Informatics Department
Aristotle University of Thessaloniki

OSWINDS
research group

A distributed framework for early trending topics detection on big social networks data threads

ATHENA VAKALI, NIKOLAOS KITMERIDIS, MARIA PANOURGIA

INFORMATICS DEPARTMENT,

ARISTOTLE UNIVERSITY, THESSALONIKI, GREECE

Presentation outline

- Introduction
- Problem Addressed
- A framework for early trend detection in social networks
- A micro-blogging trending topics prediction implementation
- Experimentation and Results
- Conclusion and Future Work

Abstract

Social networks have become **big data production engines** and their analytics can reveal **insightful trending topics**, such that **hidden knowledge** can be utilized in various applications and settings.

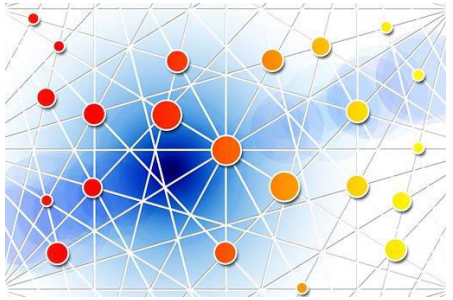
This paper addresses the problem of popular **topics' and trends' early prediction** out of social networks data streams which demand **distributed software architectures**.

Under an online time series **classification model**, which is implemented in a flexible and adaptive distributed framework, trending topics are detected.

Emphasis is placed on the early detection process and on the **performance** of the proposed framework. The implemented framework builds on the **lambda architecture** design and the experimentation carried out highlights the usefulness of the proposed approach in early trends detection with high rates in performance and with a validation aligned with a popular microblogging service (Twittter).

Social Networks = Big Data Production Engines

- Many TeraBytes of user-generated content (UGC) per day…
- **Hidden knowledge** can be utilized in various domains :
  - Advertising
  - Social Sciences
  - Politics
  - Recommendation Systems
  - many more…
- Two crucial aspects justify such knowledge's importance :
  - capture crowd's interests (i.e. topics)
  - Monitor dynamics over time slots (i.e. emerging trends)



Twitter's Trending Topics

WORLD WIDE

Worldwide Trends · Change

#MakeAHorrorFilmLessScary
26.3K Tweets

#سلمان_الحزم_بامر_بقصاص_امير
51.5K Tweets

#Aprender
35.9K Tweets

#KarmaEğitimİstemiyoruz
7,695 Tweets

#FelizMartes
54.9K Tweets

Red Dead Redemption 2
137K Tweets

Chuck Berry
22.5K Tweets

PCC e Comando Vermelho
3,824 Tweets

Forró
16.1K Tweets

ABRACADABRA QUE APAREZCAN PYF
11.9K Tweets

Trending topics are those **topics** being discussed more than others.

Twitter Trends are automatically generated by an algorithm that attempts to identify **topics** that are being talked about more right now than they were previously.

REGIONAL

Greece Trends · Change

Πεδουλακης

Κετσπαγια

#ΘΕΜΕΤΟΣ
3,437 Tweets

Μοραις

Τσατανη

Δικαιοσυνης

Ποταμιου

Ανδρο

#Xanadestetous

ΑΝΕΛ

Problem(s) Addressed

- Twitter provides information about the most discussed subjects (trending **topics**) among its users any **time**;
- … but as the production rate of such data constantly increases, questions like the following arise between both in scientific and enterprise communities :
  - Can we do it better ? // qualitative issues
  - Can we do it faster? // quantitative/performance wise
  - How to manage such data volumes need for extracting valuable hidden knowledge?

  … both in terms of software and hardware.

from data to trends … (I)

- Trend detection problem is not trivial
  - popularity is not enough … **time of appearance the topic** is a key factor too!
- Several studies address the problem as a **time series classification** problem

Proposed execution approach novelty is due to the next key points:

- training set's time series are constructed by a **small rate of the overall tweets** dataset;
- the time series of topics to be tested are **not static**, but are generated in real time in a form of a **sliding window**, maintaining low percentage of tweets for constructing the time series.
- Features used :
  - **timestamps and exact date** of the topic being declared as a trending topic;
  - definition of a **time range** from the moment just before the topic became trended, i.e. the time of the last kept time slot;
  - production of the final form of the time series by applying a set of **normalization** filters;
  - Two distance metrics of **cosine** and **squared Euclidean** are utilized for the comparison between time series.

