Aplicación de técnicas de aprendizaje no supervisado para la clusterización temática de la red TOR

14 de marzo de 2016

UTILIZA MATEMÁTICAS

David Martín Bragado

Universidad de Alcalá

Índice

1. Introducción

El objetivo es clasificar el contenido de las páginas de la red TOR. Se calcula que en TOR durante el año 2013 existían 6500 servicios webs. Se ha creado un software capaz de recorrer los servicios web de la red TOR y descargar las paginas webs alojadas en los mismos. Con todos los datos descargados, mediante el uso de distintas técnicas de data science se ha realizado una clasificación general de los servicios webs. Todas las webs que se han analizado son de temática criminal.

2. Red TOR

2.1. ¿Qué es TOR?

TOR es una red abierta que nos permite navegar por Internet de forma anónima. Esta red cifra todo el tráfico impidiendo ser rastreado por terceros. Es usado principalmente por periodistas, activistas y opositores políticos, aunque también es usado por personas que, pese a dar un uso cotidiano a Internet, valoran la preservación de su anonimato. Además de anonimizar el tráfico, TOR permite alojar servicios web de forma anónima, dando lugar a los conocidos *servicios ocultos*.

2.2. Servicios ocultos

Los servicios ocultos son páginas webs las cuales sólo pueden ser visitadas a través del protocolo de TOR. La url de los servicios ocultos son de una longitud de 16 caracteres generados de forma aleatoria y un dominio *.onion*. Un ejemplo de url sería http://wikitjerrta4qgz4.onion/.

