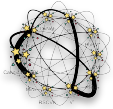
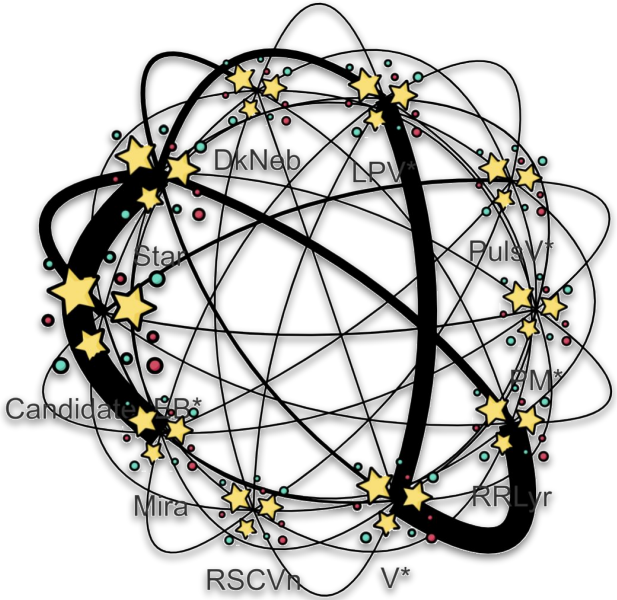
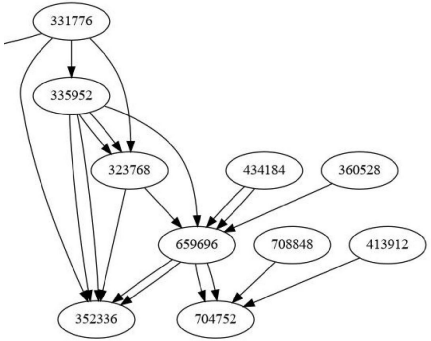


Multiclassification

Using multi-language & multi-database Applied to Fink Alerts



- **Multi-Classification**
  - Architecture
  - Classifiers
  - Applications
- **Technology**
  - Languages
  - Databases



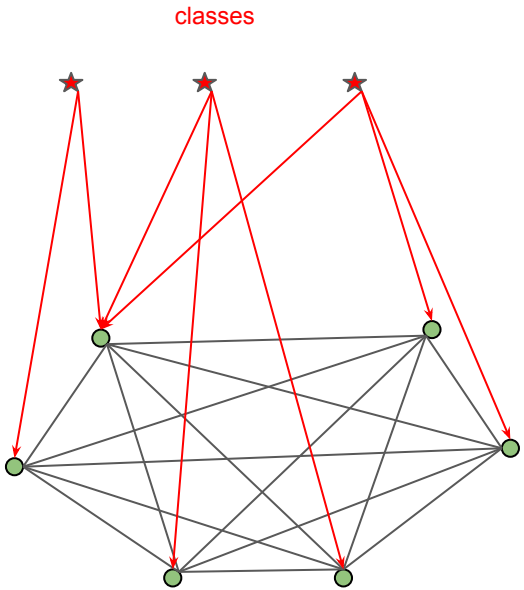
Julius Hrivnac, IJCLab
5/11/25

Project

- Using classification to organize objects (in a database)
- **Organizing classifications** of objects in structures (graphs) to allow (fast):
  - **Searching for objects classified in a certain way**
  - **Searching for ‘similar’ objects**
  - **Comparing different classifications schemas**
    - **Evaluating how similar they are**
    - **Interpreting classification with one schema by classes of another schema**
      - E.g. Interpreting AI based classification by user tags
- A kind of **Vector Database**
- Intuitive and interactive visualisation
- Applied to FINK alerts (ZTF, LSST)

