GeoSensor: Semantifying Change and Event Detection over Big Data

Nikiforos Pittaras[1,2], George Papadakis[2], George Stamoulis[2], Giorgos Argyriou[2], Efi Karra Taniskidou[2], Emmanouil Thanos[2], George Giannakopoulos[1] and Manolis Koubarakis[2]
[1]NCSR Demokritos, Greece {pittarasnikif, ggianna}@iit.demokritos.gr,
[2]Department of Informatics and Telecommunications, National and Kapodistrian University of Athens, Greece
{npittaras, gpapadis, gstam, gioargyr, efikarra, ethanos, koubarak}@di.uoa.gr

ABSTRACT

GeoSensor is a novel, open-source system that enriches change detection over satellite images with event detection over news items and social media content. GeoSensor combines these two orthogonal operations through state-of-the-art Semantic Web technologies. At its core lies the open-source, semantics-enabled Big Data infrastructure developed by the EU H2020 BigDataEurope project. This allows GeoSensor to offer an on-line functionality, despite facing three major challenges of Big Data: Volume (a single satellite image typically occupies a few GBs), Variety (its data sources include two different types of satellite images and various types of user-generated content) and Veracity, as the accuracy of the end result is crucial for the usefulness of our system. We present GeoSensor's architecture in detail, highlighting the advantages of using semantics for taking the most of the knowledge extracted from news items and Earth Observation products. We also verify GeoSensor's efficiency through a preliminary experimental study.

KEYWORDS

big data, satellite data, linked data, change detection, event detection

ACM Reference Format:
Nikiforos Pittaras[1,2], George Papadakis[2], George Stamoulis[2], Giorgos Argyriou[2], Efi Karra Taniskidou[2], Emmanouil Thanos[2], George Giannakopoulos[1] and Manolis Koubarakis[2]. 2019. GeoSensor: Semantifying Change and Event Detection over Big Data. In *The 34th ACM/SIGAPP Symposium on Applied Computing (SAC '19), April 8–12, 2019, Limassol, Cyprus.* ACM, New York, NY, USA, Article 4, 8 pages. https://doi.org/10.1145/3297280.3297504

1 INTRODUCTION

In remote sensing, *change detection* is the process of comparing two or more satellite images that depict the same area on the Earth surface, but are taken at different points in time [25, 33]. Its goal is to identify differences between the images in the form of areas with changes in land cover or land use (e.g., an area that was an olive grove in the past is now occupied by buildings). This is a crucial task, as it provides useful information for many applications, e.g., studying land cover evolution, monitoring natural disasters or support to crisis management. As an example, consider Figures 1(a) and (b), which depict snapshots of Ukhiya, Chittagong, Bangladesh before and after the settlement of Rohingya refugees on October, 2017. In situations like this, change detection allows for fast and accurate estimation of natural or man-made changes on the Earth surface, providing valuable support to decision-makers. In our example, the outcomes of change detection appear in Figure 1(c). Modern satellite technology makes this possible even for remote areas with humanitarian or security issues that are difficult to reach.

Interest in change detection using satellite images has grown recently, due to the availability of long time series of images by flagship Earth observation programmes, such as the US Landsat program[1] and the EU *Copernicus Programme*[2]. The latter is currently the world's largest Earth observation programme with almost 20 satellites, called *Sentinels*, expected to be in orbit by 2030. The Copernicus Programme already consists of a set of complex systems that collect data from satellites as well as in-situ sensors, providing users with reliable and up-to-date information on a range of environmental and security issues under a free, full and open data policy. Information extracted from this data is also made freely available to users through the *Copernicus services*[3], which address six thematic areas: land, marine, atmosphere, climate, emergency and security. Techniques for change detection using time series of satellite images are important in all of these areas [7].

To the best of our knowledge, though, there is no open-source system that addresses the following three Vs of Big Satellite Data:

- *Volume* stems from the combined effect of the inherently quadratic time complexity of change detection and the large size of satellite images. In the worst case, all pixels of the one image have to be compared with all pixels of the other image, yielding a rather time-consuming procedure for a common pair of images - each image typically occupies few GBs, containing millions of pixels of low resolution (i.e., each pixel corresponds to tens of square meters on the Earth surface). Apparently, change detection poses a quite challenging computational task for commodity hardware.
- *Veracity* requires that decision makers are able to assess the quality and correctness of the intelligence extracted from satellite images, based on relevant news content. In practice, this means that *collateral information* about news should provide reliable insights into the detected changes, ideally on real-time.
- *Variety* emanates from the diverse types of images that are produced by each satellite constellation. The two polar-orbiting

SAC '19, April 8–12, 2019, Limassol, Cyprus

ACM ISBN 978-1-4503-5933-7/19/04…$15.00
https://doi.org/10.1145/3297280.3297504

[1]https://landsat.usgs.gov
[2]http://www.copernicus.eu
[3]http://www.copernicus.eu/main/services