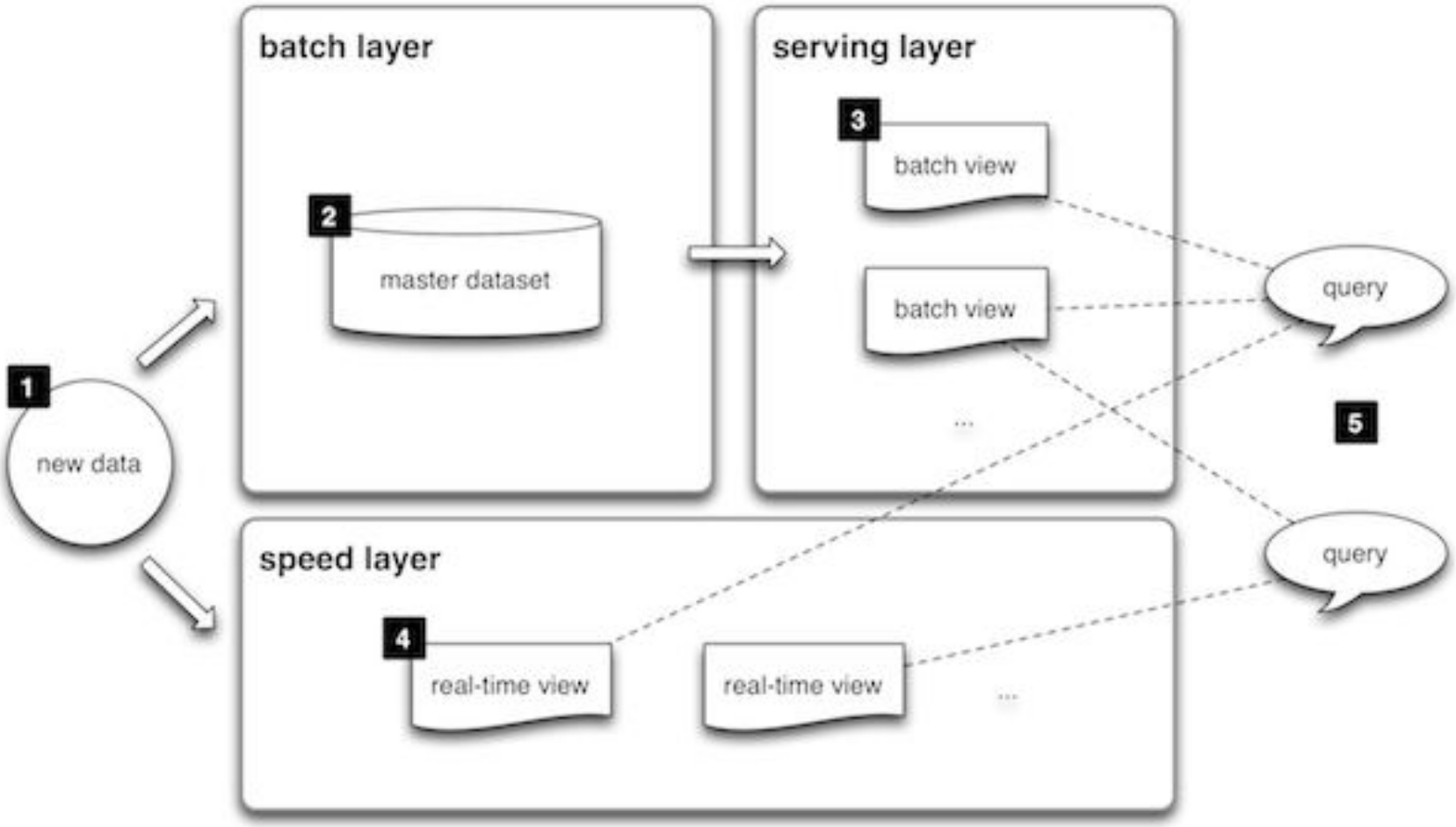
from data to trends … (II)

- Retrospective analysis of data is usually needed, but it is not enough in many cases … since
  - The basic entity of this research, the **trending topic, has a short life cycle**.
  - We need to be able to define a topic as trending **in almost real time**, i.e. shortly after the first occurrences of the topic in users' posts.
- To resolve such issues, we divide the dataset in two, continuously updated, categories :
  - The first one is **the training set** which is used for the comparison with the new topics' time series.
    - It is consisted of valid trending topics' time series (created by the framework and validated against Twitter's API)
    - It is continuously updated with new trending topics' time series, to enrich the dataset and cover new time series patterns
  - The second dataset consists of every new topic's time series, which are constructed in **real time**, are divided in 2 minutes time buckets and are updated in a manner of sliding window.
    - Pattern similarity among real time topic's and training set's time series is used to detect the new trending topics.