Simple Classification

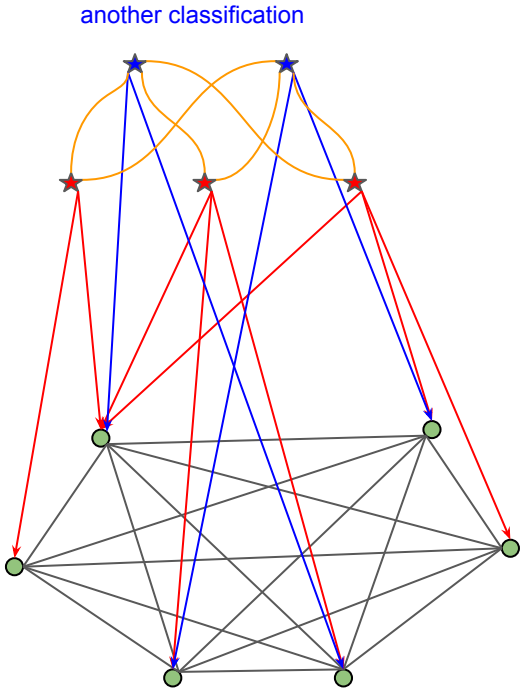
- Want to investigate relations (similarities) between objects
- We should define a **measure** (distance) to quantify objects similarity
  - Can have several different measures
- The measure gives objects relations
  - Without discrete classification, the problem is `n*n (n = number of objects)`, which is impossible 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
- Several objects types (having relations) can be classified in parallel
- Objects *Xyz* are classified via *Abc*Collection
    - All *Xyz* of a class belongs to that class *AbcCollection*
  - One object can belong to several classes (with respective **weights**)



objects with their relations

Multiple Classification

- Objects can be classified with several schemas, exploring different kind of similarities
- Classification schemas can have several ***flavours***
  - From different classification sources or hyperparams,...
- Classification schemas can be **combined** into *superschemas*
- Classification schemas can be **divided** into *subschemas*
- Once existing objects are classified, we can easily (and quickly):
  - Classify new objects
  - Find overlaps between classes in one schema or between schemas
  - Comparing truthfulness of schemas
  - Interpreting AI classes with Analytical classes



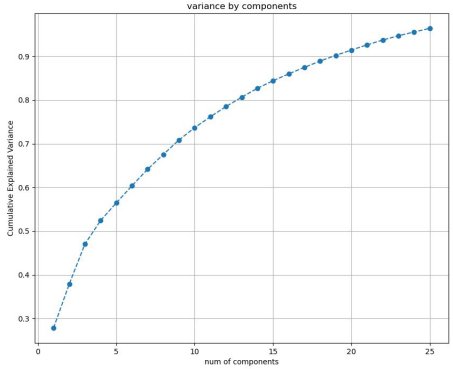
Classification Schemas

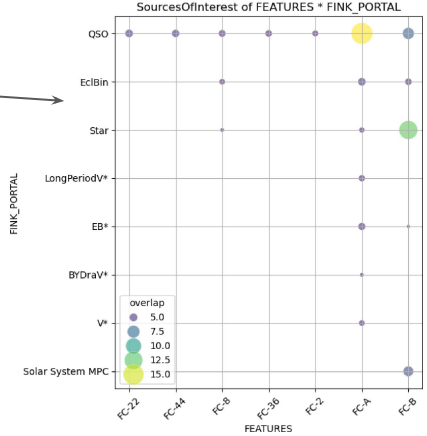
- Analytical: calculated for each object individually by an algorithm, or taken from third services
  - Fink Portal
  - External services
- AI: calculated by trained network
  - **Features**
  - **Light Curves**
- **Tags**: supplied directly by users

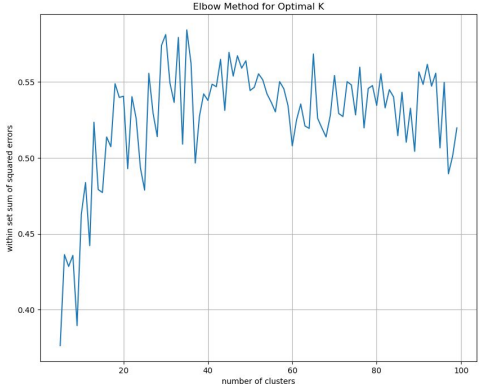
Features

PCA Clustering of alerts

- Preparing training data
  - Well classified alerts and classes with enough statistics
- PCA
  - 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
- All new alerts are (re)classified as they are arriving
- Reasonable results for some classes
- So far very naive & brute force
  - Just taking all features and existing classes
  - Various selections of training sources