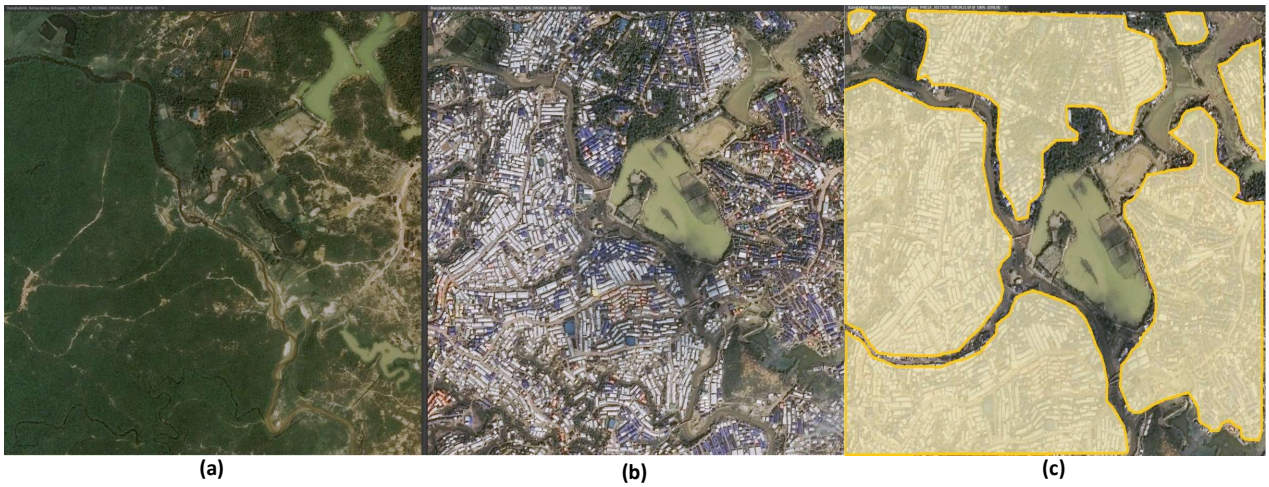

Figure 1: Satellite images showing Ukhiya, Chittagong, Bangladesh (a) before, and (b) after the Rohingya refugee crisis on October, 2017. (c) shows the main areas with changes in land cover or land use as identified by GeoSensor.

satellites of the Sentinel-1 constellation are equipped with C-band Synthetic Aperture Radar (SAR) imaging systems, which enable image acquisitions regardless of weather and light conditions (i.e., the sensor is able to acquire images in the presence of clouds and during night time). In contrast, the two polar-orbiting satellites of the Sentinel-2 mission provide High-Resolution Optical data, acquired by a wide swath high-resolution multispectral sensor. Their images have 12 spectral bands, covering the spectrum from the visible domain to the short wavelength infrared domain. Being an optical passive system, imaging is sensitive to weather conditions and depends on external illumination. Variety further increases due to the textual data that are necessary for addressing Veracity.

In this work, we present GeoSensor, a geospatial system that applies change detection to Copernicus data in a way that addresses these three Vs of Big Satellite Data. In essence, GeoSensor integrates a remote sensing component with a social sensing one into a highly scalable processing chain. Remote sensing applies change detection techniques to SAR images from Sentinel-1, while using optical Sentinel-2 images for the validation of the end result. Social sensing applies event detection techniques to cluster together news items and social media posts that pertain to the same real-world event and are located in the area, where change detection took place. For example, Figure 2(a) depicts a cluster of news items that elucidates the changes appearing in Figure 1(c). The integration of these two orthogonal components relies on Semantic Web technologies.

The rest of the paper is structured as follows: Section 2 briefly discusses related work, while Section 3 delves into GeoSensor's architecture, highlighting the three workflows that lie at its core. In Section 4, we present preliminary experiments over real-world data the demonstrate the scalability of our system and in Section 5, we conclude the paper along with directions for future work.

2 RELATED WORK

Change Detection. *Earth observation* is the use of remote sensing technologies to monitor land, marine and atmosphere. Satellite-based Earth observation relies on the use of satellite-mounted payloads to gather imaging data about Earth characteristics. We can distinguish two kinds of remote sensing. *(i)* In *passive remote sensing*, the satellite instruments monitor the energy received from the Earth, due to the reflection and re-emission of the Sun's energy by the Earth's surface or atmosphere. Optical or thermal sensors are commonly-used passive sensors (e.g., Sentinel-2 images). *(ii)* In *active remote sensing*, the satellite sends energy to Earth and monitors the energy received back from the Earth's surface or atmosphere, enabling day and night monitoring during all weather conditions. Commonly used active sensors are lasers and radar images, like the SAR images provided by Sentinel-1.

Recent works on change detection use Deep Neural Networks [19, 20] in a data-driven fashion, performing classification to detect changes in pixels or areas in the images. Other works use hierarchical object-based classification methods [10]. Such *supervised algorithms*, though, lie out of our scope, due to the lack of publicly available labeled datasets. Developing such datasets from scratch is a rigorous process that requires heavy human involvement, even in-situ inspection of identified changes.

Instead, GeoSensor considers *unsupervised algorithms* for change detection. At the moment, it is equipped with the established approach implemented in ESA's SNAP Toolbox[4]. Yet, its modular architecture allows for seamlessly extending it with additional state-of-the-art approaches, like the clustering technique in [12].

Event Detection. A review of text event detection is presented in [37], with more recent surveys covering a large variety of detection methods that are crafted for social media [4, 31]. In [8], the authors utilize a semantically-enabled convolutional neural network (CNN) to categorize social media posts, reporting that their model outperforms TF-IDF and Word2Vec pre-trained embeddings.

[4] http://step.esa.int/main/toolboxes/snap

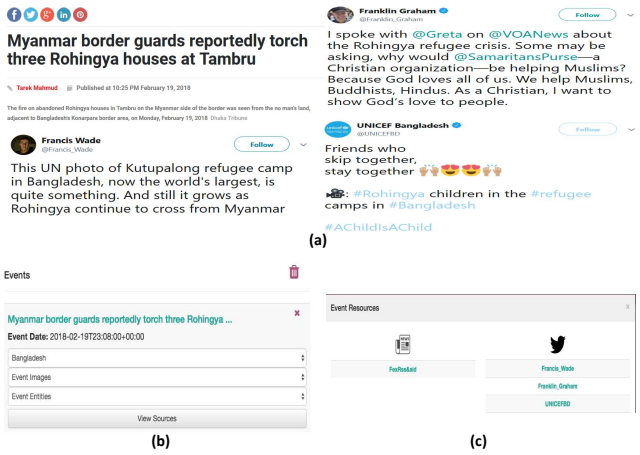

Figure 2: (a) A set of news items referring to the Rohingya refugee crises, (b) the corresponding event created by GeoSensor, and (c) the menu providing access to the individual news items of the event.