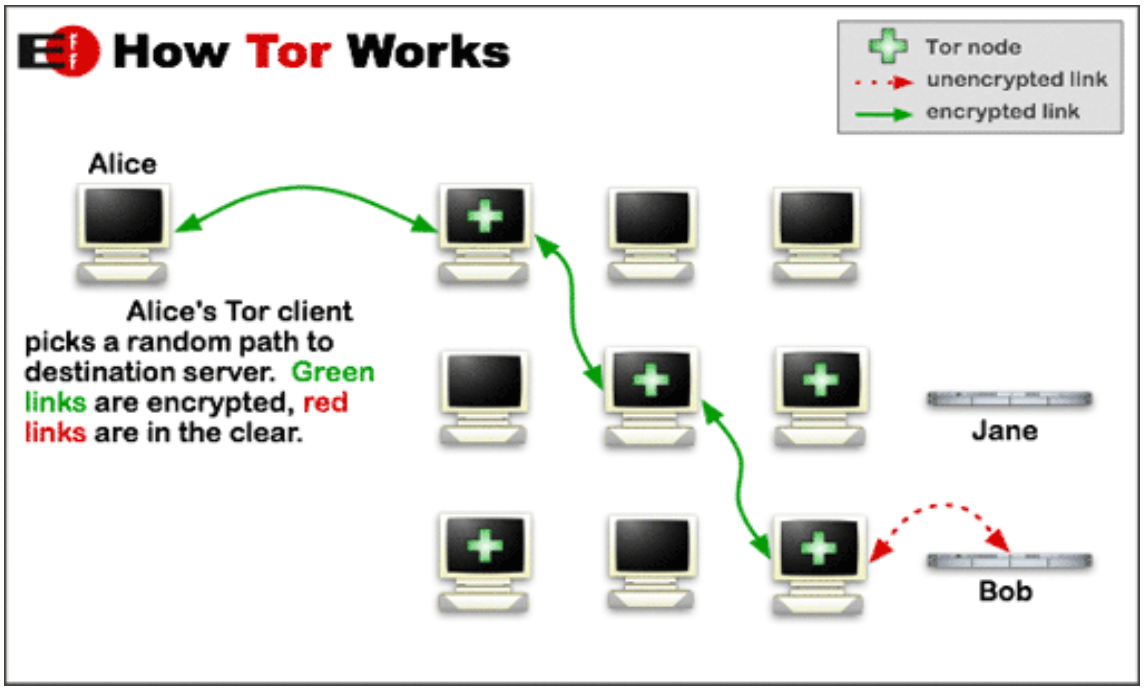

Figura 1: Funcionamiento de TOR

2.3. La importancia del anonimato

Debido al anonimato que nos ofrece TOR, muchos de los servicios web que ofrece son ilegales en gran cantidad de países. No obstante, desde el punto de vista del autor, estos servicios forman una pieza clave para evitar la censura. Ejemplo de ello son las distintas webs que denuncian la corrupción de gobiernos, las páginas de descarga de documentos considerados ilegales en ciertos países, los diferentes foros de opinión que permiten la libre expresión alrededor de distintos temas gubernamentales controvertidos, etc.

El problema de este anonimato es que también puede usarse de una forma amoral: mercados online donde comprar drogas, armas y tarjetas de crédito, foros sobre terrorismo, paginas dedicadas a la pornografía infantil, etc.

3. Obteniendo los datos mediante crawling

3.1. ¿Qué es un crawler?

Un ***crawler*** es un software capaz de navegar por la web de forma automatizada e ir recopilando información de las paginas webs que visita. El crawler emula el comportamiento de un usuario frente a una web. El objetivo es que parezca lo más humano posible para evitar ser detectado y así poder recopilar toda la información posible. Los crawlers también son conocidos por el nombre de *arañas web*.

3.2. Funcionamiento del crawler

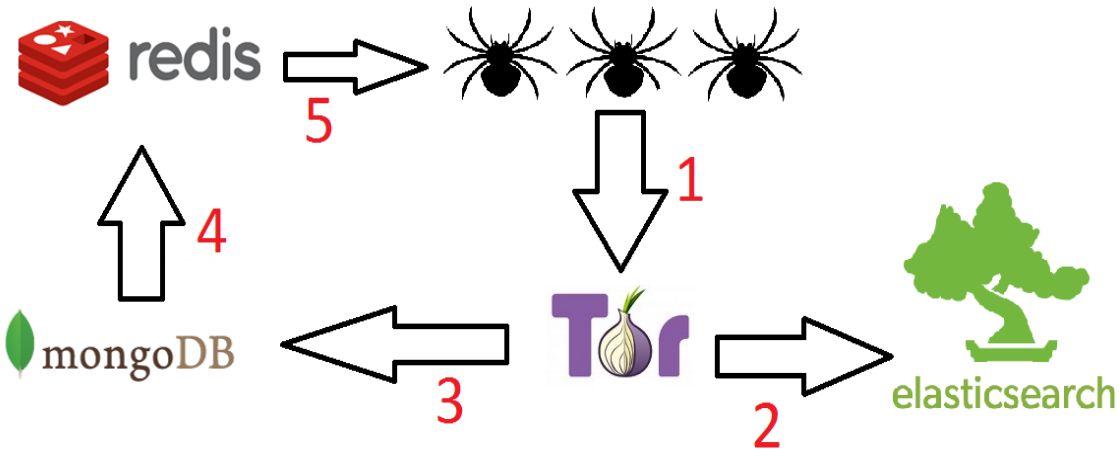

Figura 2: Esquema del funcionamiento del crawler

El funcionamiento del crawler está descrito en la ***Figura 2***.

1. Cada araña accede a una página de TOR de la cual obtiene su código fuente.
2. En una base de datos basada en ElasticSearch guarda la url a la que ha accedido la araña y el código fuente. El empleo de ElasticSearch tiene un doble objetivo: servir como base de datos de consultas y a la vez como un buscador semántico.
3. En mongoDB introducimos los enlaces que ha encontrado en la página web en 2 colecciones. La primera colección está destinada a realizar un mapeado mediante grafos de la red TOR, para así saber que páginas apuntan a otras. La segunda colección guardará todas las url que encuentra y gestionará su envío a las colas que implementa el motor de bases de datos en memoria Redis.

