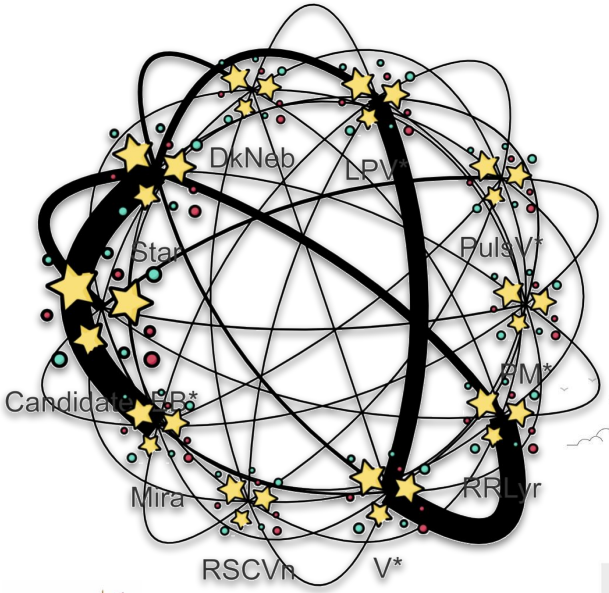
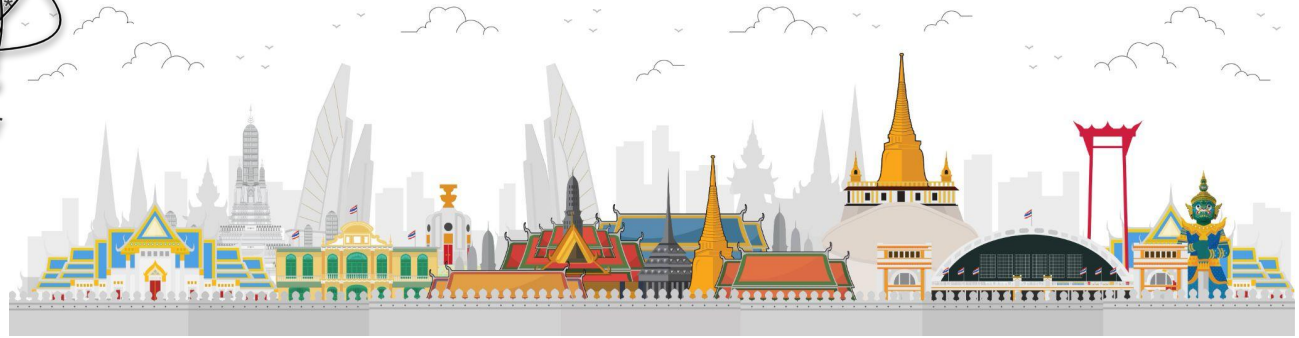
Inteligent Database
of Astronomical Alerts from LSST & ZTF



- **LSST**
  - Fink Broker
- **Classification Graphs**
  - Project
  - Implementation
  - Classification Schemas
- **Usage**
  - CLI
  - WS

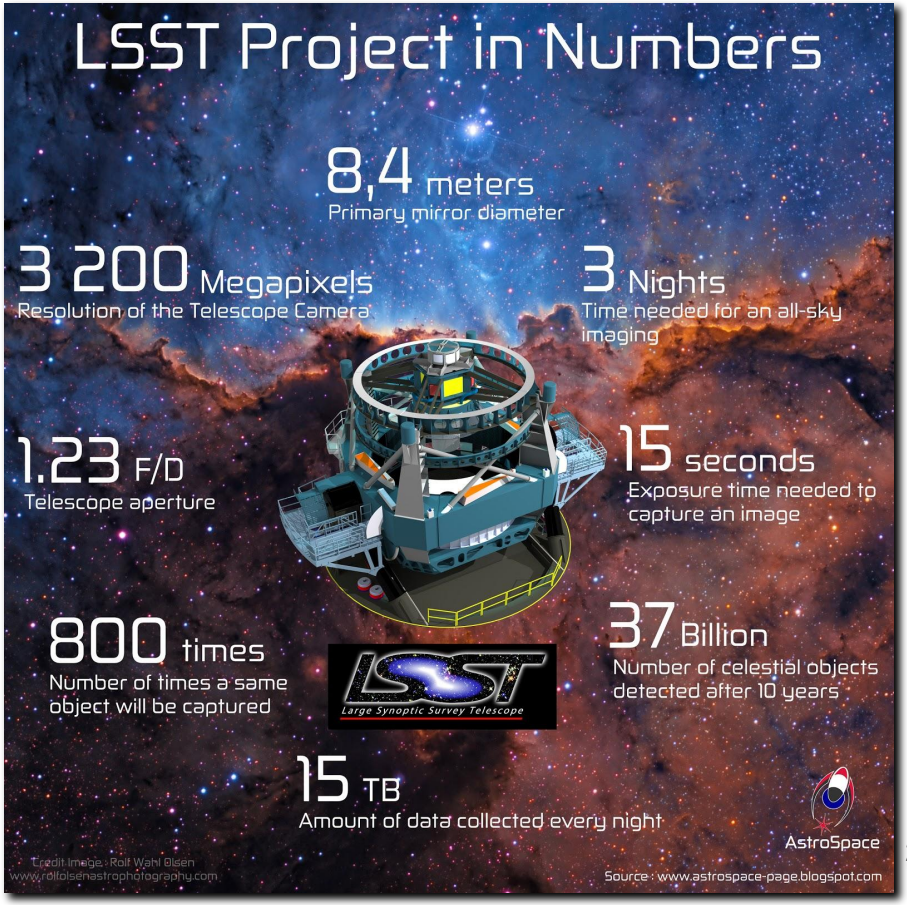
Julius Hrivnac, IJCLab
for FINK Project

CHEP 2026
25-29/5/26
Bangkok



LSST - Vera C.Rubin Observatory

The **Vera C. Rubin Observatory** is an astronomical observatory located on Cerro Pachón in Chile. It will repeatedly scan the southern sky over ten years to create a deep, time-resolved map of the Universe, enabling studies of transient phenomena. It is already delivering first images and alerts.