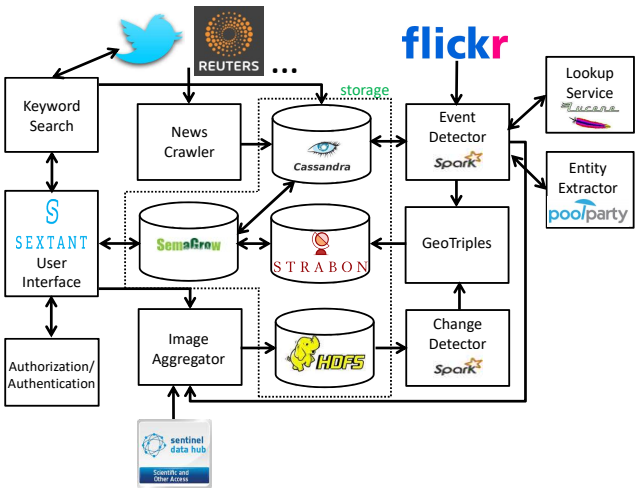

Figure 3: The system architecture of GeoSensor.

Other works incorporate CNNs for the joint detection of events and topics [8, 11, 28]. Yet, these methods rely on supervised learning, requiring a labeled dataset, unlike our unsupervised approach.

On another line of research, several works use unsupervised, semantically-aware clustering for event detection. For example, a semantically rich multiple-vector representation is used in [26, 27], while [30] uses a co-occurrence-based semantic expansion of words to produce event groups. These works report superior performance over non-semantic baselines. In [34], the authors employ a classification-based cleaning phase that is followed by content- and temporal-based clustering. [1] performs a clustering on keyword-based features over tweets, while the structure of the underlying social network lies at the core of the approaches presented in [2, 21]. However, all these works mainly rely on vector space features that capture frequency-related statistics, ignoring the positional information of tokens in the source text (i.e., bag of n-grams). In contrast, our approach relies on graphs of n-grams, which effectively capture token context both in long, curated documents like news articles and in short, noisy texts like tweets [3, 18, 32].

3 APPROACH

We now present GeoSensor, explaining how it addresses the above three Vs of Big Satellite Data.

To tackle Variety, GeoSensor relies heavily on state-of-the-art Semantic Web technologies, which provide time efficient, unified access to the outcomes of the remote and the social sensing components. In this way, it is capable of seamlessly processing a rich diversity of data sources, which range from the graphic information in SAR and optical satellite images to the textual information of news articles and social media posts.

To address Volume, GeoSensor exploits the distributed processing of a cluster based on the *BDI platform* [5], the open-source, semantics-enabled Big Data infrastructure that was developed in the context of the EU BigDataEurope[5] project. The BDI platform combines the massive parallelization capabilities offered by Apache Spark[6] with an inherent support for Semantic Web technologies.

To tackle Veracity, GeoSensor uses Sentinel-2 images in combination with the knowledge extracted from the social sensing component for the verification of changes detected from Sentinel-1 images. For better verification, GeoSensor is also able to fetch the latest raw data from social media through the live Twitter keyword search that is offered by its GUI.

Figure 3 depicts GeoSensor's architecture. It consists of 11 components that are organized into 3 workflows, one for each horizontal layer: the *change detection layer* is formed by the components at the bottom (i.e., Image Aggregator, HDFS and Change Detector), while the *event detection layer* is implemented by the components at the top (i.e., News Crawler, Apache Cassandra, Event Detector, Lookup Service and Entity Extractor). The rest of the components comprise the *semantic layer*, which acts as GeoSensor's backbone. Next, we describe the functionality of each layer in detail.

3.1 Change Detection Layer

This layer implements the gist of GeoSensor, retrieving and comparing pairs of satellite images in order to detect changes in land cover or land use. It consists of three components.

The first one is the **Image Aggregator**, a RESTful web service that downloads from ESA's Copernicus Open Access Hub[7] the pairs of Sentinel-1 and Sentinel-2 images with the largest overlap with the user-defined area of interest. In our example, the Image Aggregator is responsible for downloading the images in Figure 1(a) and (b), after the user specifies Ukhiya, Chittagong, Bangladesh as the area of interest. This process requires also the user to define temporal acquisition criteria, in the form of the images' sensing dates, i.e., the time of interest together with a reference date in the past, before the change for took place. In our example, the time of

[5] https://www.big-data-europe.eu
[6] https://spark.apache.org
[7] https://scihub.copernicus.eu/

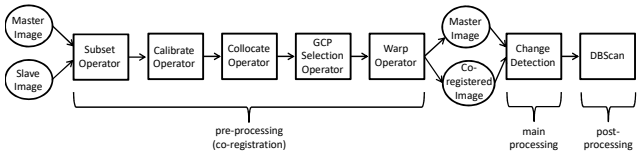

Figure 4: The workflow implemented by Change Detector.

interest - for Figure 1(b) - is October 26, 2017, while the reference date - for Figure 1(a) - is anything before June, 2017.

The downloaded Sentinel images are then stored to the Hadoop Distributed File System (**HDFS**). This is necessary for distributing parts of the images to all nodes in the cluster at hand in order to make them available to the parallel image processing code in a scalable and fault-tolerant way.

Finally, the **Change Detector** applies the workflow depicted in Figure 4, which implements in parallel the state-of-the-art *unsupervised* approach offered by ESA's SNAP Toolbox. Its goal is to compare the downloaded images in order to identify the changes in land cover or land use. This workflow consists of three stages: *(i)* Pre-processing uses *co-registration* [36] to ensure that the selected images have identical dimensions and correspond to the same geolocation. *(ii)* Main processing compares the individual pixels in the images to assess their difference. *(iii)* Post-processing clusters together the pixels with high likelihood of changes, forming broader areas with changes in a way that reduces false alarms, i.e., it excludes outliers caused by *noise*, which is either inherent in the satellite images or introduced by inaccuracies of previous steps.

In more detail, we call *master image* the one corresponding to the earliest date - Figure 1(a) in our example - and *slave image* the one corresponding to latest date - Figure 1(b). Typically, their dimensions and characteristics are quite different, because they were taken under different settings, such as the angle of the satellite. Therefore, pre-processing (co-registration) is indispensable for aligning the two images in such a way that each pair of corresponding pixels represents the same point on the Earth surface.

