Social Networking Trends and Dynamics Detection via a Cloud-based Framework Design

Athena Vakali
Department of Informatics
Aristotle University
54124 Thessaloniki, Greece
+30 2310998415
avakali@csd.auth.gr

Maria Giatsoglou
Department of Informatics
Aristotle University
54124 Thessaloniki, Greece
+30 2310998236
mgiatsog@csd.auth.gr

Stefanos Antaris
Department of Informatics
Aristotle University
54124 Thessaloniki, Greece
+30 2310991863
santaris@csd.auth.gr

ABSTRACT
Social networking media generate huge content streams, which leverage, both academia and developers efforts in providing unbiased, powerful indications of users' opinion and interests. Here, we present Cloud4Trends, a framework for collecting and analyzing user generated content through microblogging and blogging applications, both separately and jointly, focused on certain geographical areas, towards the identification of the most significant topics using trend analysis techniques. The cloud computing paradigm appears to offer a significant benefit in order to make such applications viable considering that the massive data sizes produced daily impose the need of a scalable and powerful infrastructure. Cloud4Trends constitutes an efficient Cloud-based approach in order to solve the online trend tracking problem based on Web 2.0 sources. A detailed system architecture model is also proposed, which is largely based on a set of service modules developed within the VENUS-C research project to facilitate the deployment of research applications on Cloud infrastructures.

Categories and Subject Descriptors
H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – *clustering*; H.3.4 [**Information Storage and Retrieval**]: Systems and Software – *distributed systems*; H.3.5 [**Information Storage and Retrieval**]: Online Information Services – *Web-based services*; I.7.5 [**Document and Text Processing**]: Documents Capture – *Document analysis*

General Terms
Design, Algorithms, Performance, Experimentation

Keywords
Social networks, microblogs and blogosphere dynamics, social Web data clustering, cloud service deployment.

1. INTRODUCTION
Social media applications have emerged as powerful means of communication for people seeking to share and exchange information on a wide variety of topics. These topics range from popular, widely known ones (e.g., a concert by a popular music band) to smaller scale, local (e.g., a local social gathering, a protest, or an accident) and their popularity fluctuates with time. User-contributed messages posted on social media sites can typically reflect these topics in their actual dimension and for this reason the content of social media sites is particularly useful for real-time trends' identification. Detecting topic-specific trends is surely of significant interest primarily due to the fact that trends:

i. can be used to detect emergent or suspicious behaviour in the network;

ii. can be viewed as a reflection of societal concerns or even as a consensus of collective decision making.

Microblogging applications have become key actors in social media and they have become mostly popular due to their abilities for broadcasting information in a real-time fashion. In Twitter, for example, tweets are generated by users in massive sizes, currently reaching 200 million per day[1]. Twitter itself currently employs a proprietary algorithm for displaying "*trending*" topics, consisting of terms or phrases whose usage exhibits trending behaviour. While Twitter's trending topics sometimes reflect current events (e.g., "world cup"), they often include keywords for popular conversation topics (e.g., "#bieberfever", "getting ready") without distinguishing the different types of content. Raw information from Twitter has been exploited in research for predicting the revenue of forthcoming films and stock prices, as well as for the real-time identification of earthquakes (in Japan) and the analysis of users' reaction towards certain events (political debates). It is thus currently widely acknowledged that microblogging (as practiced via tweets) forms a valuable source of latent information about the dynamics involved in the public's opinions, views, and moods. This is further justified by the fact that such applications capture the momentum and the pulse of a large public's scale (Twitter only has more than 300 million registered users [13]).

At the same time blogging platforms have been established as a popular form of communication on the Web. The blogosphere is a rich information source representing the dynamics and the "voice of the public" which is useful for extracting and mining public opinions on certain topics or events. Opposed to typical online textual information applications, blogs are mainly characterized by their "social pulsing" nature since in blogging:

i. information circulated is primarily opinion-oriented, reflecting the author's freely expressed point of view;

ii. there are threads of several articles covering a wide range of topics.

Copyright is held by the International World Wide Web Conference Committee (IW3C2). Distribution of these papers is limited to classroom use, and personal use by others.
WWW 2012 Companion, April 16–20, 2012, Lyon, France.
ACM 978-1-4503-1230-1/12/04.

[1] http://blog.twitter.com/2011/06/200-million-tweets-per-day.html

Thus, microblogging/blogging activities can serve as major social dynamics barometers since their parallel information flows embed valuable and often hidden information about trending users' interests and opinions. In such a context, it is also a fact that microblogging and blogging share common data "bridges" since users' interests are influenced by both such social media. E.g. users' tendency to add hyperlinks in their tweets is a common practice, as it appears from the around 25% rate of all created tweets to contain at least one hyperlink[2] (as of September 2010).

This position paper places emphasis on designing a real-time data analysis system scenario with capabilities for localized trending topics detection. Trends dynamics are harvested and analyzed over user-contributed content from both microblogging (Twitter) and blogosphere activities through a text clustering approach. To support such a system, which is on high processing and data management demands, a Cloud-based deployment has been designed and is proposed here.