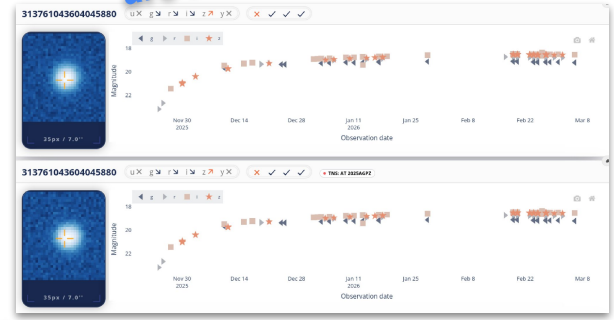


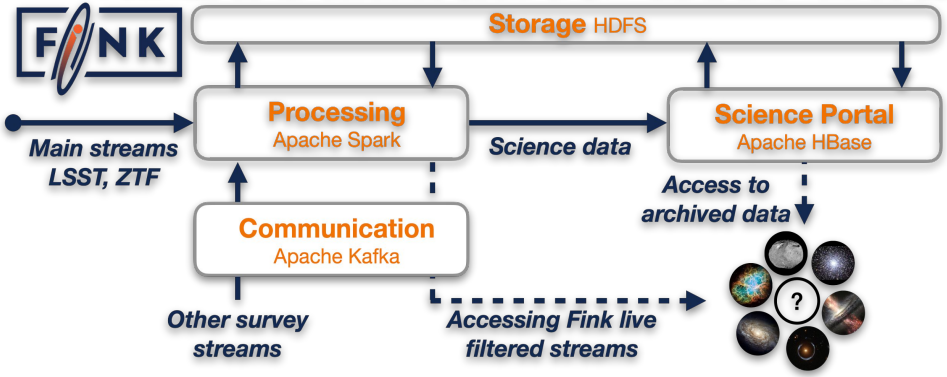
AstroLab - Fink Broker

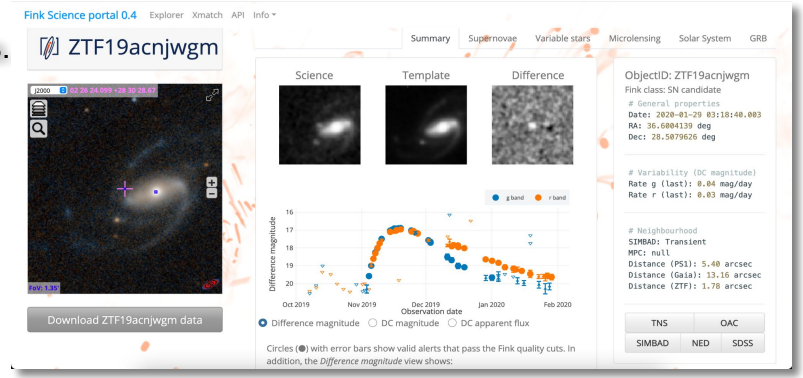
10 000 000 alerts per night

searching for objects which are moving/changing

- **Fink** is an astronomical **alert broker** designed to process the high-volume **transient alert** stream from the *Vera C. Rubin Observatory* and *Zwicky Transient Facility (ZTF)*, acting as an **intermediary between those alerts and the scientific community**.
- It ingests, stores, annotates, enriches, classifies, filters and redistributes alerts adding contextual information needed for rapid scientific interpretation.
- Fink applies feature engineering, stream cross-matches, and machine-learning **classifiers** to identify astrophysical phenomena such as *supernovae*, *variable stars*, *solar-system objects*, etc.
- Fink is open-source and designed to support user-defined science modules, personalized filters, and follow-up workflows. It was designed to be modular and to facilitate the inclusion of personalized analyses.
- Fink has so far collected 250M of ZTF alerts and 10M od LSST alerts.







Classification Graphs

to organise alerts in a useful way

Project

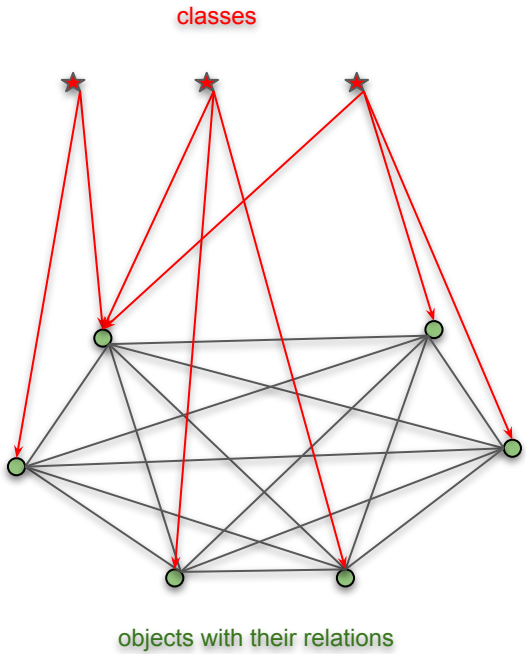
- To organize access to data (**alerts**) allowing **similarity** search and navigation
  - **Vector databases** functionality
- To be flexible about what **similarity** means (its defining object properties) and how to evaluate it (which metric to use)

- For finding similar objects, we should in principle calculate the distance to all other objects
  - Practically impossible for large amount of data
  - To make it feasible, we should
    - either sort the objects (but there is no unique sorting measure)
    - or clusterise them (classification)
- Organizing **classifications** in **graphs**

