algorithms

MDPI

Article

Counter-Terrorism Video Analysis Using Hash-Based Algorithms

David García-Retuerta [1], Álvaro Bartolomé [1], Pablo Chamoso [1,*] and Juan Manuel Corchado [1,2,3]

1 BISITE Digital Innovation Hub, University of Salamanca, Edificio Multiusos I+D+i, Calle Espejo 2, 37007 Salamanca, Spain; dvid@usal.es (D.G.-R.); alvarob96@usal.es (Á.B.); corchado@usal.es (J.M.C.)
2 Department of Electronics, Information and Communication, Faculty of Engineering, Osaka Institute of Technology, Osaka 535-8585, Japan
3 Pusat Komputeran dan Informatik, Universiti Malaysia Kelantan, Karung Berkunci 36, Pengkaan Chepa, Kota Bharu 16100, Kelantan
* Correspondence: chamoso@usal.es; Tel.: +34-677-522675

Received: 15 April 2019; Accepted: 20 May 2019; Published: 24 May 2019

Abstract: The Internet is becoming a major source of radicalization. The propaganda efforts of new extremist groups include creating new propaganda videos from fragments of old terrorist attack videos. This article presents a web-scraping method for retrieving relevant videos and a pHash-based algorithm which identifies the original content of a video. Automatic novelty verification is now possible, which can potentially reduce and improve journalist research work, as well as reduce the spreading of fake news. The obtained results have been satisfactory as all original sources of new videos have been identified correctly.

Keywords: video processing; image matching; big data; software tool

1. Introduction

Researchers have studied extremist strategies which had done between 70% and 80% of their networking online [1]. In addition, Schmid et al. [2] point out that "the willingness to look more critically at the role of the media and the Internet is notably absent in current counter-terrorism efforts in most democratic states". Radical groups have been using new propaganda techniques in the last few years. One that has been gaining continuous popularity is the joining of fragments of old terrorist attacks to make a seemingly new propaganda video. The aim of this work is to determine the sources of a newly released video efficiently and automatically.

The number of terrorist attacks declined worldwide between 2006 and 2018. Although in 2018 there have been fewer terrorist attacks than in 2006, the number of victims was similar. As explained by UNODC (United Nations Office on Drugs and Crime) [3], even though terrorists have found many ways of using the Internet in furtherance of illicit purposes, their online activity also gives us the opportunity to gather intelligence and evidence for the prosecution of such acts. Image intelligence has successfully been used to identify radical profiles on Social Networks [4] and other methods for identifying radicalization on Twitter have also been developed in recent years [5].

Olston et al. [6] stated that even though web crawling may seem a breadth-first-search application, it indeed experiences a wide range of problems, from system concerns to theoretical questions. The management of a very large data structure requires the definition of a system for the classification and storage of information. Thus, in this research the defined structure allows us to recover online videos that contain terrorist propaganda, analyze them, and extract conclusions.

As Meyer [7] describes: groups need to construct and maintain a collective identity, or more specifically, a belief in their own political or social efficacy. Radical groups assure the authenticity and novelty of their propaganda, which will be analyzed and whose original sources will be published. In addition, members of such groups trust their leaders and the fidelity of their propaganda. Their trust will be proven misguided.

Big technological companies (like Facebook or YouTube) are rumored to have similar methods of removing extremist videos automatically [8], but there have been no academic publications on that topic yet.

This paper combines web-scraping and video comparison to identify extremist sources automatically through the usage of proven methods. Having introduced the topic of this paper briefly, we further elaborate on it in the next six sections. In the context of locating the original sources of a new video, Section 2 briefly describes the algorithm used to compare videos and the criteria that are used to classify matching videos; videos match when the new video contains fragments of the source video. Section 3 describes the dataset used and the methodology details. Section 4 is related to the developed web crawler and its foundations. Section 5 describes the results and Section 6 provides some practical implementation details. Finally, Section 7 presents the conclusions and proposes future lines of research.