The proposed framework's contribution is summarized in its following objectives:

- *dealing with the large scale reality in the Web 2.0 scene* (with huge and rapidly evolving data) by developing methods for handling efficiently such data in real time, which can be used in real world application settings;
- *supporting the analysis of text data from different web sources* which may be generated at various rates in a unified way;
- *proposing a methodology for unsupervised detection of local trends* by combining content from different web sources to enrich the detected trends' context;
- *capturing* the shaping and evolution of users' interests in time depending on the users' broader geographical location and the type of data source;
- *designing a Cloud-based data processing methodology* to support a streaming web data clustering scenario under a parallelized computation setting.

The rest of the paper is structured as follows. Section 2 reviews the current state regarding trend detection approaches that leverage microblogging and blogging data, discussing their limitations. Section 3 presents an incremental social data analysis and mining methodology for detecting trends, and discusses its potential and advantages over existing approaches. In Section 4, we discuss the limitations of deploying such an application on a single computer and propose an architecture for porting and operating such an application on a Cloud infrastructure. Finally, Section 5 presents the implementation details of an early deployment of the proposed trend detection application, Cloud4Trends, on a Cloud infrastructure and concludes the paper.

2. MICROBLOGGING AND BLOGGING TREND DETECTION: CURRENT STATUS

The massive sizes of user generated content (UGC) in microblogging and blogging applications, sets the potential to proceed with its consideration as a raw data source for real-time localized trend detector and "public's pulse" monitoring tool. Such trend detection strongly dictates the need for efficient scalable and/or summarizing methodologies. Current clustering approaches (based primarily on Twitter) focus on either finding: (i) clusters of users densely associated via follower or message (@message) links, or (ii) groups of tweets using text mining techniques, such as exploiting common word co-occurrences.

A typical approach to trend analysis involves tracking users' interests in different keywords across time. In this context, traditional statistical methods based on the total number of keyword occurrences are applied to identify temporal trends. Such methods provide a general indication about how popular a keyword is (or a sequence of keywords, e.g. bigrams) in some timeframes but lack in identifying the different topics/interests. Temporal trend analysis based on keyword frequency has appeared in several commercial blog and Web search engines such as: Google Hot Trends[3] and BlogPulse[4]. Google Hot Trends shows regularly trending search topics, referred to as *Hot Searches*, which are phrases whose popularity is statistically calculated by their frequency of appearance in what people are searching for on Google search engine at the current date. Although Google Hot Trends analyzes millions of web searches in order to identify trends, it does not emphasize on the social data analysis which is a collective source of intelligence that can be used to obtain opinions, ideas, facts and sentiments. Moreover, such trends are available only for the US. BlogPulse is an online service that discovers trends from blogs on a daily basis. It follows a composite approach [1] that combines a number of statistical techniques for finding trending phrases based on their frequency of appearance in relation to other phrases, and that also exhibit a "bursty" trend line. Then, a merging clustering technique is applied on the trending phrases with the resulting clusters representing topics that are characterized by phrases that frequently co-occur in blogs. Finally, there are many online services that present statistically identified term frequency trends focused on Twitter such as e.g. Trendistic[5]. Trendistic provides "hot" terms or phrases (regardless of whether they are semantically significant) that appear in tweets including their start and peak time, the duration of their popularity, and their average and peak popularity. Their popularity is represented as the percentage of tweets including the term or phrase at a given period. Twitter itself also presents local trends[6] (for some locations) as keywords that are popular at the current time and at a particular city. However, its analysis seems to be based solely on term frequency, without providing any additional context for the trending keywords. Tweets that include a given keyword are available to users, but without taking in account the location filter.

Clustering has been widely applied on content generated in web social media to uncover latent associations, while recently the feature of time [7] and the temporal evolution of clusters [9] have been researched. Some web social data clustering approaches have been applied for trend detection, as in [2] where associations existing between blogs in relation to references from one blog to the other are modeled with a graph structure, and then a hierarchical graph clustering algorithm is applied. Each resulting cluster includes blogs densely connected via trackback links, while a number of trending topics is identified for each cluster from the terms contained in its blog members via a TF-IDF-based [10] method. This approach operates on a static dataset, as it is not tailored for real-time online operation. Also, the clustering

[2] http://techcrunch.com/2010/09/14/twitter-seeing-90-million-tweets-per-day/

[3] http://www.google.com/trends

[4] http://www.blogpulse.com/trends.html

[5] http://trendistic.com/

[6] https://support.twitter.com/articles/101125

approach followed does not allow grouping similar posts (in terms of content) that are not connected through a reference.

In the same context, the BlogScope infrastructure [3] collects information from the Blogosphere, news sources, social networks, and other online forums automatically, and performs spatiotemporal burst detection emphasizing on the algorithm's scalability. This approach is focused on event discovery based on a given query and is motivated by: i) the fact that the volume of posts that relate to an event rapidly increases (information burst) when the event takes places, and ii) the fact that such events (and thus information bursts) usually have a temporal and geographical scope. Bursts are identified by determining the geographic locations where documents related to the query keywords exhibit a surge, using spatiotemporal statistics and heuristics. The part that is more related to our work is the method followed for describing the identified bursts that is based on the fact that when a query term is bursty for a time interval, then the keywords that occur frequently together with this term will probably exhibit a burst themselves over the same time period. After identifying such keywords, the method calculates their coexistence frequency with the query terms and selects the ones with the highest frequency.