Simple Classification

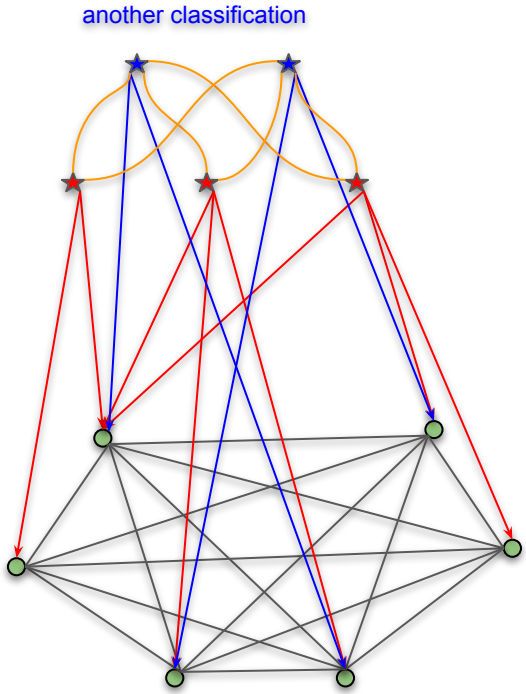
how classification enables similarity search

- How objects classification enables similarity search
  - Without discrete classification, the problem is **`n*n`** (`n = number of objects`), which is undoable for large number of objects
  - With a discrete classification scheme, the problem is **`n*m + m*m + m`** (`m = number of classes, m << n`)
    - `n*m` - classification of new objects, only done when they arrive
    - `m*m` - overlaps, when needed
    - `m` - to get information for a particular object
  - Some classifications should be trained, but that can be done on the subset of objects and re-done only when something changes
- Object distance can be calculated by various metrics:
  - **Jensen-Shannon** - the most trustworthy, but slower
  - Euclidean, Cosine,... - faster, less trustworthy



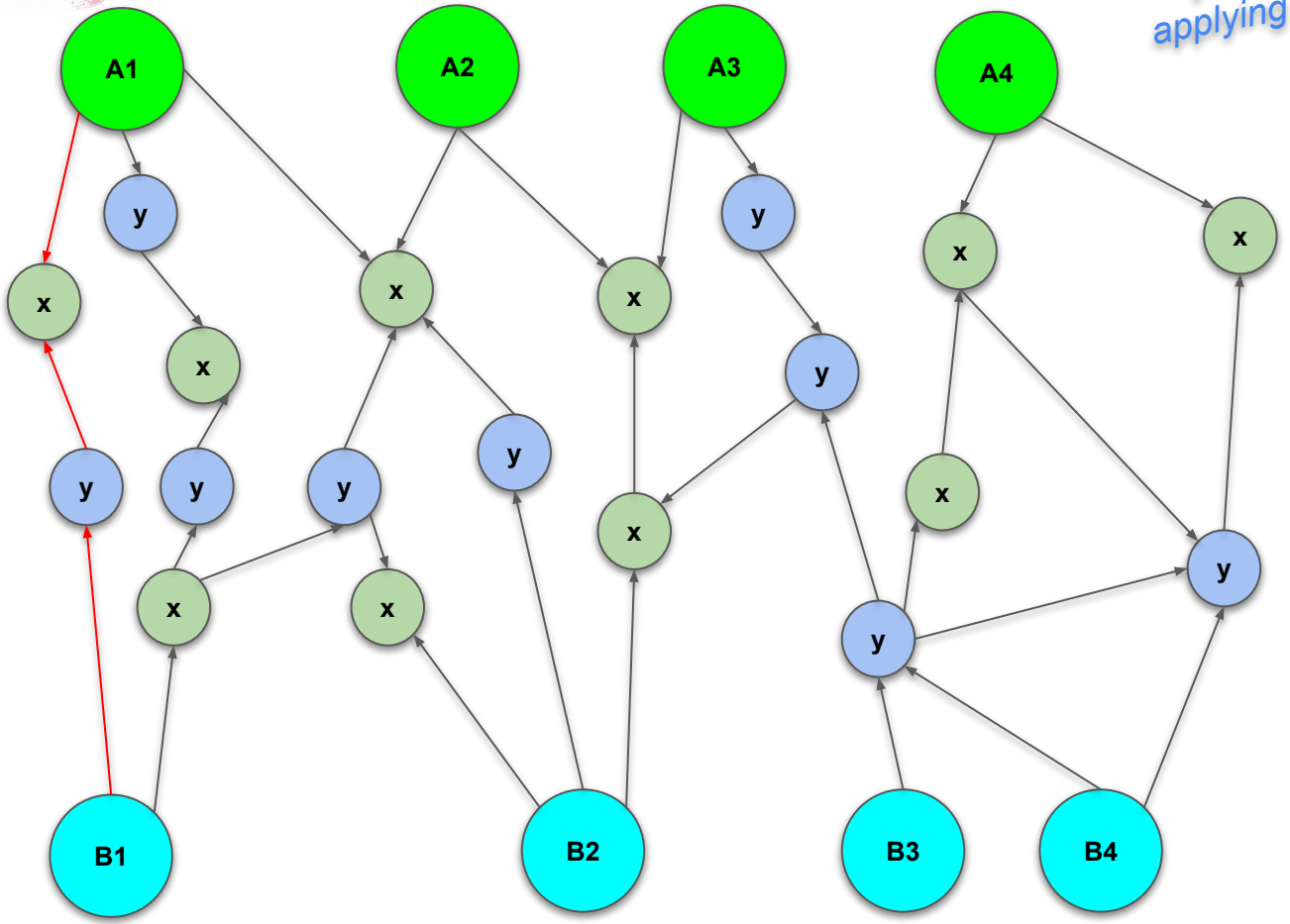
Multiple Classification

- Objects can be classified with several schemas, exploring different kind of similarities:
  - Analytical: calculated for each object individually by an algorithm, or taken from third services (mostly user-supplied filters)
  - AI: calculated by trained network, PCA,...
  - Tags: supplied individually by users
- Classification schemas can have several ***flavours***
  - from different classification sources or hyperparameters,...
- Classification schemas
  - can be **combined** into *superschemas*
  - can be **divided** into *subschemas*
  - can be organised **hierarchically** (for big amount of data)
- Class Distances
  - Are calculated using modified **Graph Flow** algorithms
    - Often used in recommendation systems
- Once existing objects are classified, we can easily (and quickly):
  - Classify new objects
  - Find overlaps between schemas
  - Comparing truthfulness of schemas
  - Interpreting one schema by another schema