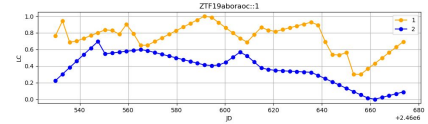


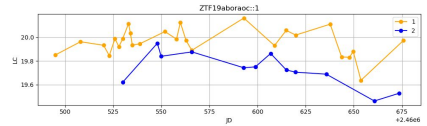
Light Curves

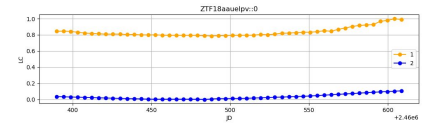
- Clean the curves
  - using naive brute force
- Combine composite curves
  - can use AI to find useful compositions
- Randomise and split data: 75% for training, 25% for testing
- Configure long short-term memory recurrent NN (LSTM-RNN)
  - Keras LSTM by dl4j
- Train & test
- So far very naive & brute force
  - Just selecting non-trivial curves
  - Or their combinations

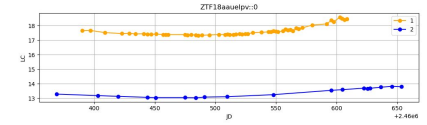
Light Curves

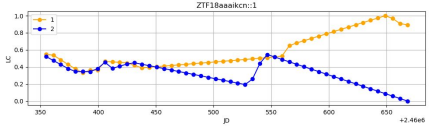
cleaning

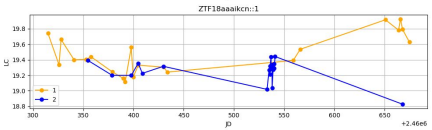


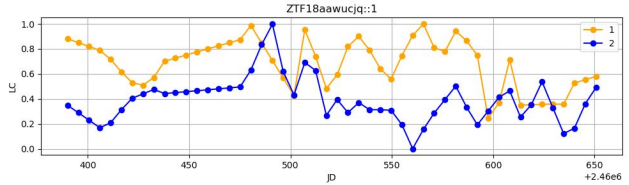


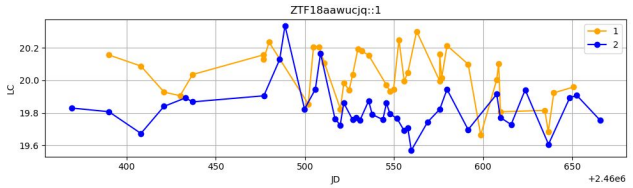


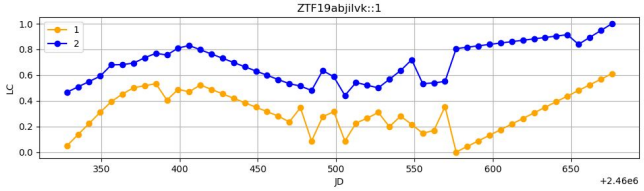


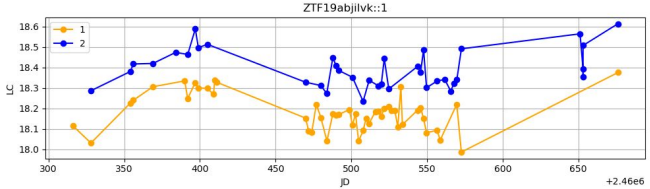


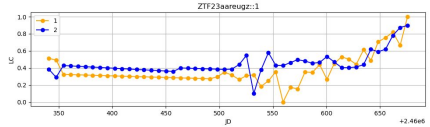


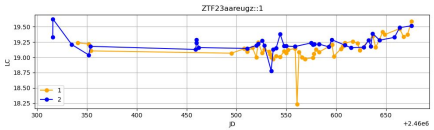


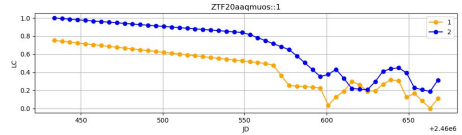


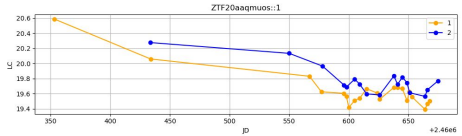


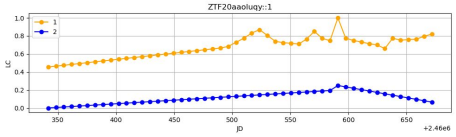


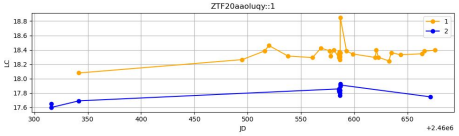






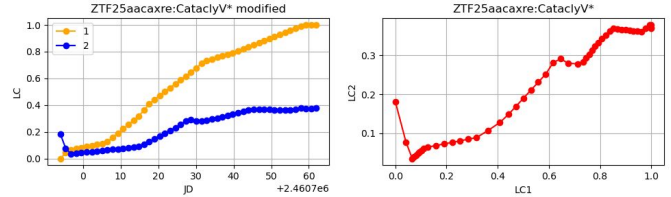


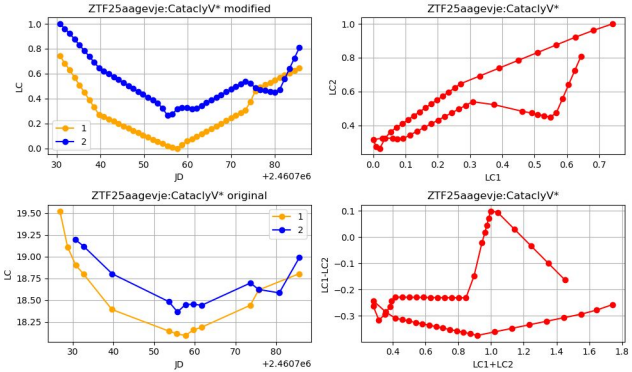


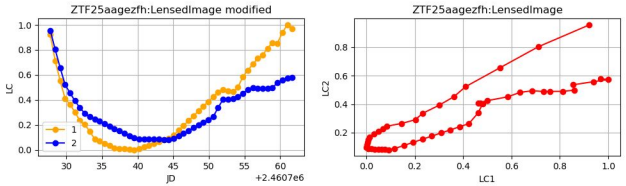


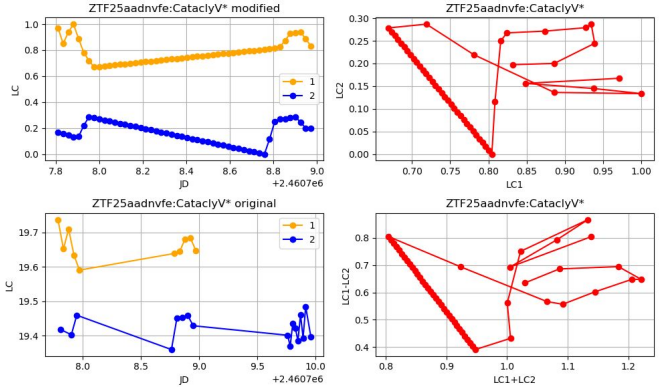
Light Curves

cleaning & composing (PCA-like)









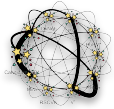
Light Curves

training

```
conf = new NeuralNetConfiguration.Builder()
                                 .seed(123)
                                 .weightInit(WeightInit.XAVIER)
                                 .updater(new Adam(0.002))
                                 .gradientNormalization(GradientNormalization.ClipElementWiseAbsoluteValue)
                                 .gradientNormalizationThreshold(0.5)
                                 .list()