Retrieved videos have been accepted as genuine and results are satisfactory. This method performs well in real life situations and could be implemented as part of a counter-terrorism strategy.

2. Overview of the Method

This section gives an overview of the frames comparison chain, which is summarized in Figure 1. Implementation details are postponed until Section 6. This method is based on hash functions whose output will only keep the most relevant information about the original image [9]. Similar features have been used in recent years [10]. The basic idea is that images can be encoded so that only their most unique features are stored. This is achieved by causing a big information loss to the images and by making both images follow certain criteria. For better invariance to considerable changes in the original image, it is also useful to omit certain differences in the obtained hashes [11]. This can be done by establishing an appropriate threshold for the distance between the two hashes.

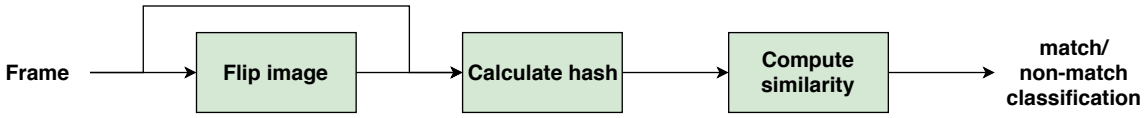

Figure 1. Method preview.

3. Datasets and Methodology

Datasets. We tested our detector with videos obtained by the web crawler described in Section 4 and comparing them against a fictitious propaganda video. The detector contains 2215 videos of different duration. The total size of this dataset is 34 GB and it contains 5,834,570 frames. Both videos are based on the same images. However, the colors, brightness, watermark, logos, resolution, and aspect ratio are different. Figure 2 shows some examples.

Methodology. We selected a fixed number of frames/second (f/s) and then extracted the corresponding frames from the videos. The hash database contains about 1,100,000 elements. The hash of each frame of the new video was compared with the whole database, and matching frames have been stored. This process was repeated several times, each time at different frame extraction rate to identify at which rate the system performs best. Databases were implemented using MongoDB and algorithm implementation was carried out in Python; an i7 7700 processor was used. Parallel computing is implemented for the CPU.

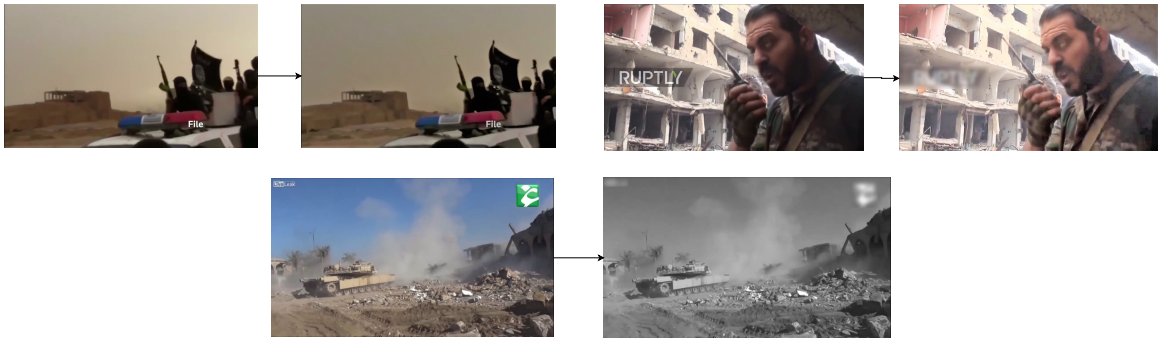

Figure 2. Frame examples.

4. Architectural Framework

This section presents the architectural framework for video extraction. The proposed framework is based on ETL (Extract, Transform and Load) techniques for Big Data, designed specifically for the extraction of terrorist propaganda videos from the Internet, using search engines such as Google or Bing. Big Data is the distinctive feature of this project because it allows for a distributed workflow in video retrieval, performing the process at a much higher speed than if the workflow were non-distributed. Hence a distributed system is more reliable, and it can be easily expanded for future work.