Handling Big Data streams

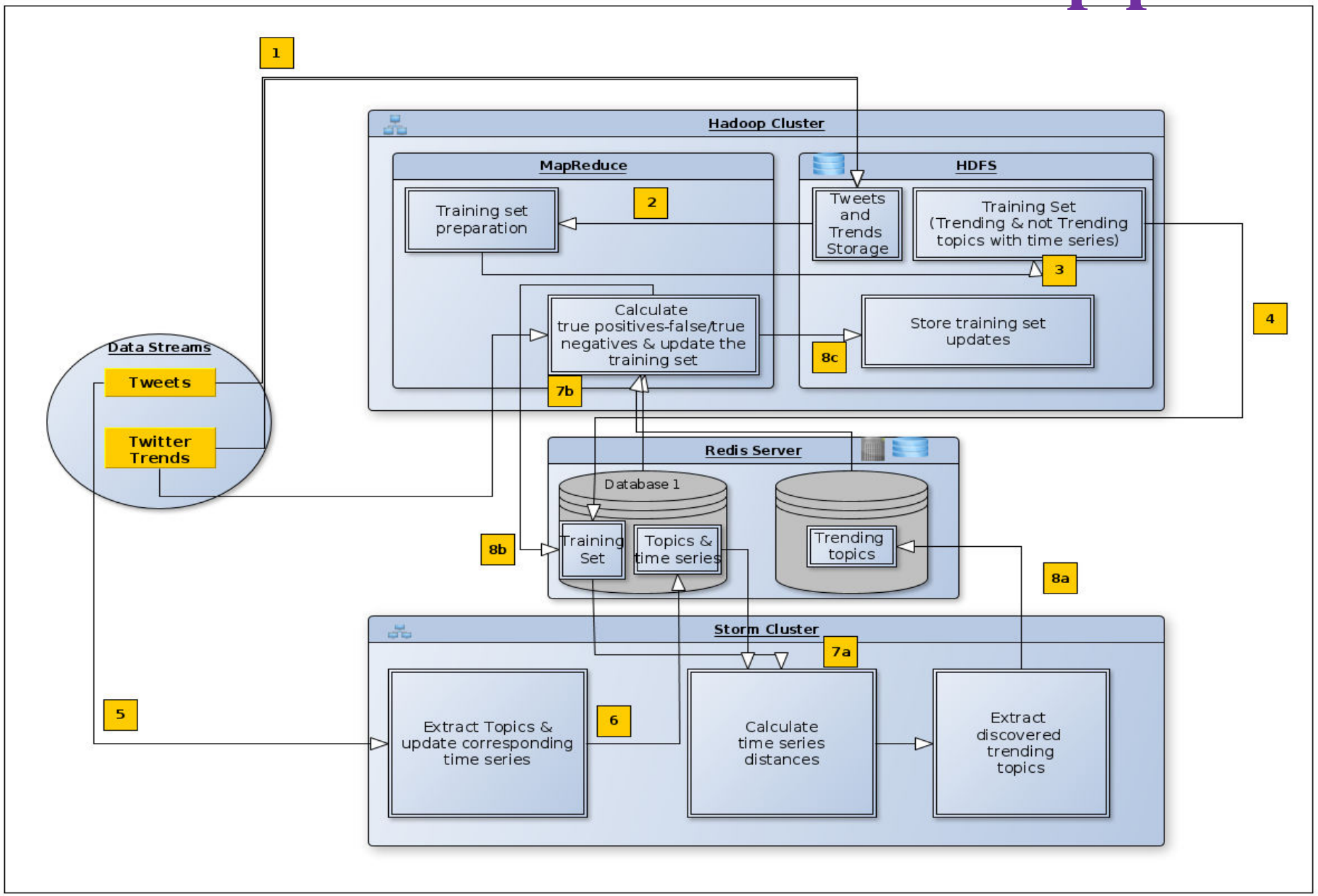
- Complex architectures have been designed and tested on evolving big data management.
  - Current work is based on an architecture that is inspired from **Lambda Architecture**'s principles, allowing both batch and real-time stream data processing.
    - **Batch Layer**
      - Apache Hadoop (MapReduce, HDFS)
      - Executes periodic processing in whole dataset.
    - **Speed Layer**
      - Apache Storm
      - Computes data views incrementally by balancing high latencies of the batch layer with real time views computations.
    - **Serving Layer**
      - Redis servers
      - The first one gets the output from the batch layer which contains pre-computed views and exposes views for querying
      - The second one stores the real time created time series for each topic.