Given that individual satellite images typically cover a very large area on Earth, the **subset operator** crops the original satellite images to the borders of the user-defined area of interest. This operation curtails the running time to a significant extent, restricting the computational cost to the absolutely essential parts of satellite images. Its complexity is very low, requiring no parallelization.

The cropped images are given as input to the **collocate operator**, which resamples the pixels of the slave image into the geographical raster of the master. This operator requires accurate geopositioning information for both images in the form of *ground control points* (GCPs), i.e., markers for certain geographical positions within a geo-referenced image that are described by their geo-coordinates and by textual descriptions in the image meta-data.

Next, the **GCP selection operator** generates a set of uniformly spaced GCPs in the master image and computes their corresponding GCPs in the slave image. This is done through an iterative process: for each master GCP, the corresponding slave GCP is approximated based on their geo-coordinates. Using a predetermined window size, the areas surrounding each GCP are cross-correlated in order to adjust the slave GCP to a more accurate position. This procedure is repeated until the new slave GCP is located within acceptable limits, or a maximum number of iterations is carried out.

Based on the selected GCPs, the **warp operator** computes the *warp function*, which will be used for mapping the pixels of the slave image into the co-registered image. This is a linear function that is estimated by repeating the following process until convergence: a warp function is initially computed using the available master-slave GCP pairs. The resulting function is used to map the master GCPs to the slave image. Then, the residuals between the mapped master and the corresponding slave GCPs are computed along with the root mean square (RMS) and the standard deviation of all residuals. Next, the master-slave GCP pairs are filtered to eliminate those exceeding the mean RMS. Upon completion of this process, the remaining master-slave GCP pairs are filtered with a predetermined RMS threshold and the warp function is derived from the retained pairs.

Finally, the *co-registered image* is generated using the resulting warp function in combination with bilinear interpolation. This means that every point of the original slave image is projected to a point in the master image as the weighted sum of the warp projection of its four surrounding pixels.

Using the master and the co-registered image as input, the **change detection** algorithm computes the ratio of the corresponding pixels in the two images. The pixels exhibiting very large or very low ratios indicate candidate areas with changes.

Lastly, **DBScan** [13] is applied for post-processing the set of candidate areas with changes. DBScan groups together those pixels that are closely packed together (i.e., pixels with many nearby change indicators), while treating as outliers those pixels that lie alone in low-density regions, with their nearest neighbors located far away. The end result is a set of areas with changes in land cover or land use. In our example, DBScan produces the image in Figure 1(c), yielding the 7 yellow clusters that correspond to such areas.

Due to the high time complexity of all processes (except the Subset operator), they are massively parallelized in Apache Spark. Due to space limitations, we omit the parallelization details.

3.2 Event Detection Layer

To address Veracity, this layer attaches a set of recent events to every area with identified changes in land cover or land use, providing users with a possible explanation and verification of the detected changes. This functionality is offered by the five components at the top layer of GeoSensor's architecture in Figure 3.

The first component is the **News Crawler**, which scans at half-hour intervals specific social media sources and news agencies for the latest news items (posts and news articles, respectively). For the time being, these sources include most of the RSS feeds that are freely provided by Reuters in English[8] as well as several selected public accounts in Twitter[9], also in English. The crawler structure, though, is extensible, facilitating the integration of more information sources, or even the extension with other operation modes. For example, it has been used as a basic data collection infrastructure in a summarization application [16] and in the EU

[8] https://www.reuters.com/tools/rss
[9] https://twitter.com

project "NOMAD"[10]. In our running example, the News Crawler is responsible for gathering the news items in Figure 2(a).

All data gathered by the News Crawler are stored in the second component, namely **Apache Cassandra**[11]. We opted for this particular data management system, due to its capacity to store a large volume of information, while offering linear scalability and fault-tolerance (i.e., it provides high availability with no single point of failure). In fact, Cassandra is crafted for large-scale infrastructures like the BDI platform, offering robust support for clusters with multiple commodity servers. Besides, it is an open-source NoSQL database that is compatible with the SemaGrow component, which is used by the semantic layer for federated access to the details of individual news items or entire events (cf. Section 3.3).

The news items stored in Cassandra are periodically processed by the **Event Detector** module at half-hour intervals. They are grouped into real-world events by a modified version of NewSum[12] [16], a summarization algorithm providing commercial-grade performance. NewSum uses n-gram graphs [15] to model its textual input, a representation that has been shown to be effective in noisy settings in multiple genres (i.e., blogs, articles, microblogging and social media) [18, 32]. In addition, NewSum is robust to multilingual data, ranking among the top performers in multilingual, multi-document summarization tasks [17].

In more detail, Event Detector first builds a coarse-grained set of events. Pairs of news articles are compared with each other using their n-gram graphs representation and the corresponding graph-based textual similarity measures [15]. Appropriate thresholding is then applied to retain only the pairs with high similarity. Those pairs are then grouped into larger sets (pools) of news articles based on a transitivity analysis that forms clusters from connected components in the similarity graph. The *pools of news articles* with a very low support are discarded, whereas the remaining pools are considered as "real-world events". Due to its high time complexity, this process is parallelized in Apache Spark, as shown in Figure 5. The same procedure is applied independently to Twitter data, yielding a set of *tweet pools*. Each tweet pool is then compared with every pool of news articles. If their similarity exceeds a predefined threshold, the tweet pool is added to the pool of news articles. Then, every pool of news articles goes through a summarization process that builds its event description (e.g., title selection) and enriches it with relevant metadata, i.e., spatiotemporal information, named entities as well as image elements from its member documents. These metadata are extracted from its content directly, or with the help of RESTful-based tools and services, internal (Lookup Service and Entity Extractor) and external ones (PoolParty[13] and Flickr[14]).