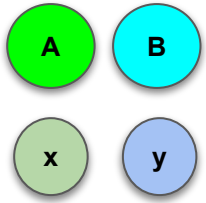


Class Distance Problem

just one example of applying graph algorithms



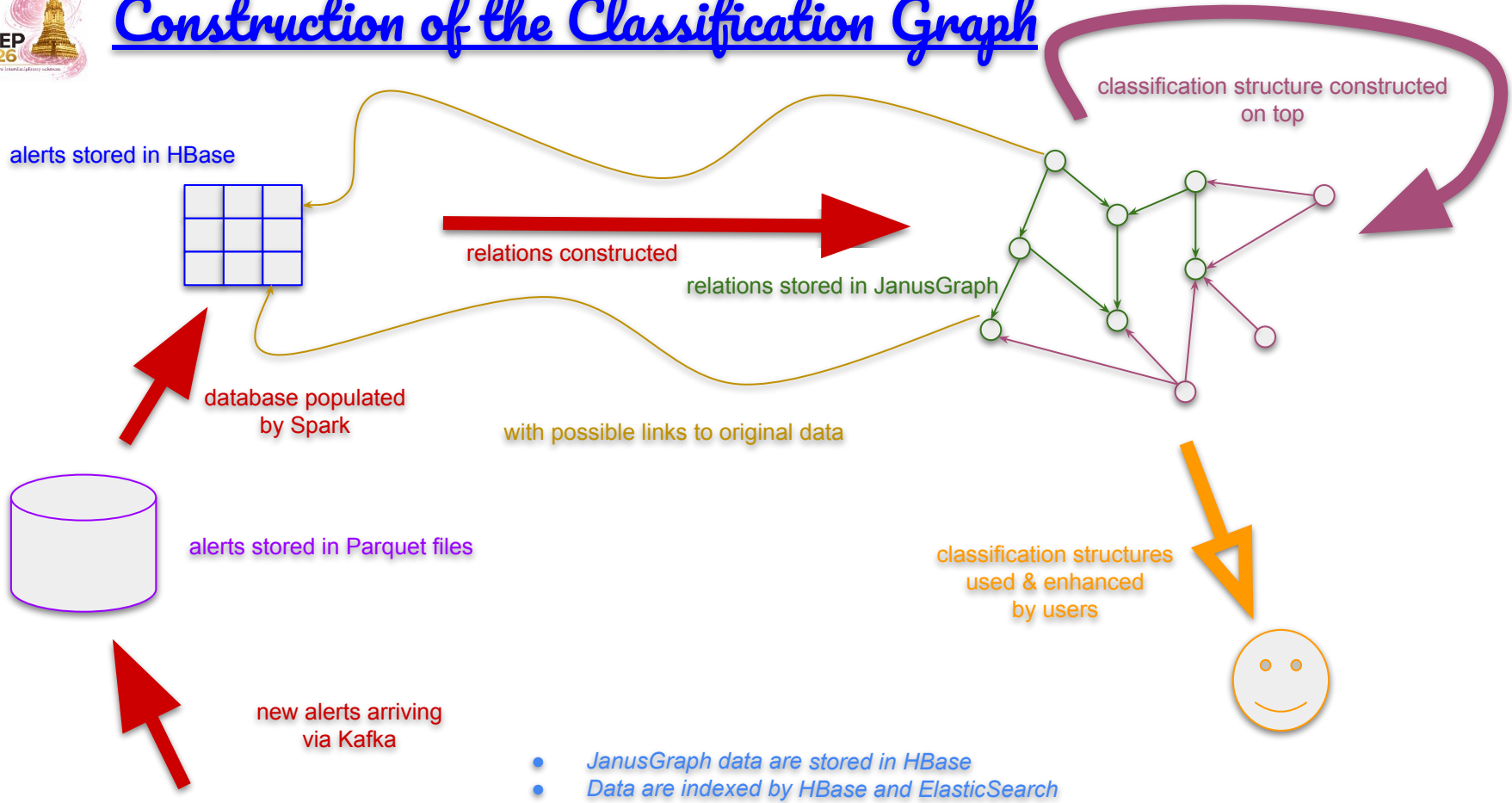
- Find overlaps (correlation) between A and B classes with respect to x and y objects
  - how to take into account Edge weights ?
  - how to include multiple-paths to the same x ?
  - should work for A == B
  - should work for oriented and unoriented Edges
- using **modified Flow graph algorithm**



classification classes (two different schemas)

classifies objects (alerts sources)

Construction of the Classification Graph



- *JanusGraph data are stored in HBase*
- *Data are indexed by HBase and ElasticSearch reverse indexes*

Classification Schema Examples

- Fink supports user-supplied classifiers/filters searching for particular types of alerts
  - Often using AI
- External classification source
  - When alert source is already known
- Manually tagged alerts (by users)

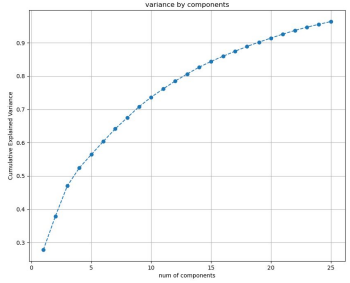
- **Features** - classification using PCA + Clustering on the alerts feature space
- **Light-Curves** - classification based on analysing alerts light-curves

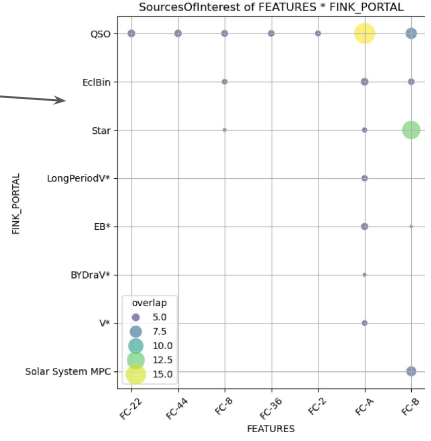
global classification based on alerts properties