Handling Big Data streams



http://lambda-architecture.net/img/la-overview_small.png

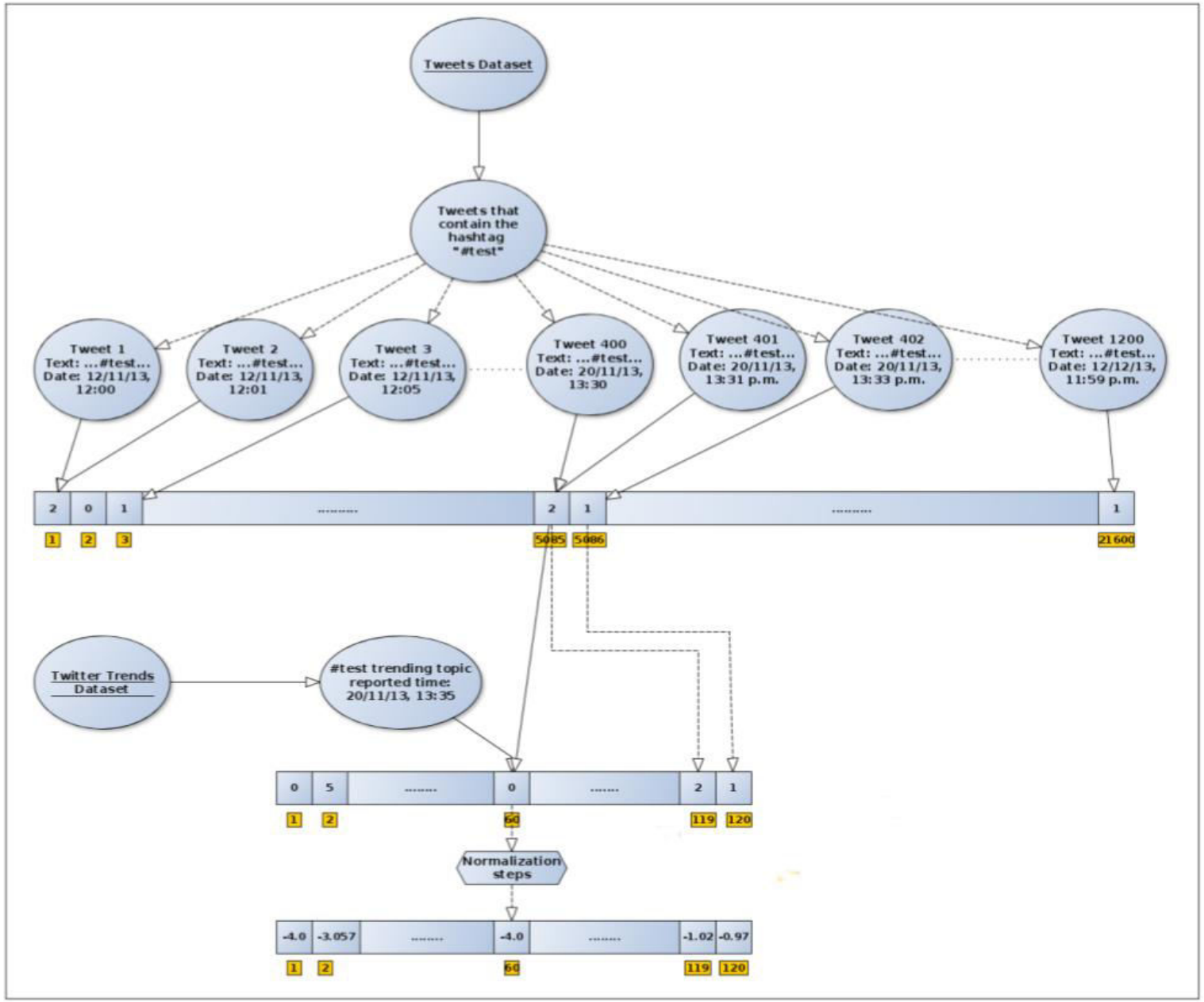
An abstract view of the overall 8-step process