In more detail, the **Lookup Service** associates the location names from news items with their actual geo-coordinates so that they can be joined with areas with detected changes in land cover or land use. The location names are identified and extracted from the text data in each news item using Apache openNLP[15]. In the example of Figure 2(a), the location of Kutupalong refugee camp

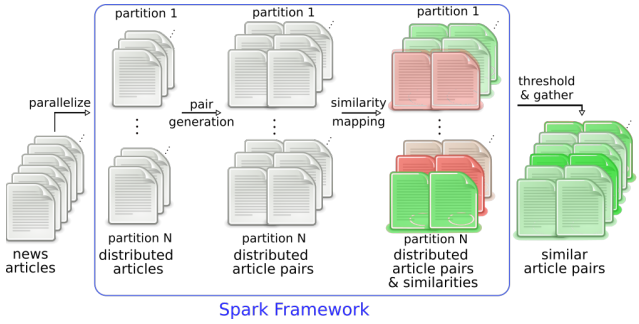

Figure 5: The Spark-based implementation of Event Detector.

(Ukhiya, Chittagong, Bangladesh) will be converted into the following geo-coordinates: POLYGON ((92.0455551147462 21.3476104736329, 92.2031173706055 21.3476104736329, 92.2031173706055 21.1280899047852, 92.0455551147462 21.1280899047852, 92.0455551147462 21.3476104736329)) – note that the output is in the form of the OGC[16] standard Well Known Text (WKT).

This conversion may seem a trivial task, given that there is little ambiguity in our example. In reality, though, location names typically suffer from high levels of noise. There are homonymous locations (e.g., London, UK and London, Ontario, Canada) as well as spelling mistakes (e.g., Landon), due to errors in the extraction process. To address both challenges, the Lookup Service poses every place name as a keyword query to an Apache Lucene[17] index that contains about 180,000 location names of administrative areas worldwide (GADM dataset[18]). Lucene's fuzzy query functionality deals with spelling mistakes, while homonymy is addressed by ranking the candidates in decreasing order of the ratio "string similarity/area". The WKT polygon coordinates corresponding to the top ranked location are finally returned as output.

Valuable metadata are also provided by the **Entity Extractor**, which enriches the event description with named entities that are extracted from their textual content, thus empowering a Semantic Web view of the produced information. This view allows for improved indexing and disambiguation of the main players in an event, based on the URIs mapped to each extracted entity. At the core of this functionality lies the **PoolParty Semantic Suite**, which constitutes a state-of-the-art thesaurus management tool that is based on Linked Data [35]. Specifically, a "Famous People" thesaurus was constructed, containing almost half a million entities of well-known actual and fictitious personalities, each grounded to a URI. Two RESTful APIs were implemented and hosted by PoolParty. Given an input text or a news item url, the first endpoint, called Extractor API, provides a list with entities deemed relevant to the supplied content. The entity URIs are stored in Cassandra, along with their corresponding thesaurus id. The second endpoint, called Metadata API, retrieves descriptive metadata related to the entity, whose URI is given as input. These procedures are illustrated in Figure 6.

The Entity Extractor also associates every detected event with publicly available images from **Flickr**. Using the Flickr search API,

[10] http://www.nomad-project.eu
[11] http://cassandra.apache.org
[12] https://github.com/scify
[13] https://www.poolparty.biz
[14] https://www.flickr.com
[15] https://opennlp.apache.org/

[16] The Open Geospatial Consortium - http://www.opengeospatial.org
[17] https://lucene.apache.org
[18] http://www.gadm.org

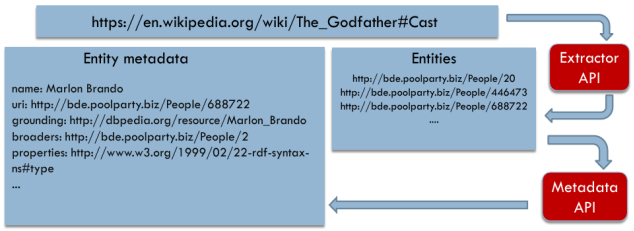

Figure 6: Entity extraction example, illustrating the Extractor and Metadata API.

it retrieves photographs geo-tagged within the geolocation(s) of each event that have been uploaded at a close enough date.

Finally, all event descriptions, including their metadata, are stored into Cassandra in the appropriate tables that distinguish them from individual news items. Duplicate events are discarded and Strabon is notified for the new entries (see below for details).

3.3 Semantic Layer

This layer constitutes GeoSensor's backbone, bringing the gap between the two orthogonal operations of change and event detection. This is achieved by the four components in the middle of Figure 3, which encapsulate state-of-the-art Semantic Web technologies.

The first component is **Geotriples** [24], a tool for transforming geospatial data from their original formats into RDF. In our case, it converts into RDF the descriptions of areas with changes in land cover or land use (from change detection) as well as the event summaries (from event detection). We selected GeoTriples, as it is an established system that supports a wide variety of data formats [23].

The output of Geotriples is stored into **Strabon** [22], a state-of-the-art open-source spatio-temporal triplestore that efficiently executes GeoSPARQL and stSPARQL queries. Strabon supports spatial datatypes, enabling the serialization of geometric objects in the OGC standards WKT and Geography Markup Language (GML). It has been implemented by extending the established RDF store Sesame (now called RDF4J[19]), using the spatially-enabled database PostGIS[20] as back-end so as to exploit its large variety of spatial functions and operators. Thorough experiments have demonstrated that Strabon is the most efficient spatio-temporal RDF store available today [6, 14].