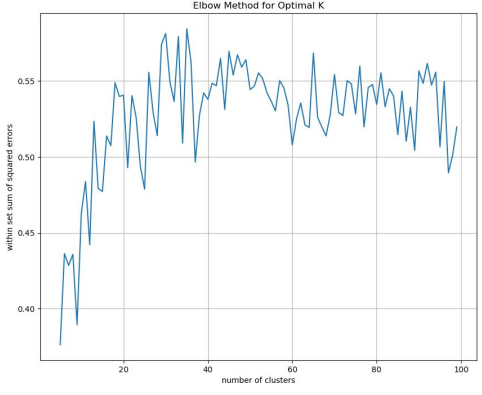
Features

- Preparing training data
  - Well classified alerts and classes with enough statistics
- **Principal Component Analysis**
  - choose number of pca parameters @ 80% variance = 13
- **KMeans Clustering**
  - choose number of clusters @ maximum silhouette = 30
- Output clustering model
- Creating FEATURES graph
- Generating overlaps
  - Merging similar classes

```
"mean",
"weighted_mean",
"standard_deviation",
"median",
"amplitude",
"beyond_1_std",
"cusum",
"inter_percentile_range_10",
"kurtosis",
"linear_trend",
"linear_trend_sigma",
"linear_trend_noise",
"linear_fit_slope",
"linear_fit_slope_sigma",
"linear_fit_reduced_chi2",
"magnitude_percentage_ratio_40_5",
"magnitude_percentage_ratio_20_10",
"maximum_slope",
"median_absolute_deviation",
"median_buffer_range_percentage_10",
"percent_amplitude",
"mean_variance",
"anderson_darling_normal",
"chi2",
"skew",
"stetson_K"
```

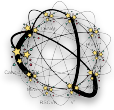


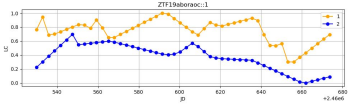


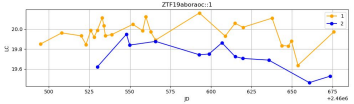


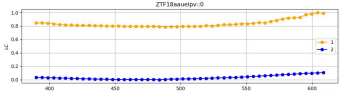
Light-Curves

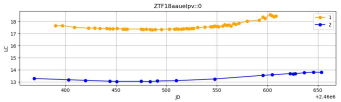
global classification based on alerts light-curves (from series of alerts)



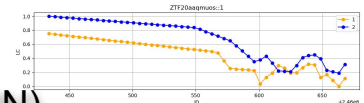


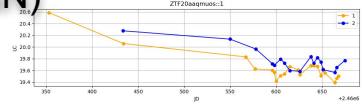


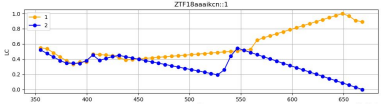


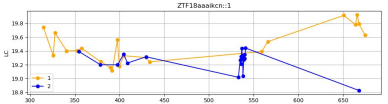


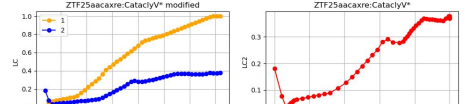
- Cleaning the curves
- Combining composite curves
  - using AI to find useful compositions
- Training **long short-term memory recurrent NN** (LSTM-RNN)