4. Las colas de Redis reciben las urls de mongoDB y las agrupan en una de las diferentes colas según la prioridad que queramos darle. Aquellas urls que pertenezcan a dominios nuevos o con pocas visitas entrarán en una cola con una prioridad alta, de lo contrario irán a una cola con una prioridad más baja.

5. Cada araña recibe una url de la cola con menos prioridad y repite de forma iterativa el proceso desde el paso 1.

4. Explicación del proceso de aprendizaje no supervisado

4.1. Fase 1 - Detección de la similitud entre los documentos

De forma previa a la detección de grupos temáticos en los textos contenidos en la red TOR, es necesario contar con una métrica de la similitud que existe entre ellos. Para ello se seguirá una técnica, denominada ***Bag of Words***, en la que cada uno de los documentos analizados representará un vector de términos, siendo cada uno de estos términos una dimensión. El valor asignado a cada dimensión será una ponderación de la importancia que tiene el término t_i en el documento d:

$$d_j \rightarrow \vec{d_t} = (w(t_1, d_j), ..., w(t_{|V|}, d_j))$$

A continuación se detalla cómo se modela el aporte de contenido semántico/información de un término en un documento.

4.1.1. Term Frecuency - Inverted Document Frecuency

Existe una primera etapa en la que se define la matriz de frecuencias de términos de los documentos (*DTM, Document-Term Matrix*). En ella, se indica, para cada término detectado, su frecuencia de aparición en cada uno de los documentos, resultando en una estructura como la siguiente:

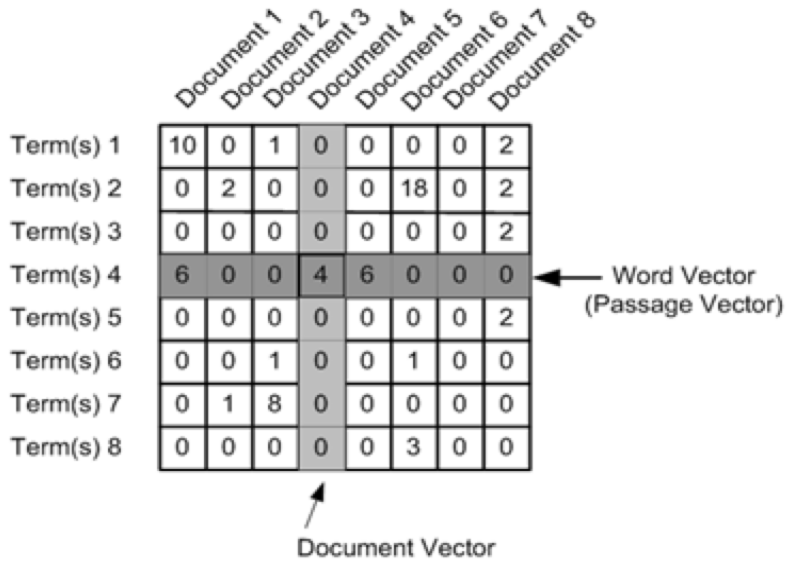

Figura 3: Term Document Matrix

Tras ello, trataremos de ponderar la relevancia semántica de cada n-grama [1] en la totalidad de los documentos: *aquellos términos que aparezcan de forma frecuente en un documento pero no tanto en el resto, recibirán un peso más elevado, ya que se entenderá que contienen mayor significado en relación a ese documento. Del mismo modo, un término que aparece 10 veces en un documento debería aportar bastante más contenido semántico que otro que aparezca tan solo en una ocasión.*

Entendiendo $Tf_{i,j}$, como la frecuencia del término t_i en el documento d_j, siendo Q el número de documentos de la colección y n_i el número de documentos donde aparece el término t_i, se define el *idf* del término t_i como:

$$idf_{t_i} = \log_{10}(Q/n_i)$$

De este modo, un n-grama que apareciese en todos los documentos, debería tener $idf = 0$ y uno que figurase en el 10 % de la colección, tendría $idf = 1$.