The third component of this layer is **SemaGrow** [9], a query processing system that provides a single SPARQL endpoint for federating multiple remote SPARQL endpoints. It is also capable of transparently optimizing queries and dynamically integrating heterogeneous data models by applying the appropriate vocabulary transformations. To boost federated query execution, it employs vocabulary mapping techniques and a balanced query optimizer, considering instance statistics from the federated bases, where available. SemaGrow is highly efficient, consistently outperforming the state-of-the-art in federated query processing [9]. In our case, SemaGrow federates Cassandra and Strabon, offering a unified SPARQL endpoint for both of them to GeoSensor's user interface. In this way, GeoSensor gains in query performance (with respect

[19]http://rdf4j.org
[20]https://postgis.net

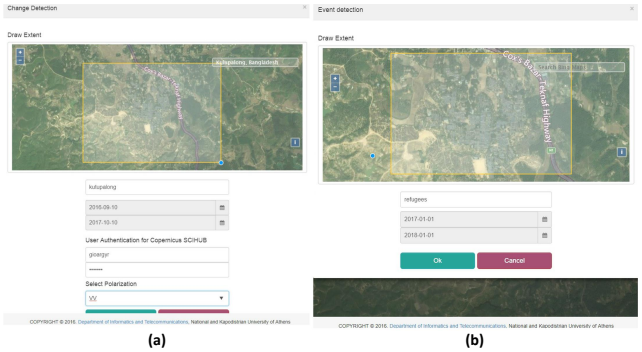

Figure 7: User criteria for triggering (a) Change Detection, and (b) Event Detection.

to other systems, e.g., FedX and SPLENDID) and has increased extensibility – in case new sources need to be added in the future.

GeoSensor's interface is offered by **Sextant** [29], a web-based application for exploring, interacting and visualizing time-evolving linked geospatial data. Sextant is also capable of creating, sharing, searching and collaboratively editing maps and of producing statistical charts out of statistically enhanced data sets. Even though it relies heavily on Semantic Web technologies, it offers an intuitive interface that allows both domain experts and lay users to exploit all available features. Being the entry point for GeoSensor, Sextant has been widely extended to cover all its requirements. Three are the new functionalities it offers:

(i) Core functionality. Sextant provides an intuitive interface for initiating the event and the change detection processes of GeoSensor. The window for launching change detection appears in Figure 7(a). The user selects an area of interest either by typing its name (with the help of auto-complete), or by highlighting it on the Earth Map. The credentials for Copernicus Open Access Hub are also required along with the reference and the target date. For event detection, Figure 7(b) depicts the window that prompts users to define three optional search criteria: an area of interest, a time window defined by two dates, or a keyword that pertains to events of interest. The last criterion can be a combination of location or entity names, or any other words that are likely to appear in an event title. Users can also search for events by setting as the area of interest one that appears in the results of change detection.

(ii) Authorization/authentication. To support history over each user's actions, Sextant implements a sign-up and login functionality. At its core, lies a database located in GeoSensor's server that holds all account information along with the encrypted passwords. To ensure security over the network, Sextant can be deployed using the HTTPS protocol. When GeoSensor first loads, the user is prompted to create a new account, or to log-in using an existing one. Three types of users that are supported: (a) The *administrators* have full access to all the supported functionality, including the history panel, and are responsible for accepting or declining sign-up requests by new users. (b) The *classified users* are the main users of the application and have full access to all the supported functionality, including the history panel. (c) The *unclassified users* are potential trial or occasional users that have limits in using the supported

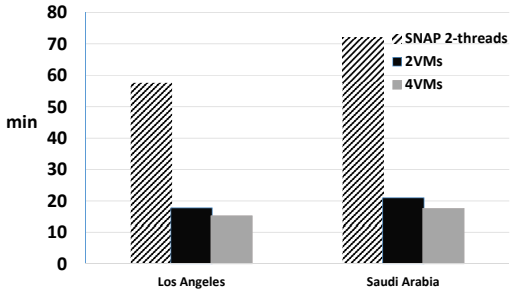

Figure 8: Execution times for parallelization approaches of the change detection workflow.

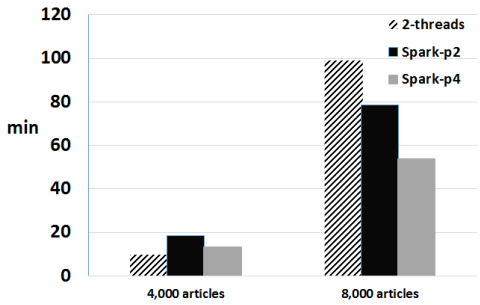

Figure 9: Execution times for parallelization approaches of the event detection workflow.

functionality: they lack a history panel, they cannot search for events using keywords, and their event detection searches return up to 5 events. They are also deprived of the "SMART" buttons that alternate change and event detection.

(iii) Live Twitter keyword search. To further clarify the map visualization with the latest raw information, overcoming the processing delay of the event detection layer, Sextant offers an emdeded Twitter keyword search function that supports all Twitter API filters, such as "#" or "@". Using up to five keywords in the search field, Sextant returns the relevant tweets in chronological order, with the most recent one appearing first. There is also a refresh button that fetches the latest results, if they are available. The results are presented using an infinite scroll technique that allows for quickly loading the tweets in the interface.

4 EXPERIMENTS

We now present a preliminary experimental evaluation of GeoSensor's main functionalities, namely the change and the event detection workflows. Note that our evaluation focuses on time efficiency, aiming to assess the response time of each workflow. In other words, effectiveness lies out of the scope of this evaluation, as GeoSensor employs unsupervised state-of-the-art methods for each operation.

4.1 Change detection

For change detection, we evaluate the time efficiency of two different approaches: *(i)* the Change Detector, which uses Apache Spark to parallelize the process depicted in Figure 4. *(ii)* the baseline approach, which corresponds to the multi-threaded implementation of the same workflow, as provided by ESA's SNAP Toolbox.

Data. As test data, we use two pairs of Sentinel-1A images. One comprising two images of Los Angeles, with file sizes of 508MB and 504MB, and one consisting two images of Saudi Arabia, with file sizes of 524MB and 526MB.

Experimental Setup. All experiments were performed on a server with Ubuntu 12.04, 132GB RAM and 4 AMD Opteron 6320 processors, each having 4 physical cores and 8 logical cores at 2.80GHz. For the Spark implementation, we created 4 virtual machines (VMs), each one comprising two cores and 20GB RAM. For each pair of images, we used 2 and 4 VMs. In each case, one VM was the master and the rest were used as slaves. The multi-threaded implementation of SNAP was run using 2 cores on the same server. For each method and configuration, we took 3 measurements of the execution time and report the average in Figure 8.