Handling Big Data – The initial Data Set -

- ***Step 1:*** One month of collected tweets and trending topics dataset, reported from Twitter.
  - Stored in HDFS
- ***Step 2:*** Processing
  - Text processing and topics detection (eg. hashtags)
  - Creation of each topic's time series based on their occurrence in users' posts
- ***Step 3:*** sampling time series and trimming
  - A sample of both trending and non-trending topics' time series is chosen
  - The time series trimming
    - Trending topics: 2 hours before and 2 hours after the topic has been reported as trending
    - Non trending topics: Random 4 hours range
  - Normalization of time series
- ***Step 4:*** Storage of the training set time series to the serving layer so they can be accessible from the speed layer

Trending topics time series example



Handling Big Data – Real time execution

- ***Step 5:*** Real time Tweets collections from twitter Streaming API
  - Speed layer (Apache Storm)
- ***Step 6:*** Topics extraction from real time tweets and the creation/update of the corresponding time series.
  - *EXAMPLE* : Time series of 180 time buckets, each one lasts 2 minutes i.e. whole range of a topic's time series is 6 hours.
  - Topic's first appearance
    - Before six hours of execution:
      - The initialization of the topic's time series is performed by calculating the specific 2 minutes time range from the difference between the tweet's date and the start time of the program.
      - The corresponding time bucket is assigned with value 1,meaning that there is a tweet posted in the specific time range that contains the topic.
      - The rest 179 time buckets are assigned with value 0 in this phase.
    - After six hours of execution:
      - the first occurrence of a topic in this case signifies the construction of its time series with the first 179 time buckets' values to be 0 and the value of the last time bucket to be 1.
  - Time series update (topic's reappearance):
    - Increase corresponding time bucket by one
    - Shift time buckets by one position every 2 minutes
      - Discard the oldest one and add a new one

Handling Big Data – Real time execution

- ***Step 7a:*** Periodic calculation of the distance between examined time series and training set's time series, using the following equation:

$$R(s) = \frac{\sum_{r \in R_+} exp\left(-\gamma \min_{k=1,\ldots,N_{ref}-N_{obs}+1} \sum_{i=1}^{N_{obs}} \mathbf{d}(s_i, r_{i+k-1}) \right)}{\sum_{r \in R_-} exp\left(-\gamma \min_{k=1,\ldots,N_{ref}-N_{obs}+1} \sum_{i=1}^{N_{obs}} \mathbf{d}(s_i, r_{i+k-1}) \right)}$$

r	A time series of the training set
R_+	The set of time series of the training set that correspond to trending topics.
R_-	The set of time series of the training set that correspond to not trending topics.
N_{ref}	The length of each time series of the topics of the training set.
N_{obs}	The length of each time series of the topics their trending probability is being observed.
s_i	The i_{th} time bucket of a topic's time series that is being observed.
r_i	The i_{th} time bucket of a topic's time series that belong in the training set.
γ	A scaling parameter for fine-tuning.
$\mathbf{d}$	The distance metric been used, among Euclidean, Squared Euclidean and Cosine distances.

- ***Step 8a:*** Storage of time series that their ratio is above a predefined threshold, which means that they are possible trending topics