                                 .layer(new LSTM.Builder()
                                              .activation(Activation.TANH)
                                              .nIn(1)
                                              .nOut(20)
                                              .build())
                                 .layer(new RnnOutputLayer.Builder(LossFunctions.LossFunction.MSE)
                                                          .activation(Activation.SOFTMAX)
                                                          .nIn(20)
                                                          .nOut(numLabelClasses)
                                                          .build())
                                 .build();
```

Tags

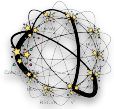
- To give user a possibility to tag ‘interesting’ sources
  - To remember them
  - To find similar sources (using multiple classification)
- Creation of Tags (i.e. collection of tagged objects) may require confirmation
  - To prevent multiplication of identical Tags
- We can start from a set of pre-defined Tags
- Actual tagging could be open to anyone (it will be signed anyway)
- Tags can contain further annotation
  - With external references
- Fink Portal will show tags for each object.
- There will be also a page with overview of existing tags with s possibility to search.
- We may create an alert filter based on Tags + Classification
  - ***Send me all new alerts with are classified in the same way as alerts tagged by me as X ! How are they tagged by others ?***



Exploring Classification Graphs

- Neighborhood
  - Searching similar objects
- Re-classification
  - Interpreting one classification schema with another one
- Overlaps
  - Between classes of objects