In terms of video retrieval, the main focus of the case study was to detect sample videos of terrorist attacks by launching crawler instances with terrorism-related keywords as inputs. Since terrorism is a sensitive topic, search engines such as Google and Bing need to be analyzed to check what kind of videos they provide. After some manual searches looking for terrorism videos, it was concluded that keywords: *terrorism* and *attack* together gave better results than using single word inputs or just radical group names. By better results we mean a higher number of real terrorist attack videos or their fragments. After watching the retrieved videos via web crawling, it was determined that the estimated length for a terrorist attack video was usually under 5 min, because when the video was longer it tended to be either a documentary film, a reportage or a fiction video (from movies or video games). Hence, a limit of 5 min has been established as the maximum length per video, this helps prevent the overloading of the system with useless videos while providing more concrete results.

As mentioned before, the ETL techniques implemented in the *distributed system* use web crawlers to retrieve Google and Bing search results, as described in Figure 3. All the search results are obtained from Google's and Bing's Video Section, so video retrieval is based on the input keywords and additional parameters offered by both search engines e.g., such as the video length.

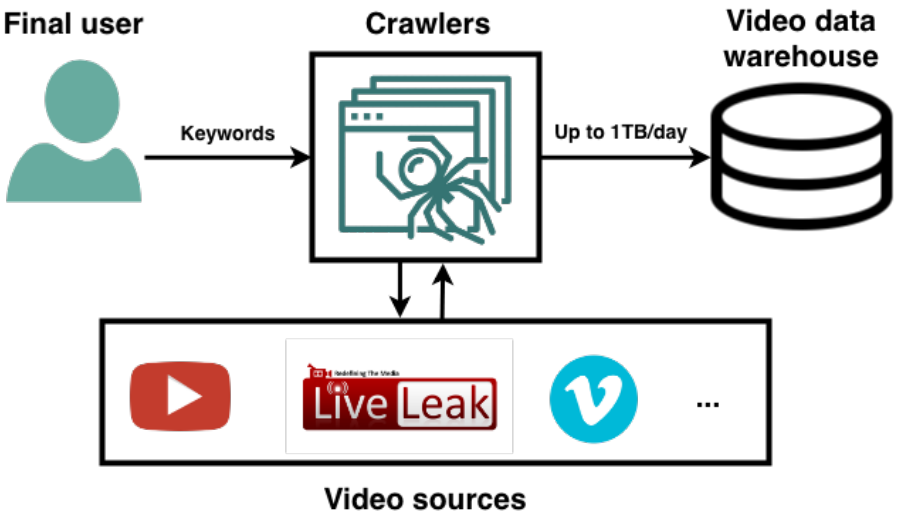

Figure 3. Proposed system for storage of retrieved videos.

Having explained in detail the different elements of the video retrieval architecture, below we outline every principle of the ETL process and its application:

- **Extract**: the data retrieved from Google and Bing Crawlers, such as links to terrorist attack videos, are used as the entry point of the extraction process. As established, the length of the videos whose information is extracted is always under 5 min. Once all the links are listed in MongoDB, the output of the Web Crawler goes directly to a noSQL database, as a result those links need to be transformed into useful data i.e., videos.
- **Transform**: this process, which in this case can be broken down into two steps, entails transforming data into a desired output:

    1. The video is downloaded using a stream and it is written in the system as an MP4 file for its subsequent analysis.
    2. Parse and decompose the video into frames that are individually converted into a hash.

- **Load**: afterwards all this data is inserted in MongoDB as a single document for every video, containing the respective hash for every listed frame. This is the final output of the ETL process, hence videos can now be processed and analyzed as explained before.