NewsStand [4] is an online news aggregator service that monitors RSS feeds from several online news sources and retrieves articles at the time of their publication. Afterwards, it extracts geographic content that appears in the articles for detecting their geographical focus based on the GeoNames Ontology[7]. The collected articles are grouped into news story clusters with an online clustering technique based on textual content and each story is attributed a geographic context depending on its members geographical focus. The clustering algorithm followed in NewsStand is also followed in TwitterStand [5], a work by the same authors that focuses on news detection from tweets. TwiterStand collects data online from the Twitter's GardenHose service (that provides a sample of Twitter's public timeline), and some handpicked (Seeders), or identified by special algorithms during the service's running time, users that are known to publish tweets that relate to news. Manual selection of users who will contribute the news content is used in an effort to alleviate the noisy nature of Twitter. Such a selection though, bears the danger of resulting in biased news, and to deal with noise, TwitterStand also filters out tweets that are unrelated to news via a classification method based on the Naïve Bayes Classifier. After that, the tweets are clustered with an online method that holds many similarities to the one followed in Cloud4Trends application (which is presented here). In particular, the TwitterStand's algorithm extracts TF-IDF feature vectors for the tweets and the clusters and performs clustering based on their similarity, while it also incorporates the temporal dimension in the clustering process in the same way as Cloud4Trends does.

TwitterMonitor [6] is another significant approach towards online trend detection over Twitter, following an approach similar to BlogScope [3]. The first step is the detection of some bursty keywords based on their appearance frequency with an online burst detection algorithm that operates on streaming data. Next, for each bursty keyword its recent history of tweets is retrieved and keywords are clustered based on their co-occurrences in the retrieved tweets into a "trend". Then, TwitterMonitor enriches the description of a given trend by applying a context extraction algorithm (such as PCA) on the recent trend history and finds strongly correlated keywords. This approach does not seem to exploit additional referenced content, since it occasionally involves some news sites in the trend description.

3. A MULTI-ATTRIBUTE TRENDS AND DYNAMICS DETECTION DESIGN

Here, we raise the fact that clustering tweets is more useful once aiming at public trend and dynamics detection in a real-time fashion. It should be highlighted here that brief information streams are the very nature of microblogging (e.g. tweets are up to 140 characters), thus users express opinions (or post pieces of information) in a very concrete and sharp way. Allowing users to include hyperlinks to other sites opens microblogging window to other types of information (webpages, articles, videos, etc). In practice, the framework presented here, Cloud4Trends, enables the online identification of trends dynamics, using Twitter and the Blogosphere as data sources. To the authors knowledge, and according to the state-of-the-art methods (discussed in the previous section), this is among the very initial efforts which combines clustering and analysis on both tweets and blog posts towards trend identification.

The proposed approach applies text clustering in an incremental fashion for detecting and maintaining a set of dynamic clusters based on the assumption the analysis at a "document" instead of at a "term" level is more promising for providing trending topics that have a meaningful context for users. Our clustering approach is inspired by and extends the earlier work in TwitterStand [5], since here we focus on expanding the original tweet content by additional information as well focusing on *trend detection* rather than *news detection*. Moreover, in our approach clusters which are active at a given time constitute *active topics* which are of users' interest and can be ranked based on their observed activity for indentifying the most popular (trending). By dynamically observing the clusters' updating rate, we can identify trends at their peak and detect the topics that are no more trending, instead of applying a fixed-threshold based method that sets as inactive clusters after a predefined period of time, such as in TwitterStand. Moreover, compared to the aforementioned system, the proposed application separately collects and clusters tweets that pertain to a desired geographical area, rather than examining the geographical scope of the resulting clusters as a post-analysis process.

Our approach builds on the idea of extending a microblogging tuple (as captured in a tweet) by replacing hyperlinks appearing in tweets' text. This replacement is made such that this process will "enlarge" the initial tweet content by following the included hyperlink, distinguishing between two cases: i) using the web pages' title and content to replace the hyperlink once the latter leads to a blog post, or ii) using solely its title for any other type of web page. We have chosen to promote hyperlinks leading to blogs, as we want to leverage the opinion-directed information published by individuals in a Web 2.0 platform to obtain a more substantial opinion-oriented tweet. To our knowledge this is the first work that performs simultaneous analysis of trending topics in: i) tweets, ii) blogs posts, and iii) tweets with extended content (referred to from now on as *extended tweets*), focused on certain geographic areas and attempts to analyze and evaluate differences in the way information is spread in the different media, depending on their type and orientation.

Thus, we propose that microblogging analysis is performed on a real-time streaming fashion in order to capture constantly changing trending users' interests. Moreover, the proposed analysis is further extended by the following facts:

[7] http://www.geonames.org/ontology/documentation.html

- it exploits associations based on the broadcasting time, alleviating gaps in earlier efforts such as in [1], which employs a clustering method after identifying a set of trending phrases and focuses only on the latter, in an offline fashion;
- it deals also with the respective user's physical location (exploiting the tweet geo-location feature).

This type of joint multi-feature analysis is expected to produce more fine-grained high-quality clusters of tweets which will correspond to actual topics that are popular at a given location and time period. It is also expected to alleviate the generally acknowledged problem of noisy microblogging data (i.e. data that contain uninteresting or meaningless information, as the joint consideration of location (even at filtering level above the clustering operation) and time will generally improve the clustering quality and contribute to filtering out noisy tweets.

The general trend detection task that the proposed approach addresses is outlined below.

Trend Detection Problem Formulation: Given a time-ordered stream of users' posts P_t, t = [1,… ∞), arriving in real-time (tweets) or at a given time granularity (blog posts), identify topics and associated posts that are popular ("*trending*") at any given time, and monitor their evolution across time in terms of popularity.