Neighborhood



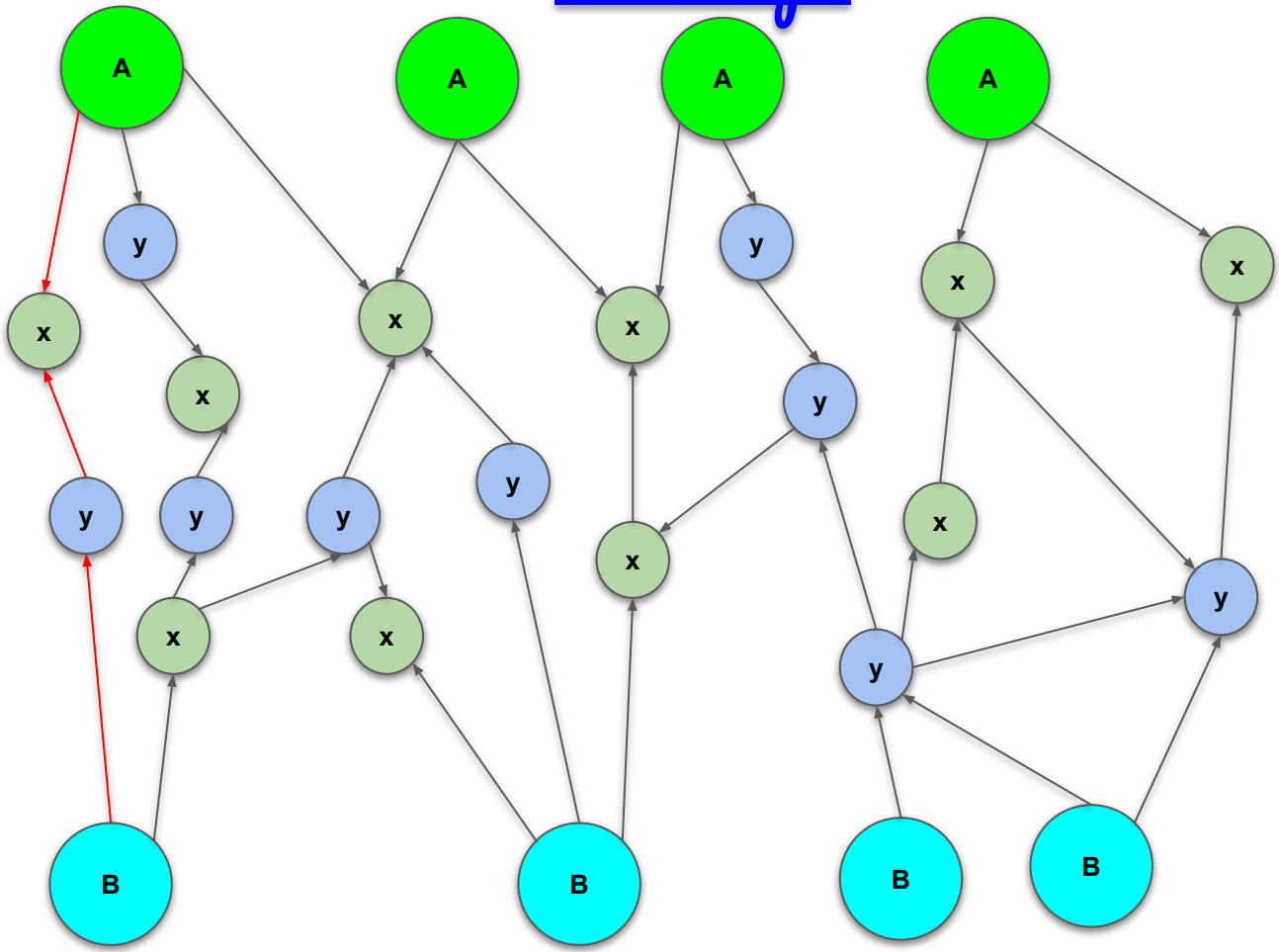
- Searching for ‘similar’ objects
- With respect to a classification schema
- And using certain measure
  - Jensen-Shannon
  - Euclidean
  - Cosine
- Once we get ‘similar’ objects, we can do more detailed comparison

```
gr.sourceNeighborhood('ZTF25aaqyygm', 'FINK', 10, 'JensenShannon', 0.0, false)

  -   calculating source distances
      from ZTF25aaqyygm[SN candidate:0.782608695652174, Early SN Ia candidate:0.21739130434782608]
      using [nmax:10.0, metric:JensenShannon, climit:0.0, allClasses: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}
```