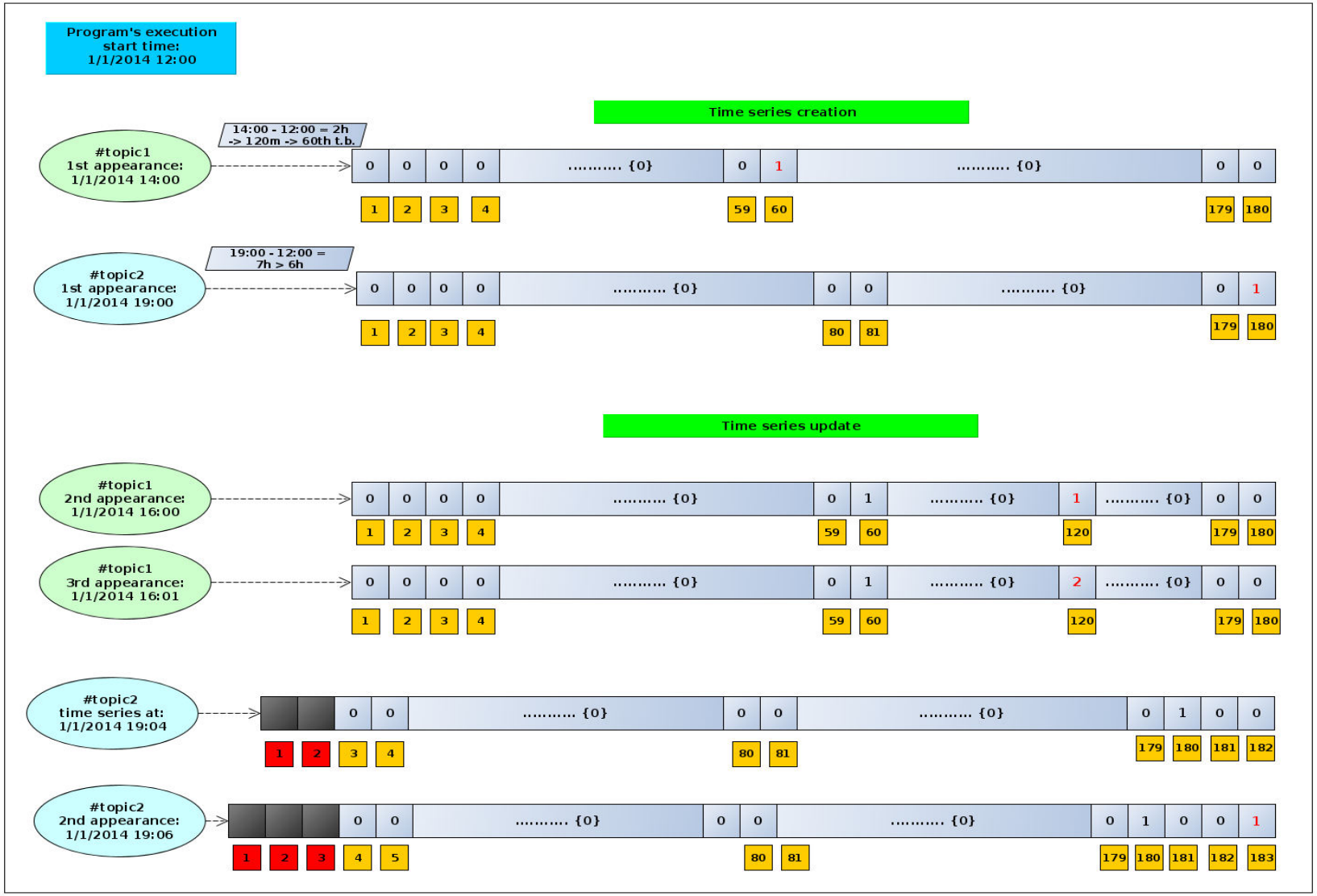
Handling Big Data – Training set's Periodic update

- ***Step 7b:*** Time series, that will be added in the training set, contributing in the next execution cycle of the speed layer, are consisted of:
  - Topics' time series that are stored in the first Redis database of the serving layer at the specific time that the execution cycle is performed.
  - Topics' time series that have been characterized as **potential trending topics** from the procedure that has been executed on the speed layer until the aforementioned time.
  - Topics' time series that have been reported as trending topics form Twitter and have been collected through the Twitter API, until the aforementioned time.

- ***Step 8b:*** Normalization of the previous step's time series and migration to the serving layer
  - For availability in the next evaluation cycles

- ***Step 8c:*** Migration of the previous step's time series in the batch layer
  - **Periodic checks** on this dataset to avoid biased time series patterns' in the evaluation dataset

Example of real time, time series creation & update

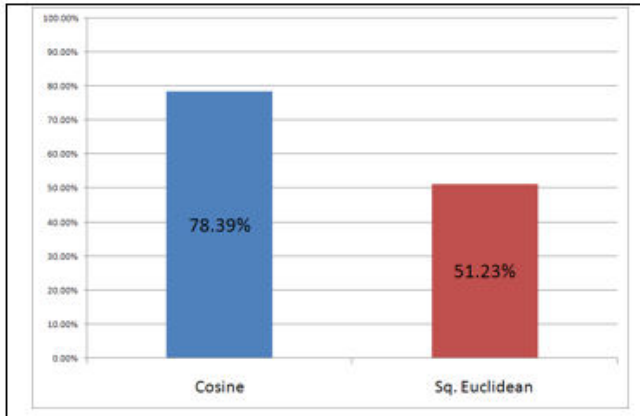


Experimentation and Results (I)