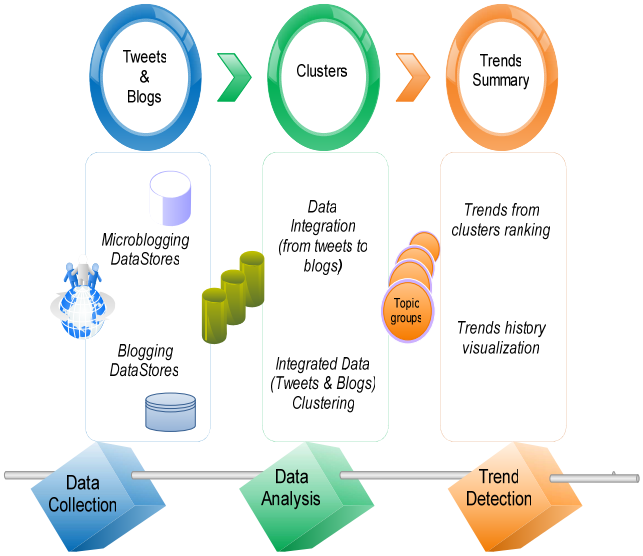

Figure 1. Microblogging and Blogging Trend Detection Outline

The problem outlined above distinguishes between: i) streams of content pushed to the application as soon as it is generated (e.g. such as tweets due to the availability of a streaming API), and ii) new content which needs to be pulled by the web source in which it is generated at a given rate (e.g. such as Google Blogger posts due to the availability of a REST API). In the latter case, the new data's pulling rate determines how much the identified trends correspond to the real-time users trending topics.

The proposed application is outlined in Fig. 1 and it actually involves a 3-tier design that deals with the: i) collection of data in a streaming manner from Twitter as well as from a pool of selected blogs focused on a number of geographic areas, ii) application of an online clustering technique on the data to detect recent trending topics, and iii) refinement and ranking of clusters such that trends are detected and visualized. These three tiers are summarized in the next subsections.

3.1 The Data Collection Tier

The Data collection tier involves special online data aggregators for collecting recently published content from Twitter and the Blogosphere. The content corresponds to some specific geographic area (such as a city level), leveraging the Twitter Streaming API[8] and Google Blogger API[9] (other possibilities in blogging and microblogging platforms can also be considered). While the first API provides a continuous stream of recently generated posts the second one is based on REST requests. To this end, based on a collection of identifiers of blogs owned by Blogger users who have declared that they reside within the monitored geographic area, we use the API for requesting new posts for each blog at a fixed time interval (e.g. daily, which is reasonable since we do not expect blogs to be updated as frequently as content in Twitter).

Special parsers have been developed for extracting hyperlinks from tweets, with a separate thread undertaking the task of retrieving the associated content from the web page they lead to (either simply its title or also additional information if it leads to a blog post). In order to identify clusters in both tweets, blog posts, and extended tweets, three separate representation models have been built for each type of content and for the analysis scope. The tweet data model includes the attributes of: unique identifier, text, (hash-)tags, timestamp, while the blog post data model has an additional "title" attribute. The extended tweet model simply incorporates one (or more) blog post entities in a given tweet entity. In order to verify whether the incorporation of the referenced blog posts actually leads to more meaningful trending clusters, all tweets need to be represented in both the simple and the extended tweet model (if any references)[10] leading to two different evolving datasets.

3.2 The Data Analysis and Processing Tier

The retrieved posts (either tweets, blog posts, or extended tweets) are processed in order to produce clusters which contain posts pertaining to the same topic. Data are filtered to remove low quality content with typical approaches including filtering out tweets/blog posts with very few terms, etc. Text sanitization techniques are applied on the resources' text to filter out common words (defined in a stop word lists) and to perform stemming. Next, resources, regardless of their type, are represented with a common model that includes: a unique identifier, a TF-IDF-based key-value map, a timestamp, and the resource's type (tweet, blog post, or extended tweet). For a given resource the key-value map structure includes as keys all the resource's unique terms, taken from the initial data model's *text*, *tags* and *title* (for blogs only) attributes. Using the Lucene Search Engine library[11] separate indexes are kept for each resource type and for each attribute. Though these indexes, TF-IDF key-value maps are obtained for each attribute. In order to represent the resource's textual content with a single attribute, and taking into account that a given key may exist in more than one attributes while usually *tags* and *title* attributes are more significant of the resource's content compared

[8] https://dev.twitter.com/docs/streaming-api
[9] http://code.google.com/apis/blogger/
[10] Obviously, tweets with no references can also be represented in the extended tweet model with zero encapsulated blog post entities.
[11] http://lucene.apache.org/core/

to the main text, the TF-IDF values for each key are combined in a single value under an appropriate weighting scheme that assigns more weight to terms in titles and tags. E.g. a given term T in a blog post resource R will be assigned a *score* value as follows:

$$SC(T,R) = w_{text} * TF_{text}(T,R) * IDF_{text}(T) + w_{title} * TF_{title}(T,R) * IDF_{title}(T) + w_{tags} * TF_{tags}(T,R) * IDF_{tags}(T)$$

In the equation above, w_i stand for the weight for attribute i, which is assigned a value in the interval [0,1]. Moreover, there is the possibility of zero TF-IDF values for the attributes of R in which term T does not exist.