Time Efficiency & Scalability. As shown in Figure 8, the 2-VMs Spark implementation is three times faster that the multi-threaded one. This shows that the communication overhead of Spark is negligible in comparison to the processing time and does not affect the execution times. Furthermore, as we add more slave nodes to the Spark implementation, the execution times decrease consistently. We are working, though, on further improving this performance so as to achieve a linear speedup.

4.2 Event Detection

For event detection, we perform an empirical evaluation of the runtime performance of two approaches: *(i)* the Event Detector, which implements the Spark-based distributed similarity mapping pipeline illustrated in Figure 5, and *(ii)* the baseline approach, which parallelizes the same pipeline using the Java multi-threading library.

Data. We use the Reuters 21K news articles dataset[21]. Preprocessing discards everything but the clean text, title and publication date information, storing all data in Cassandra.

Experimental Setup. We run a set of experiments for two different input sizes, namely for input batches of $4,000$ and $8,000$ articles to be clustered into events. These sizes correspond to approximately 16 and 64 million unique article pairs. For each batch size, we apply the baseline approach using 2 threads, while for the Event Detector we vary the number of Spark partitions $p \in \{2, 4\}$. For each configuration, we perform 5 experiments and compute the mean average execution time. We run all experiments on a single, 8-core 2.6 GHz Ubuntu 14.04 virtual machine with 32 GB of memory. For data storage, we use a Cassandra 2.2.4 docker container.

Time Efficiency & Scalability. Figure 9 depicts the execution time results per configuration. For the Event Detector, we observe that the runtime drops significantly as we increase the number of Spark partitions, i.e., the number of jobs run in parallel. Yet, the baseline approach is significantly slower only for the largest batch size. The reason is that for a small number of small texts, as in Reuters 21K, Spark's parallelization overhead is higher than the speedup it achieves. We are working on improving Event Detector's implementation so that its performance is competitive even for small workloads.

[21] http://www.daviddlewis.com/resources/testcollections/reuters21578/

5 CONCLUSIONS

We presented GeoSensor, the first open-source system that applies Semantic Web technologies to a combination of remote and social sensing. The RDF data model plays a crucial role, as it offers two major advantages compared to traditional, semantic-free approaches. First, it allows for effectively dealing with Variety, seamlessly combining all data sources to produce meaningful analysis. It also facilitates the use of ontologies together with reasoning techniques so as to derive new facts that are not explicitly expressed in the available data. The second advantage comes from the power of linked open data and semantics. Transforming GeoSensor's data into RDF allows for effortlessly interlinking it with other data sources and for discovering hidden links between entities that assist in the data analysis. This linking process provides richer data and allows us to build fully automated workflows using machine learning algorithms, based on the power of semantics.

Moreover, GeoSensor can be easily deployed in any cluster. *All its components are provided as Docker images that are publicly available through the BDE repository*[22]. As a result, the whole system can be launched through a single docker-compose file, running the individual components as Docker containers within Docker Swarm[23]. GeoSensor also offers an intuitive user interface that is suitable for both expert and lay users, despite the rich information it processes. In fact, GeoSensor provides a *hands-off functionality* in the sense that all its operations are fully automatic, requiring no specialized input or domain knowledge from its users. GeoSensor thus makes a big step forward in the exploration and visualization of big data in the context of remote sensing. Our preliminary experimental study also demonstrated the high time efficiency of our system.

In the future, we plan to test GeoSensor in a rigorous, operational scenario, where decision makers require fast and easy-to-use tools to support their decision.

Acknowledgements. This work has been supported by the projects "LEO" and "BigDataEurope", which have been funded by EU FP7 and Horizon2020 programmes under grant agreements No. 611141 and 644564, respectively. It has also been supported by the program of Industrial Scholarships of the Stavros Niarchos Foundation (https://www.snf.org/).