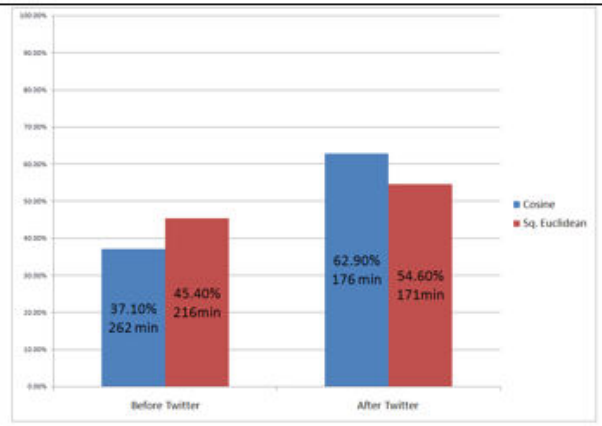
- The proposed framework has been stress tested under various **big data tweets threads** collected via Twitter streaming API, over several time windows.
- The evaluation of the proposed methodology implemented on the lambda inspired architecture has reached improved performance since almost **80% of the actual trending** topics were classified as potential trending topics by the method after 48 hours of execution.
- Interesting observation:
  - **Quantity/topics** : The percentage of potential trending topics that was actually announced as trending topics
    - Cosine similarity > Square Euclidian
  - **Quality/strength**: The percentage of actual trending topics that have been observed as such, before being announced from Twitter
    - Square Euclidian > Cosine similarity

Cosine : to measure origin/topic versus
Square Euclidian to measure strength/trend

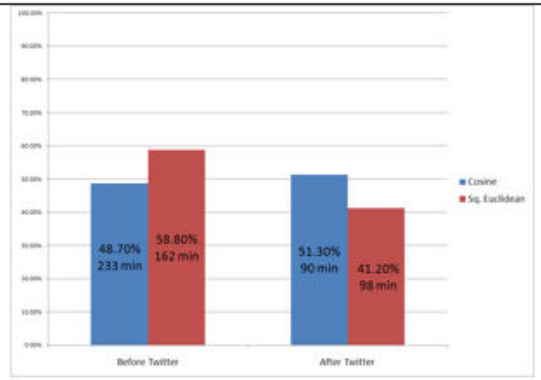
Experimentation and Results (II)



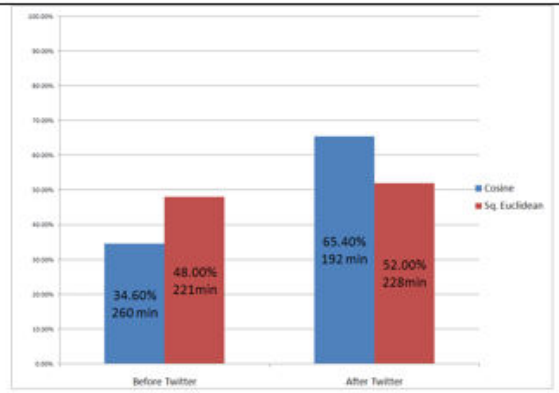
(a) true positives percentages at the end of the 48 hour execution, for Cosine and Squared Euclidean



(c) 36 hours of execution



(b) 24 hours of execution



(d) 48 hours of execution

Experimentation and Results (III)