Having a common resource representation, the data processing tier implements the resources clustering process addressing the following problem.

***Web Data Streams Clustering Problem Formulation*:** Given a time-point t, a new resource R_t created at t, and a set of n clusters C_i, [i= *1…n*] that are *active* at t, assign R_t to the cluster C_k for which the similarity $Sim(R_t,C_i)$ is maximized. If $Sim(R_t,C_i)$ < *sim_threshold* for each i in [1,n], then start a new cluster with R_t.

From the previous problem formulation it is evident that in order to perform the new resource's assignment to a cluster, a suitable representation format for the clusters is needed, as well as the selection of a similarity function. In general the framework is flexible allowing the use of any similarity function that can be applied on a given resource and cluster representation. In Cloud4trends the cluster representation model includes the following attributes: a unique identifier, a *mean* key-value map, a *mean* timestamp, and a list of its members. The mean *mean* key-value map and *mean* timestamp constitute the cluster's centroid since the first includes as keys the union of the terms included in all clusters' members and as values the average of the member's score values for each given term, while the second is the average of the timestamps of the cluster's members. Regarding the similarity function, we have selected a variation of the cosine similarity function, which is in general widely used for vector-wise similarity calculations. This function takes also the time aspect into account for the clustering process, as the timestamps are also included in the similarity calculation to bring resources closer to the clusters which on average contain members with similar timestamps, via a *Gaussian attenuator*. In particular, Cloud4Trends uses the following similarity measure (as it appeared in [5]), where *Map* and *Time* are the corresponding attributes of a (*mean* key-value map) key-value map and (*mean* timestamp) timestamp for the (cluster) resource.

$$\mathrm{Sim}(R_t, C_i) = \mathrm{CosineSim}(R_t.Map, C_i.Map) * e^{\frac{-(R_t.Time - C_i.Time)^2}{2\sigma^2}}$$

It is worth highlighting here is that the selection of the parameter's *sim_threshold* value should be wisely taken, because it affects significantly the diversity of the clusters in terms of topic, as well as the new clusters' generation rate, since a resource initiates a new cluster when all calculated similarities are below the *sim_threshold*.

3.3 The Trend Detection and Visualization Tier

This tier builds on the basis of the outcome of the data processing tier which produces three sets of clusters for the tweets', blog posts', and extended tweets' datasets. A given cluster can be characterized as *active* or *inactive* based on whether it is corresponds to topics that are popular at the given time, or it corresponds to topics that are no longer considered as trending. Clusters update rates are monitored to determine when a cluster should be made inactive due to limited activity. To this end, additional information is maintained for each cluster: *the evolution of the temporal distance between the timestamps of the last two resources assigned to the given cluster*. By taking the moving average of the aforementioned parameter to smoothen its evolution, we can identify periods of time when the cluster is increasingly rising in popularity due to users' intense activity, when it is at its peak, as well as when it should be made inactive due to a steady rise in the parameter's value. It should be mentioned here that in order to improve clusters' quality very small clusters (very few members) are considered as noise and thus are eliminated.

Active clusters are considered as representative of topics that concern web users at a given times, however, in order to identify the actual trends, clusters should be ranked in terms of an activity measure. To this end, for each type of content (and for a given monitored location) Cloud4Trends retrieves the *active* clusters and ranks them based on: i) their members' number, and ii) their mean timestamp, under the assumption that the "hottest" topics are those that are referred to in many resources, and that are additionally being created on average close to the current time.

The topics that characterize each cluster are identified as the terms with the highest scores in the cluster's mean key-value map. Cloud4Trends then generates a summary description for each cluster comprising of few member terms or phrases based on their scores and their significance (hashtags, title terms, etc), while the high-ranked clusters shape the trending topics for the given time.

By leveraging the clusters' members' information additional analytics can be extracted for each trend such as when it first appeared and with which resource, the time-period it spanned, and how it evolved.

In Cloud4Trends trends are therefore calculated based on the three different data sources for each location into investigation, while they can be visualized in a web-based user interface separately for each location and type of resource. By leveraging the analysis results, such an interface can also visualize the history of trending topics and also compares the popularity of a given topic in given different locations. Depending on the update rates of the resources' types (e.g. faster in Twitter while slower in blogs), one can decide on how often the clusters' "trending scale" will be recalculated.

4. THE TREND DETECTION CLOUD-BASED FRAMEWORK

Since the scope of the proposed application is concentrated on Web social data mining and analysis, it is prerequisite to collect and compute online large datasets from web social services. In addition, the versatile nature of the analysis in the specific field (i.e. different information is needed when mining tweets compared to blogs) poses a requirement to collect data from different Web 2.0 application services based on certain criteria, and depending on each use case scenario.

The problem addressed by Cloud4Trends is certainly data intensive as it involves concurrent analysis of large sizes of web social data in an online fashion. Since our approach handles both tweets, with unexpectedly peaks, and blogs whose sizes may be

considerably large, problems referring to handling large and fluctuating sizes of data arise, and feasible approaches have to be followed in order to address them. In particular, parallel programming techniques are required for many operations in our application, as for example:

- data should be concurrently analyzed for the different geographic areas and their analysis should be fast;
- blogs and Twitter data should be collected in parallel;
- since our data collection module should be constantly available for receiving new data, data that have already arrived should be analyzed in a different concurrent process.