El modelo de puntuación *Tf - idf* determina el peso de cada n-grama sobre un documento como:

$$w(t_i, d_j) = Tf_i \times \log_{10}(Q/n_i)$$

Para la obtención de la matriz TF-IDF (*Term frequency-Inverse document frequency*) se hará uso del siguiente código:

```
1
2                   from sklearn.feature_extraction.text import TfidfVectorizer
3
4                   """ Definicion de los parametros de la matriz TF-IDF """
5
6                   tfidf_vectorizer = TfidfVectorizer(max_df=0.8, max_features↩
                       =200000, min_df=0.2, stop_words='english', use_idf=True, ↩
                       tokenizer=self.tokenize_and_stem, ngram_range=(1,3))
7
8                   tfidf_matrix = tfidf_vectorizer.fit_transform(text)
```

Como es posible apreciar, en la función ***TfidfVectorizer*** se especifican distintos argumentos interesantes, con objeto de refinar nuestra asignación de ponderaciones a los n-gramas. Estos son:

- ***max_df***: se trata de la frecuencia máxima con la que un grama podrá aparecer en la totalidad de los documentos para ser considerado. En nuestro caso, se establece a un 80 %.
- ***min_df***: de forma análoga a *max_df*, se ajusta la frecuencia mínima. Para nuestro estudio, a un 20 %.
- ***ngram_range***: son el número de palabras que podrán integrar los n-gramas considerados. Se ha acotado unigramas, bigramas y trigramas.

El formato con que será devuelta la matriz TF-IDF será una matriz en la que para cada documento se establece un valor absoluto sobre el peso de cada n-grama.

4.1.2. Similitud entre Vectores

En este punto es posible medir la similitud entre cada documento con respecto al resto de documentos del corpus. Pese a que sería posible emplear la distancia euclidiana, la variabilidad de la longitud de los

[1] En la disciplina del lenguaje natural, conjunto de n términos con un contenido semántico determinado

documentos afectaría a la métrica. En este sentido, y dado que cada uno de los documentos están representados como vectores, sería posible usar el coseno del ángulo formado entre los mismos como estimación de su similitud.

De este modo, si los documentos son iguales, el ángulo valdrá 0 y el coseno 1. Esta medida de similitud denominada como ***similitud coseno***.

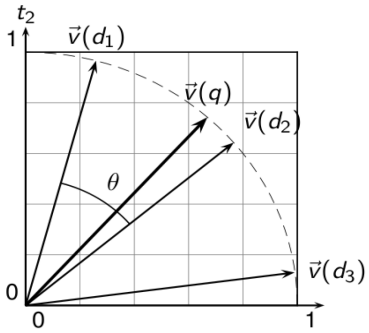

Figura 4: Medición de similitud a través del coseno

$$cos(d_1, d_2) = cos(\theta) = (d_1 \cdot d_2)/(|d_1| \times |d_2|)$$

Finalmente, calcularemos la distancia de cada documento con respecto al resto hallando la diferencia entre 1 y la similitud coseno.

```
"""  Distancia entre documentos  """
dist = 1 - cosine_similarity(tfidf_matrix)
```

4.2. Fase 2 - Clusterización jerárquica de los documentos

En este momento, una vez definidas las distancias entre cada uno de los documentos, se tratará de agrupar los mismos en distintos grupos o *clusters*. El objetivo (y reto) del proceso de clusterización será encontrar grupos de instancias tales que las mismas en un cluster sean similares entre sí (minimización de las distancias intra-cluster) y diferentes de las instancias en otros clusters (maximización de las distancias inter-clusters).