As mentioned above, to retrieve videos from search engines such as Google or Bing, it is necessary to develop web crawlers to get the links of the indexed videos on the HTML DOM Tree of the resulting webs or every search engine. Once we have all the video links listed, we use a buffer where we dump all the retrieved data via stream so it can be saved later as an MP4 video file. We propose an application based on web crawlers where we specify the input (keywords introduced into Google and Bing search engines) and the web crawlers automatically retrieve all the links indexed in their respective video pages, download them, and store them as MP4 files.

Once crawlers were implemented, some tests were made to check their accuracy and efficiency when it came to retrieval of terrorism videos. Tests showed a performance of up to 2500 videos in an hour of video retrieval since first the crawler started, involving both the scraping of Google or Bing video page and its consequent download. This meant that using crawlers for bulk video downloading was clearly more efficient than doing it manually, just like Mike Thelwall [12] briefly explained as "computer programs are needed in order to conduct any large-scale processing of web pages, requiring the use of a web crawler at some stage in order to fetch the pages to be analyzed", so its need on bulk video retrieval is justified as there are a lot of videos to retrieve and pages to crawl, which cannot be done manually and so on, this task needs a crawler to be developed.

5. Overview of the Results

Our final method has been tested against watermarks, scoreboards, texts, logos, color filters, and other image alterations. In addition, widespread techniques for avoiding copyright (such as color filters, re-sized images and even flipped images) have also been thoroughly checked. Our method has proved to be resistant to all these editing techniques, independently of whether one or several have been applied to a single video. Error rate is below 4×10^{-8}% in all our tests, which are meant to simulate all aspects of a real-world case. Two groups of datasets have been used for testing; terrorism-related retrieved videos and videos of the Football World Cup held in Russia in 2018. In both cases the algorithm have successfully identify all the sources.

The configuration which provided the best performance was a 5 f/s extraction rate in the original videos and a 12 f/s extraction rate in the new videos, a threshold of 3 units. However, we recommend using a 3–5 f/s configuration with a threshold of 4 units if fast search results of new videos are required. An example of the obtained results can be found in Figures 4 and 5.

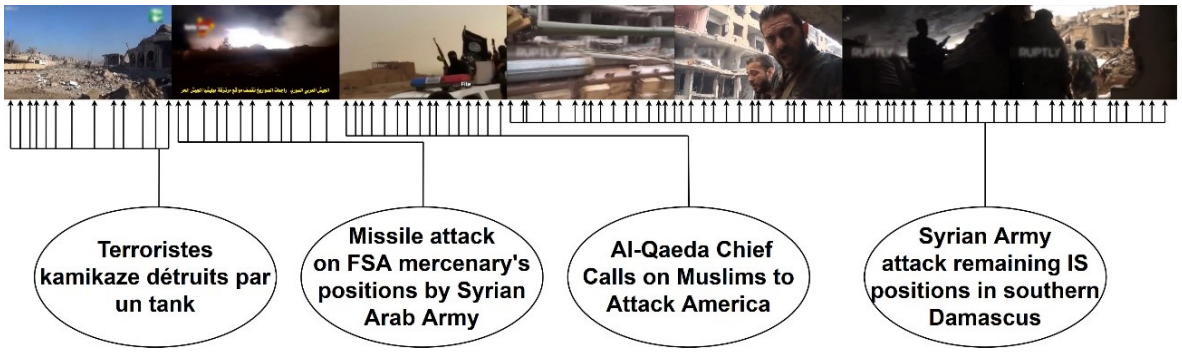

Figure 4. Schema of the process used to convert videos into hashes.