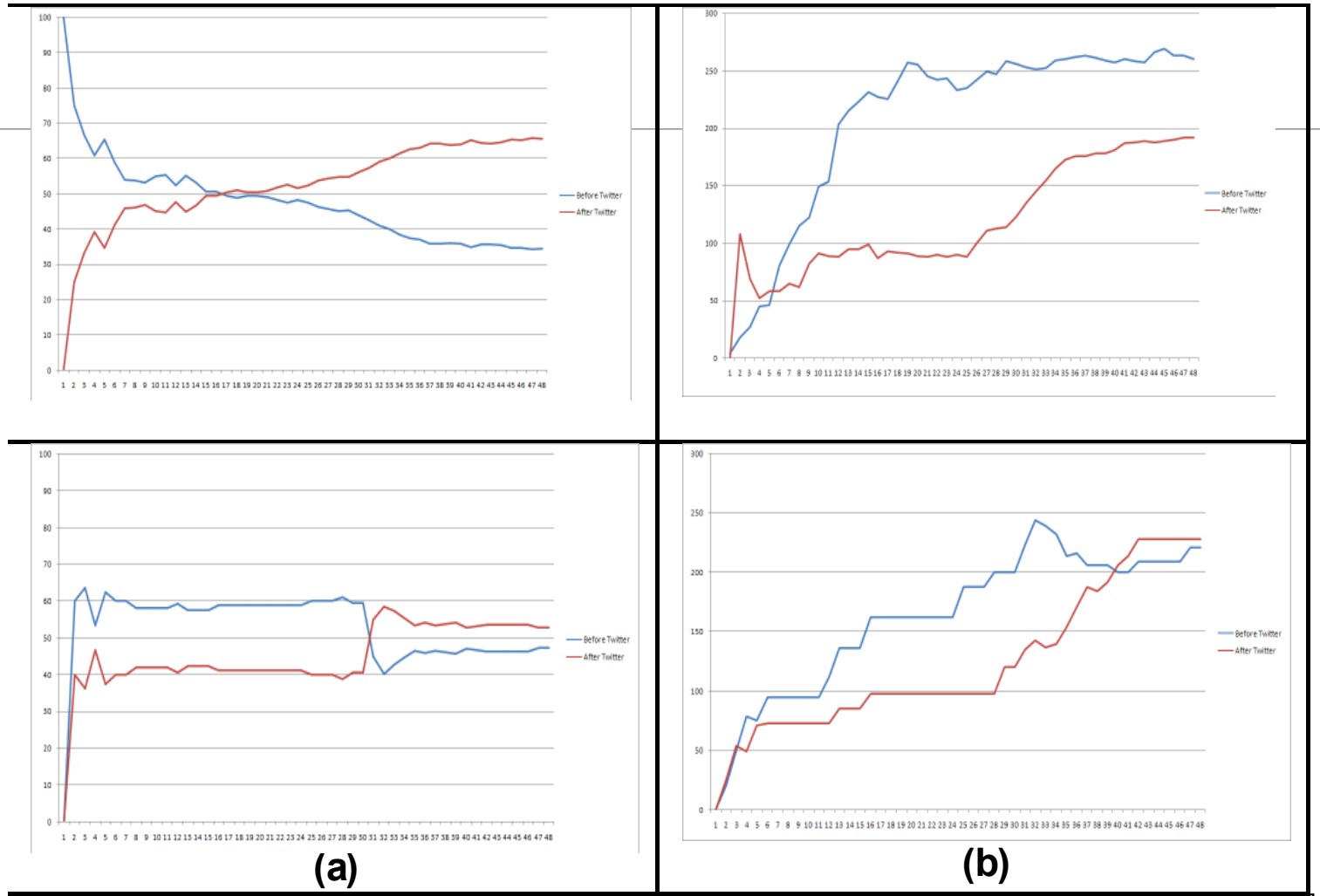


Fig. 4: Hourly Percentages (a) and Mean times (b) fluctuation for trends detection prior and after Twit?

Challenges and Future work

- While the evaluation results are quite satisfactory, further improvements can be achieved. Some of these improvements could be:
  - .. regarding the **quality of results**.
  - T he usage of bigger initial training set, which was limited in the current work due to luck of resources (mainly storage)
    - Target increased percentage of tweets acquired from the official API
    - In bigger time ranges ..
  - Update implementation with usage of newer technologies in the different layers of the presented architecture :
    - Apache Spark
    - Apache Flink
    - Several NoSQL distributed databases

Hardware Resources details ..

- Disk Storage in total: 600GB HDD
- CPU cores in total: 48
- Ram in total: 46 GB

- 14 virtual machines
  - Hadoop cluster (7 VMs)
  - Storm cluster & Zookeeper Server (6 Vms)
  - Redis servers (1 VM)

- Special thanks to GRNET (https://grnet.gr/) for providing these resources through its Cloud Provider Platform Okeanos (https://okeanos.grnet.gr/)

OSWINDS
research group

http://oswinds.csd.auth.gr/