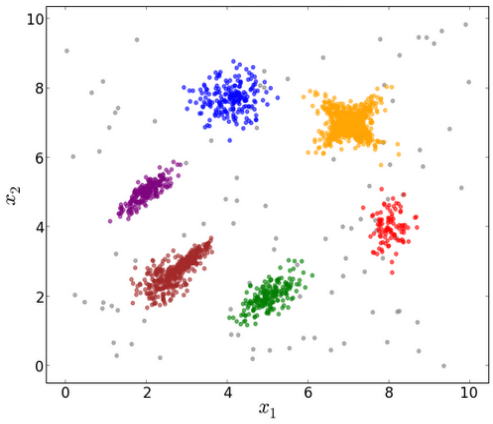

Figura 5: Agrupación de clusters claramente diferenciada

Pese a que existen distintos algoritmos de clusterización, para este análisis se ha escogido el de tipo ***Jerárquico Aglomerativo***. El principal motivo que ha llevado a escoger este mecanismo, es el desconocimiento inicial que se posee del número de grupos que integran nuestro conjunto de datos. La clusterización jerárquica produce taxones o clusters de diferentes niveles y estructurados de forma ordenada

Establecer una clasificación jerárquica supone poder realizar una serie de particiones del conjunto de individuos total $W = \{i_1, i_2, ..., i_n\}$; de forma que existan particiones a distintos niveles que vayan agregando a las particiones de los niveles inferiores.

El algoritmo que existe debajo de este mecanismo de clustering es el siguiente:

1. Cada documento extraido de TOR se definirá como un cluster. De este modo, la partición inicial tendrá la estructura $P = \{i_1\}, \{i_2\}, ..., \{i_N\}$
2. Se determinan los dos clusters más próximos i_i, i_j, y se agrupan en un único cluster, configurando la partición: $P = \{i_1\}, \{i_2\}, ..., \{i_i u i_j\}, \{i_N\}$
3. Se repetirá de forma iterativa el paso 2 hasta obtener la partición final $P_r = \{W\}$ en forma de un único cluster *padre*.

Ahora bien, ¿cómo se ha de determinar cuál esta *distancia mínima* entre clusters? El nivel de agrupamiento para cada fusión viene dado por un indicador llamado *valor cofenético* que debe ser proporcional a la distancia o disimilaridad considerada en la fusión (distancia de agrupamiento). Existen distintas alternativas en la determinación de este valor: distancia mínima inter-clusters, distancia máxima inter-clusters, método de la media, método de la mediana, etc.

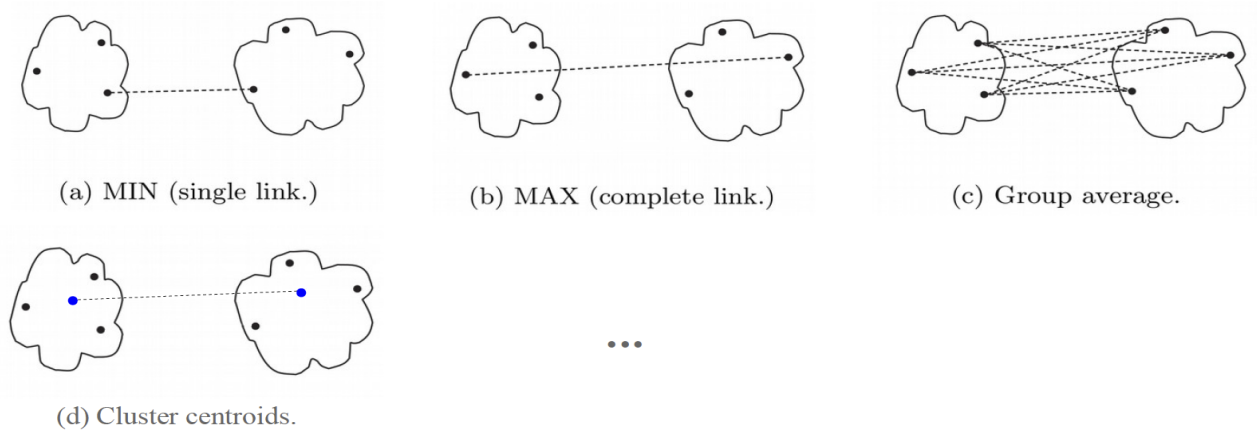

Figura 6: Distintas definiciones de distancia inter-clusters

Tras realizar un análisis de cuáles de estas técnicas se veía menos afectada por casos con ruido[2] (métodos de la distancia mínima y la distancia máxima) o por los grupos de gran tamaño (método del centroide), se optó por escoger el ***Método de Ward***.