REFERENCES

[1] Hamed Abdelhaq, Christian Sengstock, and Michael Gertz. 2013. Eventweet: Online localized event detection from twitter. *PVLDB* 6, 12 (2013), 1326–1329.
[2] Charu C Aggarwal and Karthik Subbian. 2012. Event detection in social streams. In *SDM*. 624–635.
[3] Fotis Aisopos, George Papadakis, Konstantinos Tserpes, and Theodora A. Varvarigou. 2012. Content vs. context for sentiment analysis: a comparative analysis over microblogs. In *ACM Conference on Hypertext and Social Media*. 187–196.
[4] Farzindar Atefeh and Wael Khreich. 2015. A survey of techniques for event detection in twitter. *Computational Intelligence* 31, 1 (2015), 132–164.
[5] Sören Auer, Simon Scerri, and Aad Versteden et. al. 2017. The BigDataEurope Platform — Supporting the Variety Dimension of Big Data. In *ICWE*. 41–59.
[6] Konstantina Bereta, Panayiotis Smeros, and Manolis Koubarakis. 2013. Representation and Querying of Valid Time of Triples in Linked Geospatial Data. In *ESWC*. 259–274.
[7] Francesca Bovolo and Lorenzo Bruzzone. 2015. The time variable in data fusion: A change detection perspective. *IEEE Geosc. Rem. Sensing Mag.* 3, 3 (2015), 8–26.
[8] Grégoire Burel, Hassan Saif, and Harith Alani. 2017. Semantic Wide and Deep Learning for Detecting Crisis-Information Categories on Social Media. In *ISWC*. 138–155.
[9] Angelos Charalambidis, Antonis Troumpoukis, and Stasinos Konstantopoulos. 2015. Semagrow: Optimizing federated SPARQL queries. In *SEMANTiCS*. 121–128.
[10] Ping Chen, Soo Chin Liew, and Leong Keong Kwoh. 2017. Mangrove mapping and change detection using satellite imagery. In *IGARSS*. 5717–5720.
[11] Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *ACL*. 167–176.
[12] Daniela Espinoza-Molina, Reza Bahmanyar, Ricardo Díaz-Delgado, Javier Bustamante, and Mihai Datcu. 2017. Land-cover change detection using local feature descriptors extracted from spectral indices. In *IGARSS*. 1938–1941.
[13] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *KDD*. 226–231.
[14] George Garbis, Kostis Kyzirakos, and Manolis Koubarakis. 2013. Geographica: A Benchmark for Geospatial RDF Stores (Long Version). In *ISWC*. 343–359.
[15] George Giannakopoulos, Vangelis Karkaletsis, George A. Vouros, and Panagiotis Stamatopoulos. 2008. Summarization system evaluation revisited: N-gram graphs. *TSLP* 5, 3 (2008), 5:1–5:39.
[16] George Giannakopoulos, George Kiomourtzis, and Vangelis Karkaletsis. 2014. NewSum: N-Gram Graph-Based Summarization in the Real World. In *Innovative Document Summarization Techniques: Revolutionizing Knowledge Understanding*.
[17] George Giannakopoulos, Jeff Kubina, John Conroy, Josef Steinberger, and Benoit et al. Favre. 2015. Multiling 2015: multilingual summarization of single and multi-documents, on-line fora, and call-center conversations. In *SIGDIAL*. 270–274.
[18] George Giannakopoulos, Petra Mavridi, Georgios Paliouras, George Papadakis, and Konstantinos Tserpes. 2012. Representation models for text classification: a comparative analysis over three web document types. In *WIMS*. 1–12.
[19] Maoguo Gong, Tao Zhan, Puzhao Zhang, and Qiguang Miao. 2017. Superpixel-Based Difference Representation Learning for Change Detection in Multispectral Remote Sensing Images. *IEEE Geosci. Remote Sensing* 55, 5 (2017), 2658–2673.
[20] Salman Hameed Khan, Xuming He, Fatih Porikli, and Mohammed Bennamoun. 2017. Forest Change Detection in Incomplete Satellite Images With Deep Neural Networks. *IEEE Trans. Geoscience and Remote Sensing* 55, 9 (2017), 5407–5423.
[21] Shamanth Kumar, Huan Liu, Sameep Mehta, and L Venkata Subramaniam. 2014. From tweets to events: Exploring a scalable solution for Twitter streams. *arXiv preprint:1405.1392* (2014).
[22] Kostis Kyzirakos, Manos Karpathiotakis, and Manolis Koubarakis. 2012. Strabon: A Semantic Geospatial DBMS. In *ISWC*. 295–311.
[23] Kostis Kyzirakos, Dimitrianos Savva, Ioannis Vlachopoulos, Alexandros Vasileiou, Nikolaos Karalis, Manolis Koubarakis, and Stefan Manegold. 2018. GeoTriples: Transforming Geospatial Data into RDF Graphs Using R2RML and RML Mappings. *Web Semantics: Science, Services and Agents on the World Wide Web* (2018).
[24] Kostis Kyzirakos, Ioannis Vlachopoulos, Dimitrianos Savva, Stefan Manegold, and Manolis Koubarakis. 2014. GeoTriples: a Tool for Publishing Geospatial Data as RDF Graphs Using R2RML Mappings. In *ISWC*. 393–396.
[25] Dengsheng Lu, P Mausel, E Brondizio, and Emilio Moran. 2004. Change detection techniques. *International journal of remote sensing* 25, 12 (2004), 2365–2401.
[26] Juha Makkonen, Helena Ahonen-Myka, and Marko Salmenkivi. 2002. Applying semantic classes in event detection and tracking. In *ICON 2002*. 175–183.
[27] Juha Makkonen, Helena Ahonen-Myka, and Marko Salmenkivi. 2004. Simple semantics in topic detection and tracking. *Inf. Retr.* 7, 3-4 (2004), 347–368.
[28] Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *ACL*. 365–371.
[29] Charalampos Nikolaou, Kallirroi Dogani, Konstantina Bereta, George Garbis, Manos Karpathiotakis, Kostis Kyzirakos, and Manolis Koubarakis. 2015. Sextant: Visualizing time-evolving linked geospatial data. *J. Web Sem.* 35 (2015), 35–52.
[30] Ozer Ozdikis, Pinar Senkul, and Halit Oguztuzun. 2012. Semantic expansion of tweet contents for enhanced event detection in twitter. In *ASONAM*. 20–24.
[31] Nikolaos Panagiotou, Ioannis Katakis, and Dimitrios Gunopulos. 2016. Detecting events in online social networks: Definitions, trends and challenges. In *Solving Large Scale Learning Tasks. Challenges and Algorithms*. 42–84.
[32] George Papadakis, George Giannakopoulos, and Georgios Paliouras. 2016. Graph vs. bag representation models for the topic classification of web documents. *WWW* 19, 5 (2016), 887–920.
[33] Richard J. Radke, Srinivas Andra, Omar Al-Kofahi, and Badrinath Roysam. 2005. Image change detection algorithms: a systematic survey. *IEEE Trans. Image Process.* 14, 3 (2005), 294–307.
[34] Jagan Sankaranarayanan, Hanan Samet, Benjamin E Teitler, Michael D Lieberman, and Jon Sperling. 2009. Twitterstand: news in tweets. In *SIGSPATIAL*. 42–51.
[35] Thomas Schandl and Andreas Blumauer. 2010. PoolParty: SKOS Thesaurus Management Utilizing Linked Data. In *ESWC*. 421–425.
[36] Nestor Yague-Martinez, Francesco De Zan, and Pau Prats-Iraola. 2017. Coregistration of Interferometric Stacks of Sentinel-1 TOPS Data. *IEEE Geosci. Remote Sensing* 14, 7 (2017), 1002–1006.
[37] Yiming Yang, Tom Pierce, and Jaime Carbonell. 1998. A study of retrospective and on-line event detection. In *SIGIR*. 28–36.

[22] https://github.com/big-data-europe/pilot-sc7-cycle3
[23] https://docs.docker.com/engine/swarm