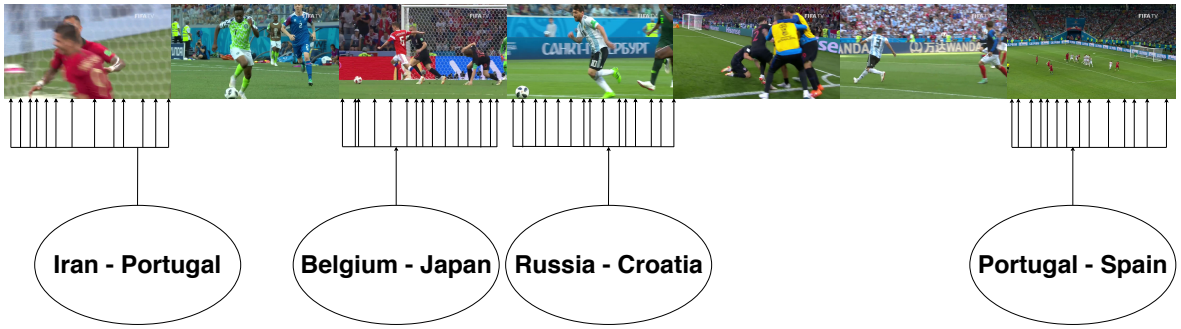

Figure 5. Schema of the process used to convert videos into hashes.

6. Implementation and Performance Study

We now give the details of our program and study the effects of various choices depending on the performance. Throughout this section, results are obtained from our default detector which has the following properties: frame extraction rates of 5 f/s and 3 f/s respectively; threshold of 4 units.

6.1. Frame Extraction Rate Choice

Our experiments try to find the perfect balance between enough frames to find matches and not enough to cause false positives. A 5% probability of getting a false positive from any given video is set to be the maximum, so we apply the following restriction:

$$n_{originalframes} \cdot n_{newframes} \cdot \mathbb{P}[false\ positive] \leq 0.05$$

where $\mathbb{P}[false\ positive] = 4.011564 \times 10^{-10}$ as calculated in our tests, $n_{originalframes}$ refers to the number of frames of the original video and $n_{newframes}$ refers to the number of frames of the new video. Our tests showed that the best performance is achieved with a 3–5 f/s rate in the new video and the original video respectively, which resulted in 1,166,914 and 5370 frames. It has a 1.214% error rate at detecting copied videos.

6.2. Choice of Threshold

To ensure that the system detects matching videos it is necessary to establish a suitable threshold; the maximum permitted distance between two frames to consider them matching frames. This may cause some false positives, but its probability is reduced by the frame extraction rate choice. Furthermore, they can be dismissed *a posteriori* by a person. Videos with less than three matching frames should by default be checked by a person, as our tests have shown that videos with false positives often have just one matching frame, exceptionally two.

We use a threshold of 4 units for our results (frames with up to 6.25% hash difference are considered matching), with a 3–5 f/s extraction rate. However, a threshold of 3 units (4.69% hash

difference) did prevent all false positives at the expense of setting a higher frame rate for the correct detection of all matching videos, which caused the processing time to triple. The high frame rate was at 5–12 f/s, but this case is not described in the present text due to performance concerns.

6.3. Hash Generation

The process for generating the hash associated with a frame is described below:

1. **Calculating the average pixel value.** We must input an 8-bit greyscale image, so that the input can be understood as a matrix in which elements are positive and lower than 256.
2. **Generating a chain of 1 s and 0 s.** We apply the following function to the values of the matrix, from top to bottom:
$$f(pixel) = \begin{cases} 0 & pixel \leq average \\ 1 & pixel > average \end{cases}$$
3. **Converting the chain to hexadecimal.** We consider the previous chain to be a binary number and change its base to 16. The resulting number must have a length of 64 characters, so 0 s may have to be added at the beginning.

6.4. Pre-Processing Algorithm

The first proposed algorithm is responsible for converting any given video into a hash sequence.

First, it divides the video into frames and selects the correct frame rate per second. If necessary, it rotates the images anti-clockwise so that they become horizontal and flips them so that the darkest half is on the left. Then, it resizes the images to 16×16 pixels, converts them to black and white, generates the associated hash (Section 6.3) and dismisses the hash if it is constant.

As a result, the method is resistant to flipped videos (which is a common practice to avoid copyright). We get a great size reduction too, 99.93% in our tests.

When applied to our database (which contains over 5 million frames), it pre-processed the videos in 7 h, 46 min, and 24 s.