Thus, the suggested ideas can heavily utilize the Cloud computing paradigm which offers a significant ground for such social streams mining applications due to its support via scalable and powerful infrastructures [8]. Our design requirements match well with the *MapReduce* computing paradigm, which codifies a generic "recipe" for processing large datasets when this processing consists of more than one stage. The MapReduce technology matches the needs of the Clou4Trends data analysis and processing tier, given that in a cloud-based deployment the *mapping* operations can be distributed into separate computer nodes. Prior to being ported to the Cloud, Cloud4Trends ran into a multi-core computer, designed over a serializable software architecture which posed obstacles in aggregating and analyzing data from both Twitter and blogs. We believe that parallel approaches in cloud computing infrastructures constitute viable solutions for real-time large-scale data mining applications.

One of Cloud4Trends research aims is to validate the quality of the resulting web content clusters and observe and quantify the differences in the trends resulting from the three data sources which represent different user groups. Moreover, we envision conducting this kind of analysis for several geographic areas. As future extension we want to implement a visualization layer so that it attracts a number of end-users who are interested and may benefit from our identified trends and their history. These end users should have the option of inputting a number of general categories of their interest which will be matched by the application, based on a tailored concept ontology, with the corresponding topic clusters and their rank in the "trending scale". The aforementioned research and exploitation primary and secondary priorities certainly impose extra demands on computing resources. Since most researchers have difficulty in acquiring a powerful datacenter, we believe that Cloud infrastructure can be leveraged for efficiently handling both the data's high scalability, the requirement for real-time tweet processing and clusters' update, as well as for ensuring quality of service for an increasing number of end users.

4.1 The VENUS-C Infrastructure

The suggested system (as described in previous section) is currently implemented in the context of the so called "Cloud4Trends" experiment[12] entitled "*Leveraging the Cloud infrastructure for localized real-time trend detection in social media*", which runs over the VENUS-C infrastructure. VENUS-C[13] (*Virtual Multidisciplinary EnviroNments USing Cloud Infrastructures*) is a pioneering project that develops and deploys a Cloud computing service for research and industry communities in Europe by offering an industrial-quality, service-oriented platform based on virtualization technologies and taking advantage of previous experience on Grids and Supercomputing, to facilitate a range of research fields through easy deployment of end-user services.

VENUS-C offers several service components to allow a wide range of end-user applications (targeting mainly research groups and SMEs) to benefit from the advantages of a Cloud computing platform, without having to develop custom Cloud-aware solutions. It offers a selection of programming models that, combined with appropriate data access mechanisms, constitute a convenient abstraction for deploying scientific applications on top of plain virtual machines. Each programming model is enacted behind a job submission service, where researchers can submit jobs and manage their workload [11]. VENUS-C programming model enactment services expose their functionality via an *Open Grid Forums Basic Execution Service* (OGF BES) [14] and *Job Submission Description Language* (JSDL) [15] compliant web service interface, and take care of the enactment of a job at a given Cloud Provider. Currently, two programming models are implemented in VENUS-C for the efficient execution of different processes on the cloud based on complex patterns (such as Map/Reduce or Workflows), namely COMPSs and a VENUS-C tailored implementation of Microsoft's Generic Worker [12]. Each enactment service deploys a specific application on a number of either Windows Azure[14] virtual machines or Unix virtual resources from open source Cloud middlewares (OpenNebula[15] & EMOTIVE Cloud[16]). To run their applications, end-users should first upload their executables at a Cloud-based application repository so as to be accessed by the enactment service depending on the description of an incoming job request.

A Cloud-based data management SDK is provided by VENUS-C for handling all data transfer operations between the Cloud infrastructure and the on-premises client applications, which supports the Storage Networking Industry Association (SNIA) Cloud Data Management Interface (CDMI) specification [16]. The CDMI interface is exposed by a separate web service deployment, and enables unified access to different cloud providers, thus promoting interoperability. Resource monitoring has also been taken into account in VENUS-C so that each end-user is to be able to identify its application's resource consumption via the VENUS-C Accounting Service.

4.2 The Cloud4Trends Cloud-based Architecture

The suggested design is implemented in Cloud4Trends under the previously described 3-tiered conceptual design structure. Cloud4Trends is implemented as a hybrid application based on the cooperation of: i) on-premises client interface components and ii) multiple job execution components with different functionalities on top of the VENUS-C Cloud services infrastructure. In particular, Cloud4Trends uses VENUS-C Generic Worker programming model for job submission and application deployment on top of the Azure Cloud infrastructure. The Generic Worker is best suited for data-driven task-based job submissions, with each job description specifying a number of input file dependencies and requirements for the generation of some output files, in a command line-like execution format.

[12] http://oswinds.csd.auth.gr/?page_id=1320
[13] http://www.venus-c.eu/
[14] http://www.windowsazure.com/
[15] http://opennebula.org
[16] http://www.emotivecloud.net/

Fig. 2 illustrates the three Cloud4Trends modules, namely: the Collect module, the on-the-Cloud module, and the Cloud services module. These modules support the proposed 3-tier design (described above) such that the Client module implements the Data Collection and Visualization tiers, whereas the Cloud-based module implements the Data Processing and knowledge /extraction tier.

