilizar *shingles* y otras en las que el excesivo tamaño del índice no merezca la pena...
- La única forma de saberlo es probar y ver qué configuración es la mejor para el problema concreto que estás tratando de resolver.

Repositorios de Información

Recuperación de información /
Una introducción “gentil” a Elasticsearch (Fin de la 2ª parte)

Entregables

- Los **ejercicios son individuales** y no se pueden hacer en equipos ni en parejas. Por favor, **no compartas ni enseñes código** cuando ayudes a otras personas.
- Todos los ejercicios **deben resolverse usando *Python*.**
- **De manera opcional**, puedes trabajar con la **colección completa** en lugar de usar la muestra más pequeña que se utilizó en los laboratorios.
- Por supuesto puedes pedir ayuda a Dani (`dani@uniovi.es`) tanto como necesites pero siempre debes proporcionar la consulta o el código que no consigues que funcione.
- **Fecha límite:** 23 de diciembre de 2022 antes de las 23:55 h.
- Tienes que entregar:
  - Todo el código fuente (incluyendo el código para crear los índices si hubieras modificado los scripts que se proporcionaron como ejemplos).
  - Un documento que explique cómo has resuelto cada ejercicio; por favor, ve más allá de una mera descripción del código.
  - La listas de "trending topics" con las entidades asociadas (y opcionalmente sinónimos y tipos) en el ejercicio 1.
  - El archivo TSV o *ndjson* generado en el ejercicio 2.

El entregable será un único archivo ZIP con el código, la documentación y los archivos de salida de datos (**¡no incluyas índices!**)

Si algún archivo de datos fuera excesivamente grande, se subirá a *OneDrive* y se incluirá la URL en la documentación.

Entregables: Ejercicio 1 (3 puntos)

- Generar un índice de tuits escritos en Español usando *shingles* de dos y tres términos.
- Escribe un script basado en el ejemplo de agregación que generaba “trending topics” y que pueda generar listas de 50 “trending topics” para cada hora en el dataset.
- El script debe procesar todos los “trending topics” para encontrar entidades asociadas a los mismos en Wikidata. Por ejemplo, si tuviéramos los términos *“michael jackson”*, *“paro cardiorrespiratorio”*, *“ataque cardíaco”* y *“farrah fawcett”*, podrían asociarse, respectivamente, con las entidades `Q2831`, `Q202837`, `Q12152` y `Q102341`.
- **¡Atención!** No sobrecargues al servidor de Wikidata, evita repetir consultas sobre el mismo término.
- **Pistas:**
  - Wikidata proporciona un API REST que **debería usarse** en este ejercicio (en otras palabras, no utilices **SPARQL**).
  - `https://pypi.org/project/requests/`
  - `https://www.wikidata.org/w/api.php?action=wbsearchentities&language=en&format=json&search=michael+jackson`
  - `https://www.wikidata.org/w/api.php?action=wbgetentities&ids=Q2831&languages=en&format=json`

Se valorará positivamente encontrar sinónimos (por favor, no confundas encontrar esa información con el uso de sinónimos en *Elasticsearch* [1][2], cosa que es innecesaria).

Por ejemplo, el término *“rey del pop”* también se resuelve en Wikidata con la entidad Q2831 y, en consecuencia, es sinónimo del término *“michael jackson”*.

También se valorará positivamente asociar los términos no solo a la entidad de Wikidata sino a su tipo (para ello será necesario descargar la información sobre la entidad de Wikidata para saber de qué tipo instancia esa entidad concreta).

Entregable: Ejercicio 2 (4 puntos)

- Generar un volcado con todos los tweets relativos a un tema especificado mediante una consulta en un solo idioma (inglés o español). El volcado incluirá el autor, la fecha de creación y el texto del tuit; puede usarse TSV o *ndjson*.
- Dicho volcado debe ser tan exhaustivo como sea posible así que **la consulta inicial debe ser expandida** (es decir, será necesaria una segunda consulta con nuevos términos añadidos a los de la inicial); para ello, se utilizará la agregación de términos significativos y, atención, se eliminarán las palabras vacías (inglés, español).
- Por ejemplo, si el tema elegido fueran las protestas de Irán, podríamos usar `"lang:en AND text:iran"` como consulta inicial (suponiendo una consulta `query_string`). Se obtendrían entonces términos significativos como *"iranelection"*, *"democracy"*, *"neda"* o *"tehran"* (que son relevantes), y otros como *"obama"*, *"employment"* o *"badcredit"* (que son poco o nada relevantes o spam que se aprovechó del "trending topic").
- Cada estudiante debe elegir un tema (ejemplos) así como el número máximo de términos significativos a incluir en la consulta expandida (no se pueden seleccionar manualmente los términos para la expansión).
- Debe razonarse la consulta original escogida para el tema elegido y hay que aportar una reflexión crítica sobre los resultados obtenidos usando diferentes métricas y distintos números de términos.

El método de expansión de consultas que se describe aquí se denomina ***pseudo-relevance feedback***.

Es obligatorio utilizar varias métricas de las mencionadas cuando se explicó la agregación `significant_terms` (al menos dos diferentes).

También es obligatorio realizar una evaluación sistemática de los resultados, incluyendo la precisión lograda con cada configuración (porcentaje de documentos relevantes). Puesto que la relevancia es subjetiva debe evaluarse manualmente y solo se realizará sobre una selección de documentos (no más de 20).

Entregables: Ejercicio 3 (3 puntos)

- Revisa la documentación de las consultas ***"More Like This"***.
- ¿Se te ocurre alguna configuración para este tipo de consulta o una forma de producir una de esas consultas programáticamente de tal forma que se pueda emular la expansión de términos realizada en el ejercicio anterior?
- Si se te ocurre describe cómo hacerlo y escribe una función en Python que reciba como parámetro una consulta en el formato aceptado por `simple_query_string`, use internamente una consulta MLT y proporcione resultados a la persona usuaria.