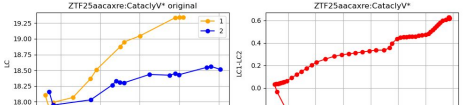


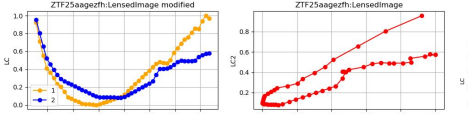


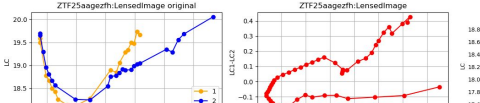


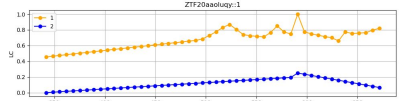


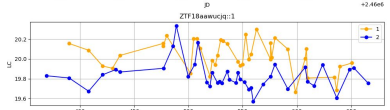


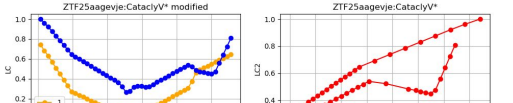


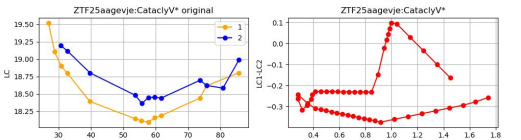


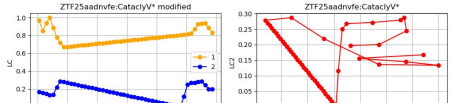


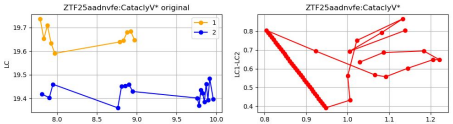


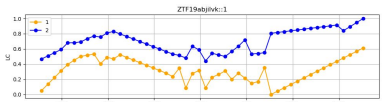


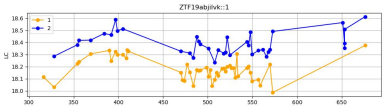












Usage - CLI

Neighborhood - ZTF

- Find ‘similar’ **alert sources**
  - With respect to a classification schema (‘FINK_PORTAL’)
  - Using certain metric (‘JensenShannon’)
- Once we get ‘similar’ sources, we can do more detailed comparison

```
// find all alerts the most similar to 'ZTF25aaqygm'
//   using 'FINK_PORTAL' classification schema
//   and 'JensenShannon' measure
gr.sourceNeighborhood('ZTF25aaqyygm', 'FINK_PORTAL', 10, 'JensenShannon', 0.0, false)

==>ZTF25aavlimb=0.002668906826373764={SN candidate=0.7857142857142857, Early SN Ia candidate=0.21428571428571427}
==>ZTF25aauhbec=0.015127377117961508={SN candidate=0.7647058823529411, Early SN Ia candidate=0.23529411764705882}
==>ZTF25aavfxlt=0.015132453397871537={SN candidate=0.8, Early SN Ia candidate=0.2}
==>ZTF25aalpqkg=0.027252196630868728={SN candidate=0.75, Early SN Ia candidate=0.25}
==>ZTF25aavozkg=0.027252196630868728={SN candidate=0.75, Early SN Ia candidate=0.25}
==>ZTF25aakbssn=0.04552313886652741={SN candidate=0.7272727272727273, Early SN Ia candidate=0.2727272727272727}
==>ZTF25aayxghj=0.12260324160315225={SN candidate=0.625, Early SN Ia candidate=0.375}
==>ZTF25aascfmn=0.12953628201346198={SN candidate=0.6153846153846154, Early SN Ia candidate=0.38461538461538464}
==>ZTF25aawjnuz=0.17896159576496104={SN candidate=0.5454545454545454, Early SN Ia candidate=0.45454545454545453}
==>ZTF25aatclux=0.17972561983947669={SN candidate=0.95, Early SN Ia candidate=0.05}
```

Re-classification - ZTF

- ‘FEATURES’ classification schema is able to reproduce (to some extent) classification of ‘FINK_PORTAL’ schema without directly using it (thanks to correlations recorded in graph)

```
// classification from ‘FINK_PORTAL’ classification
gr.classification("ZTF25aaksfzy", 'FINK_PORTAL')
==>[weight:1.0,classifier:FINK_PORTAL,class:Solar System candidate]

// classification from ‘FEATURES’ classification (PCA+Clustering)
gr.classification("ZTF25aaksfzy", 'FEATURES')
==>[weight:1.0,classifier:FEATURES,class:FC-3]

// classification from ‘FEATURES’ classification
//   interpreted by ‘FINK_PORTAL’ classification
//   using correlations between two classifications
gr.reclassification('ZTF25aaksfzy', 'FEATURES', 'FINK_PORTAL')
==>Solar System candidate=556.0
==>Tracklet=511.0
==>SN candidate=318.0
==>Early SN Ia candidate=11.0
==>Solar System MPC=8.0
==>Ambiguous=2.0
==>Kilonova candidate=2.0
==>Microlensing candidate=1.0
==>BLLac=1.0
```