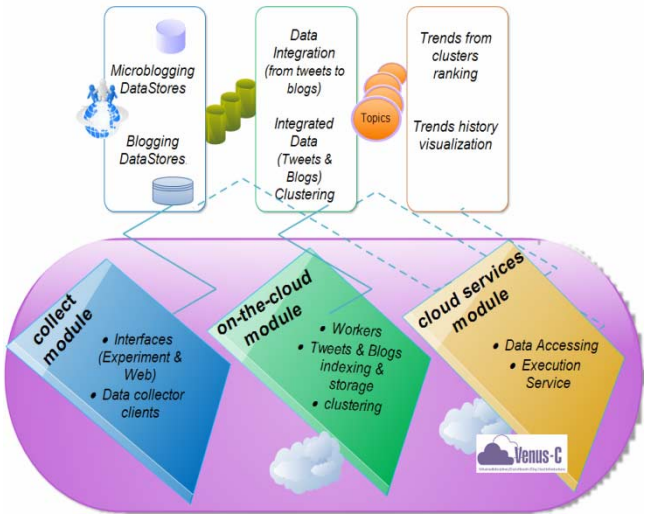

Figure 1. The Trend Detection Cloud-based framework

More specifically, the proposed framework involves three modules as discussed next:

- collect module: it operates as a client module and involves the required blogging and microblogging data collectors and their interfaces required for the communication with the VENUS-C infrastructure (and its services). These refer to the interfaces needed for supporting the experiment's setting and monitoring, as attached to the on-premises server. The client module facilitates new experiments' submission and progress monitoring for the ones currently running. The web interface is needed to communicate the Cloud4Trends results to end users, as well as to implement the Twitter and Blog Data Collectors, receiving stream data in real time or at specified time intervals;
- "on the cloud" module: involves the Cloud4Trends Data analysis and Processing Tier, which has been ported to the Cloud. In particular, data Parsing and Clustering modules have been deployed via the VENUS-C Services, i.e. the related operations are submitted as jobs via the enactment service under the Generic Worker programming model. The clustering modules are realized by the Splitter, Similarity Calculation (Mapper), and Aggregation (Reducer) modules, under the Map-Reduce paradigm. The Indexing Services module is implemented independently as a separate set of Cloud services that are responsible for creating indexes for each type of input data (tweets, blog posts, extended tweets);
- cloud services module: it involves the specific VENUS-C components used for assisting and simplifying the application's porting to the Cloud. More specifically, the VENUS-C Data Access SDK is used for accessing the Cloud Storage (Blobs and Tables) when retrieving or uploading data via a Client, while the VENUS-C Execution (enactment) Service is used for submitting, distributing, and setting up new processing jobs to the Cloud.

An indicative workflow is orchestrated as follows. The collect module, which is responsible for collecting data from the Web (Twitter and the Blogosphere) and initializing new experiments when required by the researchers, submits new **Parsing** jobs (executed by Twitter or Blog Parsers) to the cloud via the **Job Submission Client of Generic Worker,** which is hosted at the Execution Service, when new data are available. The required data for each given job are uploaded, as a batch, using the **Data Access Service** to **Azure Blobs**. Cloud4Trends' **Indexing Service Module** consists of three separate Azure services (for indexing tweets, blog posts, and extended tweets) that are responsible for the Full Text Indexing of the parsed data using the a Lucene library for Azure[17]. These services are notified that new (parsed) data are available for indexing by polling three dedicated Azure queues, respectively. When an Indexing service completes its execution it initiates a **Splitter Job** using the **Generic worker's Job Submission Client**, which receives as input data the new resources' representations under the model described in Section 3.2. Such representation (including the combined TF-IDF map structures are emitted by the Indexing services). The **Splitter** application also downloads the appropriate currently active clusters from the corresponding Azure Table. As long as our system handles the concurrent analysis of several new data batches, different jobs should be instantiated and executed in parallel to effectively process all data. According to that, different Similarity Calculation Workers (**Mapper jobs)** are submitted by each **Splitter job**, via the Execution Service module, and in particular using the Generic Worker **Local Job Submission service,** which calculate the similarity scores between the new resource and the respective active clusters. An **Aggregation Worker (Reducer job)** is also submitted by the **Splitter Job** via the **Local Job Submission service**, to collect all similarity score combinations emitted by the corresponding **Mapper** jobs, and identify the best match to an existing cluster for each resource, or initiate a new cluster for any item whose similarity with each existing cluster is below a specific threshold.

5. IMPLEMENTATION DETAILS AND FUTURE OUTLOOK

The Cloud4Trends pilot project was initiated in June 2011 and is now at an Alpha Prototype state. So far, the following steps have been taken.

- the on-premises data (tweet and blog post) collection applications have been seamlessly integrated with the job submission client into a real-time text parsing job submission system.
- the Generic Worker's suitability for frequent successive job submissions has been verified.
- the data indexing services have been deployed in Azure using the Azure Library for Lucene.NET.
- communication between the Cloud-based data parsers and the indexing services has been achieved via queues.

[17] http://code.msdn.microsoft.com/windowsazure/Azure-Library-for-83562538

- the whole job submission workflow (including the submission of parsing and distributed clustering jobs) has been deployed in the Cloud. This was accomplished with use of the VENUS-C enactment service and the Generic Worker, which allowed the implementation of the workflow via the submission and "cooperation" of data-dependent jobs, as well as through the queue-based communication between Generic Worker computing instances and the Indexing Cloud-based services.
- scalability issues are addressed by a special component that pulls the status and time duration of submitted jobs using the VENUS-C Accounting Service, and increases the number of virtual machine instances when delays are observed for a representative number of submitted jobs.
- a separate component has been developed that communicates with the dedicated Cloud Tables for monitoring the clusters' "activity" and for updating their states.
- clusters' representations are downloaded to the (local) client-side component at a parameterized time interval, and then ranked for "hottest" trends identification.