6.5. Matching Algorithm

Our second algorithm is responsible for comparing the frames and deciding whether they originated from the same image or not.

It calculates the Hamming Distance of two hashes. If it is lower than a certain threshold it will be considered a match.

Basically, most of the irrelevant information was lost during the previous process but some pieces may go through it all. The threshold is the last barrier to distinguishing edited images.

When applied to our database which contains 5370 frames, against a 3-min video, it found the matching frames in 2 h 32 min and 42 s. Please note that we are using a 4-core CPU for a program which admits a very high degree of parallelization; the performance could be greatly improved by hardware.

6.6. Pseudocode

The pseudocode for the image rotation algorithm is shown in Algorithm 1. The pseudocode for the hash generation algorithm is also presented below as Algorithm 2.

Algorithm 1 Image rotation algorithm

function IMAGEROTATION(I_0)
  $width$ = getWidth(I_0)
  $height$ = getHeight(I_0)
  if $width < height$ **then**
    I_0 = rotate(I_0,90) ▷ Longest image side over x-axis (landscape)
  end if
  $nsI = I_0[0 : width, 0 : height/2]$ ▷ Get upper half of the image
  $ssI = I_0[0 : width, height/2 : height]$ ▷ Get lower half of the image
  if $\overline{nsI} < \overline{ssI}$ **then** ▷ Highest tonality on top
    rI = rotate(I_0,180)
  else
    $rI = I_0$
  end if
  return rI
end function

Where I_0 is the original frame (input), and rI is the rotated image (output).

Algorithm 2 pHash-based algorithm

function GETIMAGEHASH(I_0)
  sI = reduceSize(I_0, 16, 16) ▷ Reduce size to 16 × 16 pixels
  gI = greyScale(sI) ▷ Convert to greyscale
  rI = imageRotation(gI) ▷ Rotate as defined in Algorithm 1
  $mean$ = calculateAverage(rI) ▷ Get the average pixel value
  for each $px \in rI$ **do**
    if $px > mean$ **then** ▷ Compare every pixel against the mean
      $hash = hash + 1$
    else
      $hash = hash + 0$
    end if
  end for
  $hash$ = convertHexadecimal($hash$) ▷ Convert from binary to hexadecimal
  return $hash$
end function

Where I_0 is the original frame (input), and $hash$ is the resulting hexadecimal number (output).

6.7. Structure

Our implementation consists of two programs and two databases as we believe it to be easily implemented in real-world situations.

Program 1 implements the pre-processing algorithm and writes the hashes in *database 1*.

Program 2 takes as input the hashes stored in *database 1* and a new video from which hashes are extracted using the pre-processing algorithm. Afterwards, it applies the matching algorithm to the corresponding frames and writes the matches in *database 2*.

Therefore, *database 1* stores our hashes, their respective video title and frame position. *Database 2* stores hash matches. It contains the title of the (original) video which had a match, and the positions of the frames which matched.

Figure 6 represents *program 1* and the initial part of *program 2*.

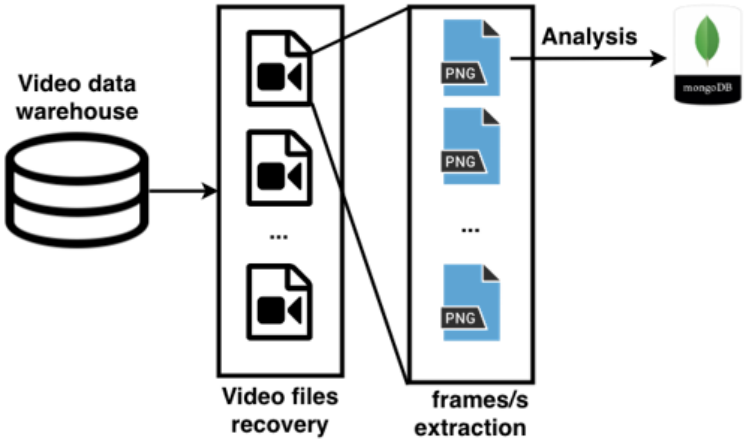

Figure 6. Schema of the process used to convert videos into hashes.