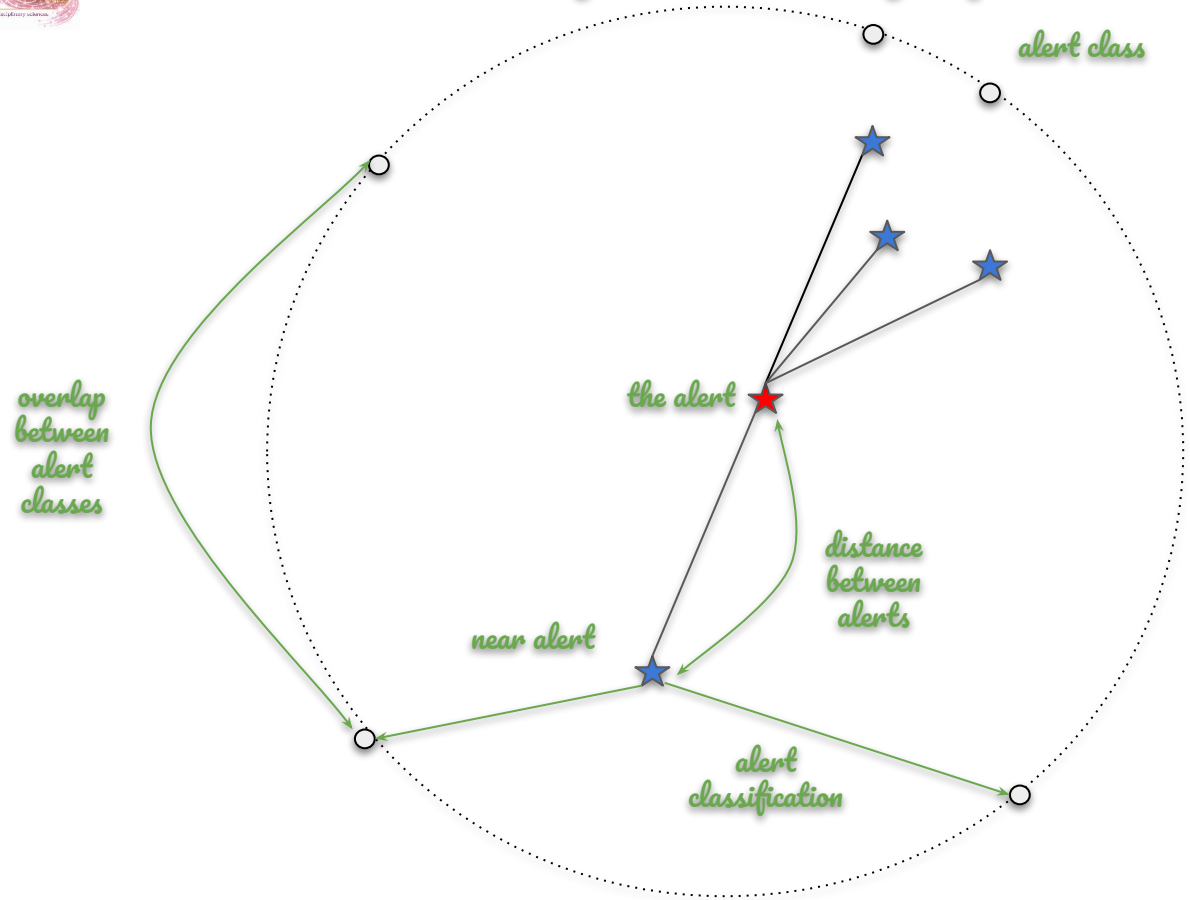
Re-classification - LSST

- Alerts sources can be tagged (annotated) and used for similarity searches and re-classification

```
// just take one object
oid = '313985349745377418'
// how it is classified
gr.classification(oid)
==>[weight:0.5714285714285714,classifier:FINK,flavor:,class:rubin.tag_early_snia_candidate]
==>[weight:0.42857142857142855,classifier:FINK,flavor:,class:rubin.tag_sn_near_galaxy_candidate]

// what are the most similar objects
gr.objectNeighborhood(oid, 'FINK', 10, 'JensenShannon')
==>313888627059851362=0.0={rubin.tag_early_snia_candidate=0.5714285714285714,
rubin.tag_sn_near_galaxy_candidate=0.42857142857142855}
==>313998539864670258=0.0={rubin.tag_early_snia_candidate=0.5714285714285714,
rubin.tag_sn_near_galaxy_candidate=0.42857142857142855}
==>313972153277481152=0.020508231493818984={rubin.tag_early_snia_candidate=0.6, rubin.tag_sn_near_galaxy_candidate=0.4}
…

// tag oid as 'TestTag' with weight=1.0 in 'TAG' classification schema
// (manual classification schema)
gr.registerOCol(Classifier.instance('TAG', 'LSST', ''), 'TestTag', oid, 1.0, '', '')

// how would be that oid classified in 'FINK'
// (it uses information about its classification in 'TAG'
//  and the correlations between classification in 'FINK' and 'TAG' schemas,
//  it doesn't use information how oid is actually classified in 'FINK')
gr.reclassification(oid, 'TAG', 'FINK', 10, true)
==>[classifier:FINK,flavor:,weight:0.5358983848622454,class:rubin.tag_early_snia_candidate]
==>[classifier:FINK,flavor:,weight:0.4641016151377546,class:rubin.tag_sn_near_galaxy_candidate]
```