Future work is focused around the following axes: i) the fine-tuning of the clustering and trend detection algorithm and the experimental evaluation of results, ii) the implementation of a shard-based distributed Indexing service since for the time being the service for each type of resource is deployed on a single instance, iii) measuring the system's performance for different design parameters, and iv) creating a web-based user interface (hosted either on Cloud or on premises) for visualization of the detected real-time trends and trends' analytics.

The benefit that Cloud4Trends offers is that it verifies that Cloud-based architectures constitute a viable solution for online web data mining applications which can be beneficial for both researchers and entrepreneurs. In particular, the proposed framework enables massive data analysis at a distributed setting, thus reducing the prerequisite for real-time applications data processing time. At the same time is allows easier testing of new algorithms/use-case scenarios, achieving high-quality results with lesser cost, allowing developers/scientists to focus on applications' refinement/testing, rather than on how to setup and operate the underlying infrastructures. Moreover, Cloud-based solutions can improve application capability sharing, by easily exposing research results to other research groups or interested end-users.

In summary the proposed framework (as realized by the Cloud4Trends experiment), demonstrates that porting trend detection into the Cloud is a very suitable solution considering the challenges posed by the data and time intensive processes involved in online collection and analysis of large and evolving Web 2.0 datasets.

6. ACKNOWLEDGMENTS

This work is partly funded and realized within the Cloud4Trends pilot project of the VENUS-C project. VENUS-C (Virtual multidisciplinary EnviroNments USing Cloud infrastructures) is co-funded by the GÉANT and e-Infrastructures Unit, DG Information Society and Media, European Commission. VENUS-C brings together 14 partners from Europe. Microsoft invests in Azure resources and manpower through Redmond and its European research centres.

7. REFERENCES

[1] N. S. Glance, M. Hurst, and T. Tomokiyo: BlogPulse: Automated Trend Discovery for Weblogs. In WWW Conference. 2004.

[2] M. Uchida, N. Shibata, and S. Shirayama: Identification and Visualization of Emerging Trend from Blogosphere, in proceedings of *International* Conference on Weblogs and Social Media (ICWSM), pp. 305-306, 2007.

[3] M. Mathioudakis, N. Bansal, and N. Koudas: Identifying, attributing and describing spatial bursts. *Proc. VLDB Endow.* 3, 1-2 (September 2010), 1091-1102. 2010.

[4] B. E. Teitler, M. D. Lieberman, D. Panozzo, J. Sankaranarayanan, H. Samet, and J. Sperling: NewsStand: a new view on news. In GIS '08. ACM, Article 18, 10 pages. 2008.

[5] J. Sankaranarayanan, H. Samet, B. E. Teitler, M. D. Lieberman, and J. Sperling: TwitterStand: news in tweets. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems* (GIS '09). ACM, 42-51. 2009.

[6] M. Mathioudakis and N. Koudas: TwitterMonitor: trend detection over the twitter stream. In *Proceedings of the 2010 international conference on Management of data* (SIGMOD '10). ACM, 1155-1158. 2010.

[7] V. Koutsonikola, A. Vakali, E. Giannakidou, I. Kompatsiaris: Clustering Users of a Social Tagging System: A Topic and Time Based Approach. WISE 2009: 75-86. 2009.

[8] I. Livenson and E. Laure: Towards transparent integration of heterogeneous cloud storage platforms. In DIDC '11. ACM, 27-34. 2011.

[9] M. Giatsoglou and A. Vakali: Capturing Social Data Evolution via Graph Clustering. *IEEE Internet Computing*. 2012. Preprint. DOI: 10.1109/MIC.2012.24.

[10] Spärck Jones, Karen: A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation* **28** (1): 11–21. 1972. DOI:10.1108/eb026526.

[11] D. Lezzi, R. Rafanell, F. Lordan, E. Tejedor, R.M. Badia: COMPSs in the VENUS-C Platform: enabling e-Science applications on the Cloud. In *Proceedings of 4th Iberian Grid Infrastructure Conference*, Santander, 2011, 73-84.

[12] Yogesh Simmhan, Catharine van Ingen, Girish Subramanian, and Jie Li: Bridging the Gap between Desktop and the Cloud for eScience Applications. In *Proceedings of the 2010 IEEE 3rd International Conference on Cloud Computing* (CLOUD '10). IEEE Computer Society, 474-481. 2010.

C. Taylor, (June 27, 2011): Social networking 'utopia' isn't coming. CNN. http://articles.cnn.com/2011-06-27/tech/limits.social.networking.taylor_1_twitter-users-facebook-friends-connections?_s=PM:TECH.

[13] Foster I. et. al:OGSA Basic Execution Service Version 1.0. Grid Forum Document GFD-RP. 108. 8 August 2007.

[14] Savva A (Editor). Job Submission Description Language (JSDL) Specification, Version 1.0. Grid Forum Document GFD-R.056. 7 November 2005.

[15] SNIA CDMI: http://www.snia.org/cdmi