7. Conclusions and Future Work

This algorithm provides a good method for tracing back a video and discovering its origins. Common techniques used to subtly modify the original frames can be easily detected in our video analysis. Therefore, this algorithm makes the new efforts of extremist groups become obsolete.

Although our current method is resistant to the most widespread video editing tricks used to copy videos, we are also working on the detection of cropped images and added borders. The algorithm has been tested with videos related to the football World Cup 2018 held in Russia, obtaining equally satisfactory results. Another goal is to further improve the performance of the system, extending this method to greater scopes, and optimizing it for GPU parallel computing.

Author Contributions: Á.B. and D.G.-R. developed the system. P.C. reviewed the of the state of the art and conducted the case study. J.M.C. formalized the problem, designed the methodology and reviewed the work. All the authors contributed to the redaction of the paper.

Funding: This research has been partially supported by the European Regional Development Fund (ERDF) within the framework of the Interreg program V-A Spain-Portugal 2014-2020 (PocTep) under the IOTEC project grant 0123_IOTEC_3_E.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Koehler, D. The radical online: Individual radicalization processes and the role of the Internet. *J. Der.* **2014**, 116–134.
2. Schmid, A.P. Radicalisation, de-radicalisation, counter-radicalisation: A conceptual discussion and literature review. *ICCT Res. Pap.* **2013**, *97*, 22. [CrossRef]
3. Ki-moon, B. *The Use of the Internet for Terrorist Purposes*; United Nations: New York, NY, USA, 2012; pp. 12–13.
4. López-Sánchez, D.; Corchado, J.M.; Arrieta, A.G. Dynamic Detection of Radical Profiles in Social Networks Using Image Feature Descriptors and a Case-Based Reasoning Methodology. In Proceedings of the International Conference on Case-Based Reasoning, Stockholm, Sweden, 9–12 July 2018; Springer: Cham, Switzerland, 2018; pp. 219–232.
5. López-Sáncez, D.; Revuelta, J.; de la Prieta, F.; Corchado, J.M. Towards the Automatic Identification and Monitoring of Radicalization Activities in Twitter. In Proceedings of the International Conference on Knowledge Management in Organizations, Seville, Spain, 18–20 September 2018; Springer: Cham, Switzerland, 2018; pp. 589–599.

6. Olston, C.; Najork, M. Web crawling. *Found. Trends Inf. Retr.* **2010**, *4*, 175–246. [CrossRef]
7. Meyer, D.S.; Staggenborg, S. Movements, countermovements, and the structure of political opportunity. *Am. J. Soc.* **1996**, *101*, 1628–1660. [CrossRef]
8. Reuters. Facebook and YouTube Use Automation to Remove Extremist Videos, Sources Say. 25 June 2016. Available online: https://www.theguardian.com/technology/2016/jun/25/extremist-videos-isis-youtube-facebook-automated-removal (accessed on 24 May 2019).
9. Chamoso, P.; Rivas, A.; Sánchez-Torres, R.; Rodríguez, S. Social computing for image matching. *PLoS ONE* **2018**, *13*, e0197576. [CrossRef] [PubMed]
10. Rosebrock, A. Fingerprinting Images for Near-Duplicate Detection. Available online: https://bit.ly/2AvDwjo (accessed on 3 January 2019).
11. Chamoso, P.; Rivas, A.; Martín-Limorti, J.J.; Rodríguez, S. A hash based image matching algorithm for social networks. In Proceedings of the International Conference on Practical Applications of Agents and Multi-Agent Systems, Porto, Portugal, 21–23 June 2017; Springer: Cham, Switzerland, 2017; pp. 183–190.
12. Thelwall, M. A web crawler design for data mining. *J. Inf. Sci.* **2001**, *27*, 319–325. [CrossRef]