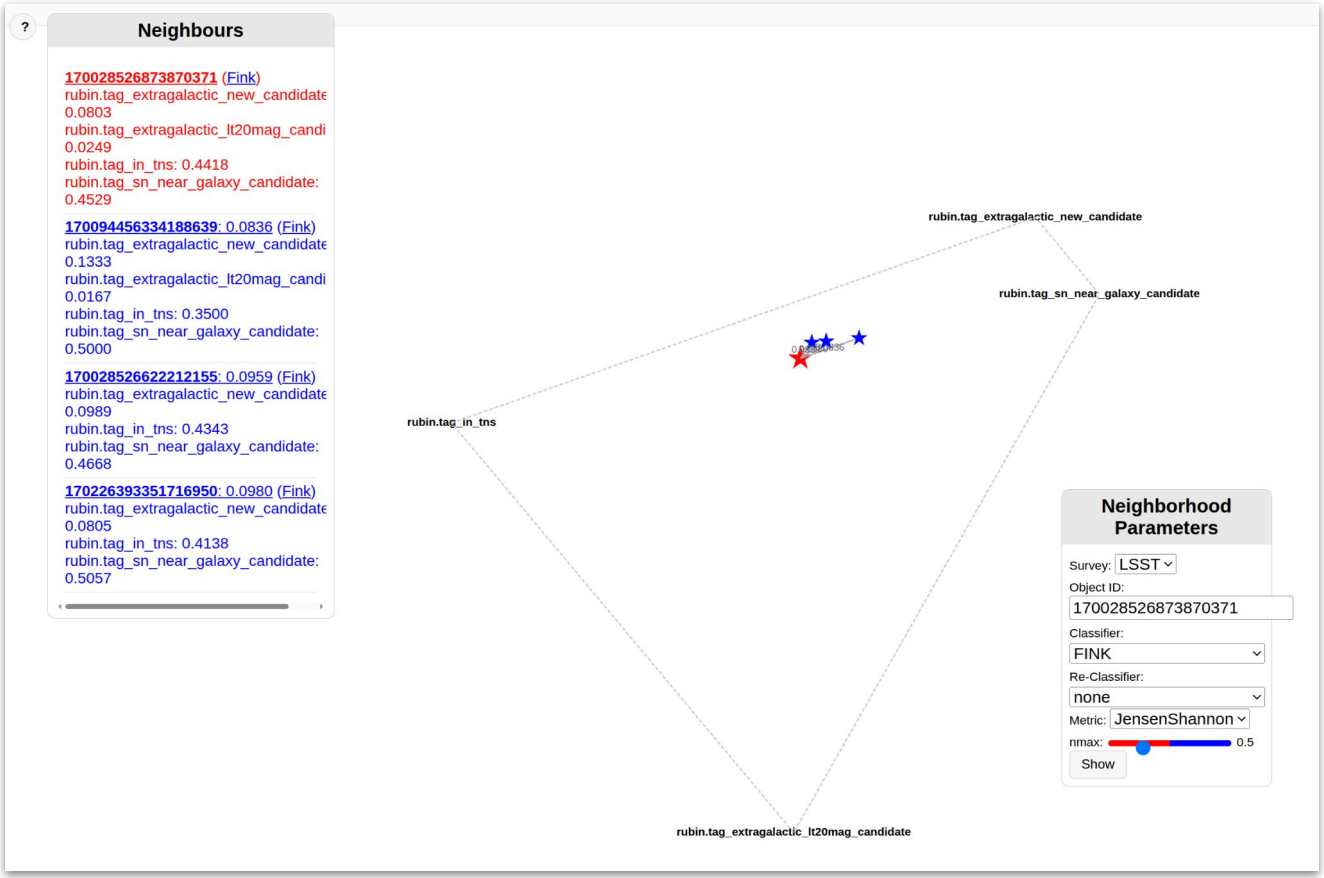
Usage - GUI/WS

Neighborhood Graphical View

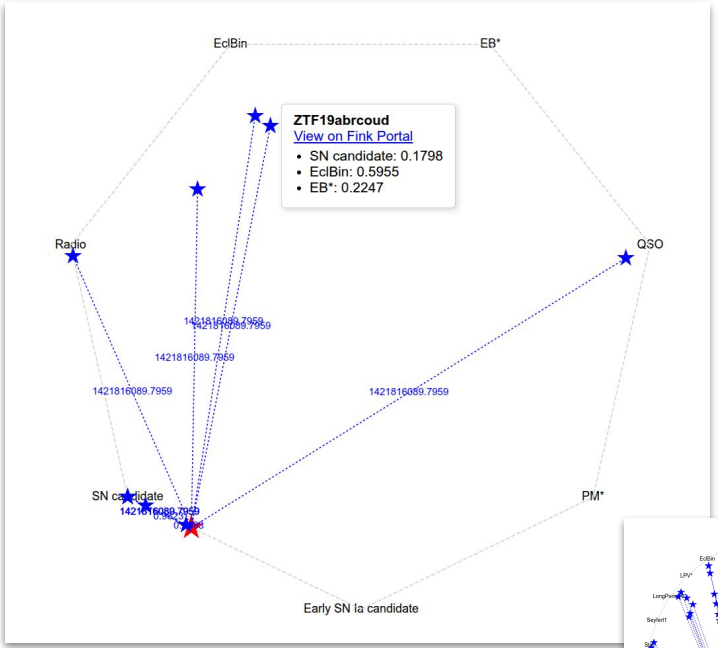


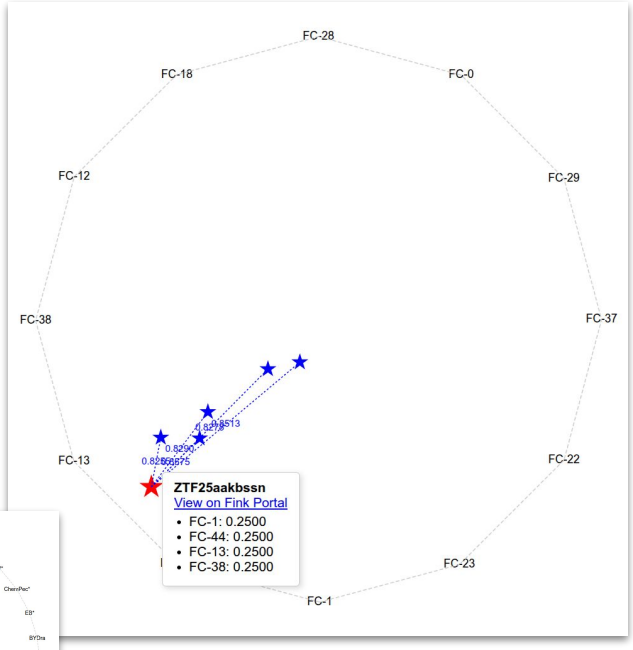
- The distance problem is multi-dimensional
- Its projection into 2-dim flat space is approximative, tuned to be intuitive
  - using AI
- It depends on
  - Chosen classification schema
  - Selected metric
- View is interactive
  - Hovering over an alert will show detailed information, with link to Fink Portal for that alert
  - Clicking on an alert will make this alert central and show its nearests alerts

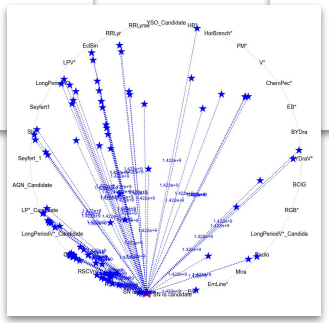
Graphical View Example - LSST

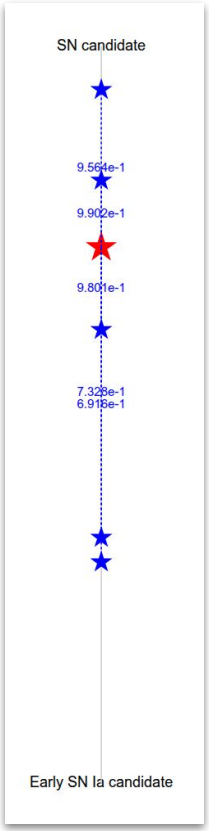


Graphical View Example - ZTF



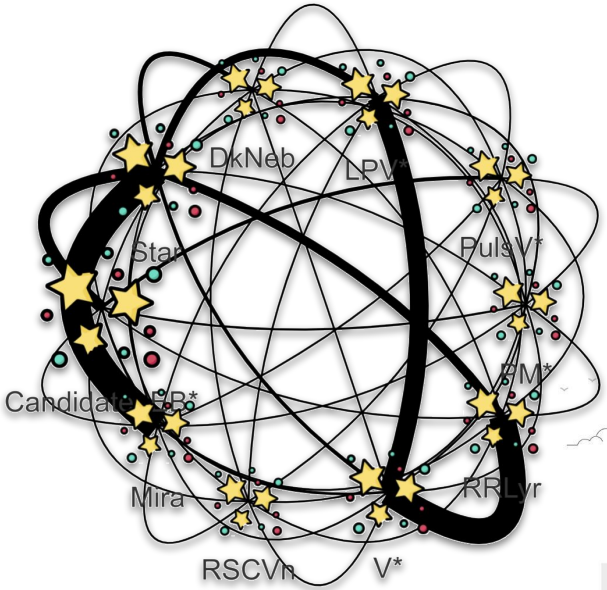






Summary

- Database system for exploitation of similarities between data elements.
- Flexible wrt **similarity definition and metric**.
- Implementation for astronomical alerts of ZTF/LSST observatories.
- Based on Lomikel toolset.
- Part of Fink Broker.



used technologies:
Spark, Kafka, Parquet, HDFS, HBase, JanusGraph, Python, Java, Scala, Groovy, Gremlin, JavaScript