Ward estipuló que la pérdida de información que se produce al integrar los distintos individuos en clusters puede medirse a través de la suma total de los cuadrados de las desviaciones entre cada punto (individuo) y la media del cluster en el que se integra. De este modo, propuso la siguiente estrategia de fusión: *en cada paso del análisis, considerar la posibilidad de la unión de cada par de grupos y optar por la fusión de aquellos dos grupos que menos incrementen la suma de los cuadrados de las desviaciones al unirse.*

[2]Casos en los que existen individuos perturbadores entre clusters bien diferenciados

Así, el coste de unión Δ de los clusters A y B sería:

$$\Delta(A,B) = \sum_{i \epsilon A \cup B} \| \vec{x}_i - \vec{m}_{A \cup B} \|^2 - \sum_{i \epsilon A \cup B} \| \vec{x}_i - \vec{m}_A \|^2 - \sum_{i \epsilon A \cup B} \| \vec{x}_i - \vec{m}_B \|^2$$

donde $\vec{m}_j$ es el centroide del cluster j.

El código necesario para realizar lo anterior, es el siguiente:

```
1            from scipy.cluster.hierarchy import ward
2            linkage_matrix = ward(dist)
```

entendiendo *dist* como la matriz de distancias de los documentos calculada en el paso previo.

El método ***Ward*** nos permite analizar cuales han sido sus decisiones. Así especifica, en forma de filas de una matriz, qué dos clusters ha decidido unir, cuál es la distancia entre ellos, y el número de instancias que componen dicho cluster.

En este punto, podemos graficar, a través del método ***dendrogram*** de la librería ***scipy.cluster.hierarchy*** un dendograma, que muestre la estructura jerárquica generada:

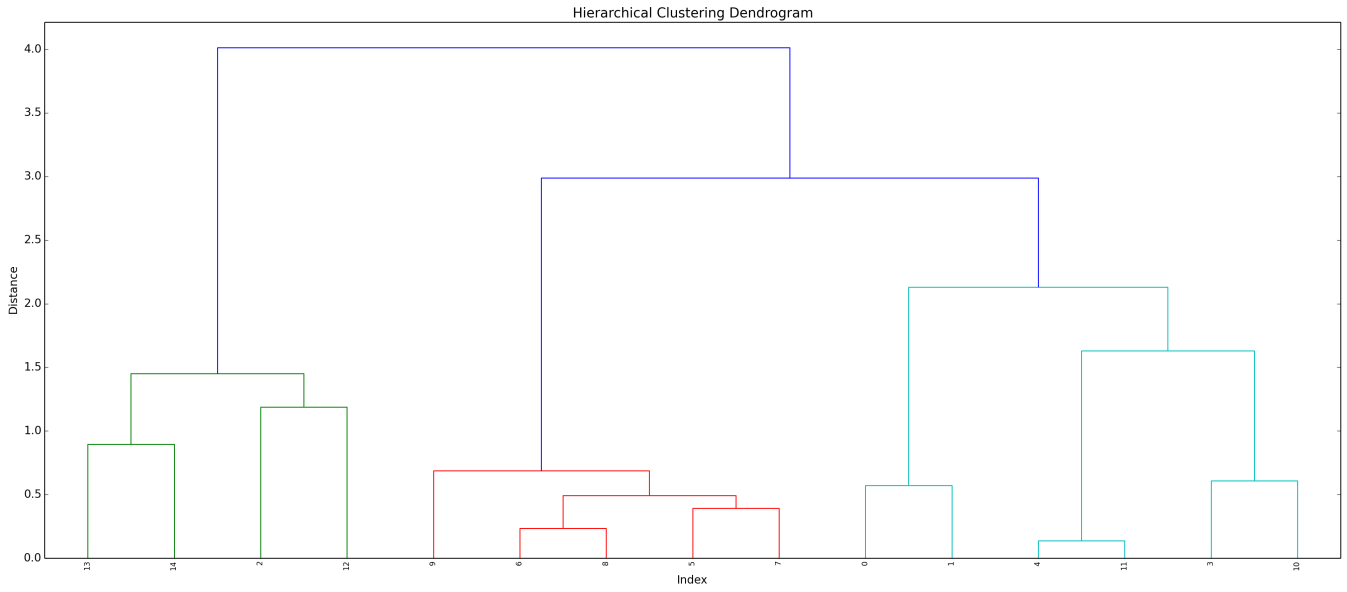

Figura 7: Dendograma resultante

El en ejemplo anterior, por motivos de ausencia de memoria disponible para cargar la totalidad de los datos descargados de TOR y por sobrecarga computacional, nos hemos limitado a estudiar un total de 15 documentos aleatorios. La configuración de una infraestructura de Big Data, nos permitiría computar de forma distribuida el grueso de datos del que se dispone.

No obstante es posible distinguir, a simple vista, la existencia de alrededor de tres clusters temáticos. Para recuperar esta estructuración en modo de lista, bastará con emplear el código siguiente:

```
1            from scipy.cluster.hierarchy import fcluster
2            num_clusters = 3
3            clusters = fcluster(linkage_matrix, num_clusters, criterion='↩
                maxclust')
```

Tras ello, extraemos los tópicos de cada cluster. Para ello, y tras realizar distintas labores de limpieza de texto (tokenización, filtrado de nombres propios, eliminación de *stopwrods*, etc.) se emplea el modelo estadístico LDA (*Latent Dirichlet Allocation*). Este, entiende que los documentos son una mezcla de diferentes tópicos y que cada palabra del documento es atribuible a los temas del mismo.

```
from gensim import models, corpora

""" Se crea un diccionario de los documentos que integran cada cluster"""
dictionary = gensim.corpora.Dictionary(documentos)
""" Se convierte el diccionario en una 'Bag of words' """
corpus = [dictionary.doc2bow(element) for element in text]
""" Se aplica LDA """
lda_multicore = gensim.models.LdaMulticore(corpus, num_topics=1, id2word=dictionary, iterations = 100, workers=cores_availables)
""" Se extraen los topicos """
topics_matrix = lda_multicore.show_topics(num_words = 5, formatted = False)
```

Finalmente, se ha obtenido un documento, en el que se indica, para cada cluster, los 5 términos más frecuentes. Este es el resultado:

Cluster	Tópicos
1	Sex, porn, pornography, erotica, xxx
2	Drugs, market,darknetmarket, agora, BTC
3	Weapons, guns, market, ammo, BTC

Como podemos comprobar existe una clara evidencia de tres grupos temáticos segmentados: de orientación sexual, estupefacientes y armas.

5. Aplicaciones prácticas

- Agrupación de documentos para explorarlos más rápidamente.
- Facilitar la búsqueda de presuntos criminales y monitorizar sus actividades en la red TOR.
- Monitorización de contenido pedófilo en la red TOR.

6. Futuras líneas de trabajo

- Empleo de otros algoritmos de clusterización y comparación de su ajuste.
- Una vez obtenidos resultados en un margen de error aceptable, emplear este conjunto como *dataset* de entrenamiento de modelos supervisados.
- Extensión a otros idiomas además del inglés.
- Ampliación del rango de aplicación a otras temáticas interesantes.
- Ejecución sobre una infraestructura Big Data para el procesamiento óptimo de *datasets* de gran tamaño.