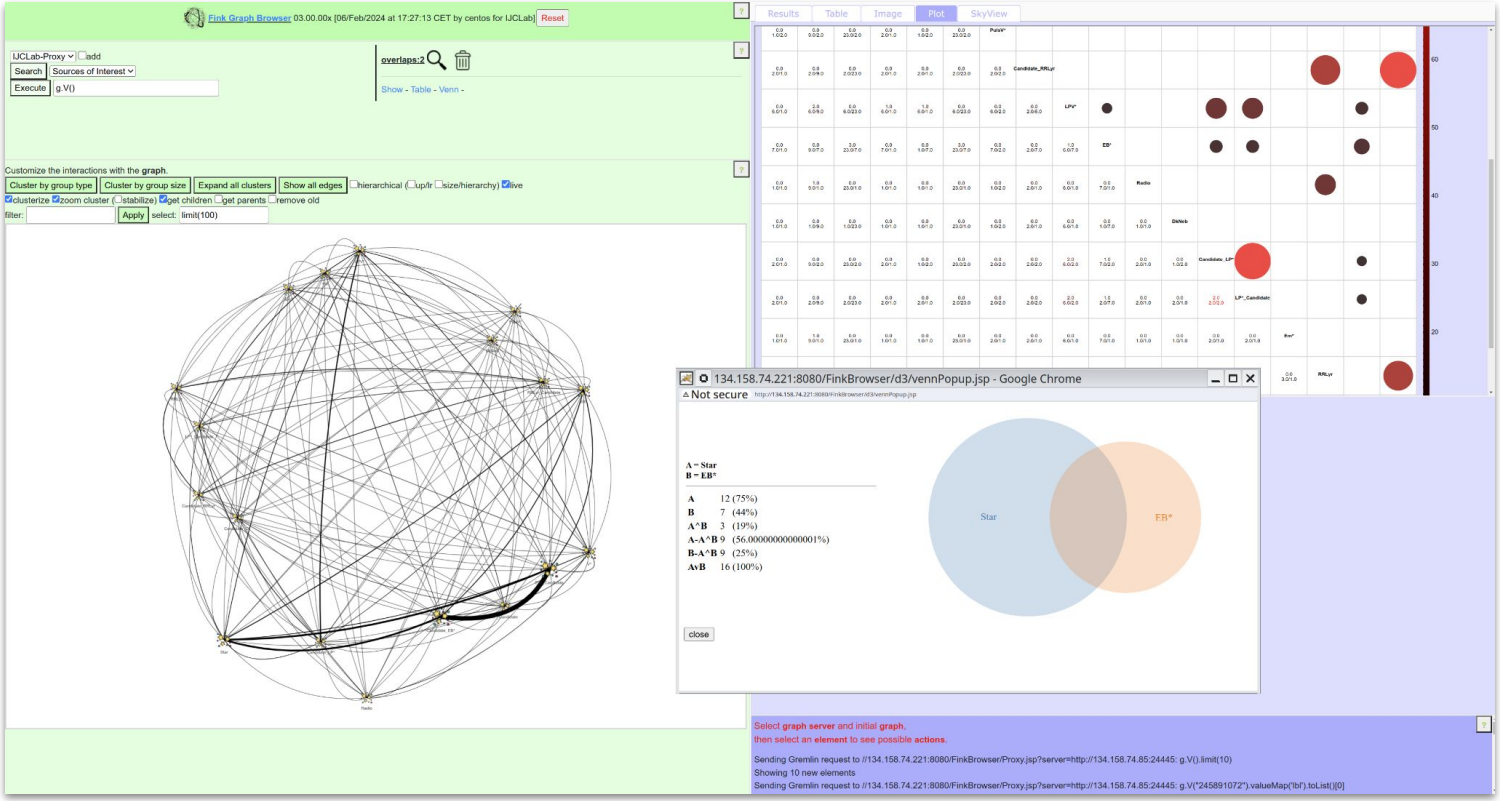
Overlaps



- Finding overlaps between A and B classes with respect to x or y objects
- ‘Similarity’ of classes
- Similar to ‘flow’ algorithm

Overlaps

- Within one classification schema
- Or between two classifications



Re-classification

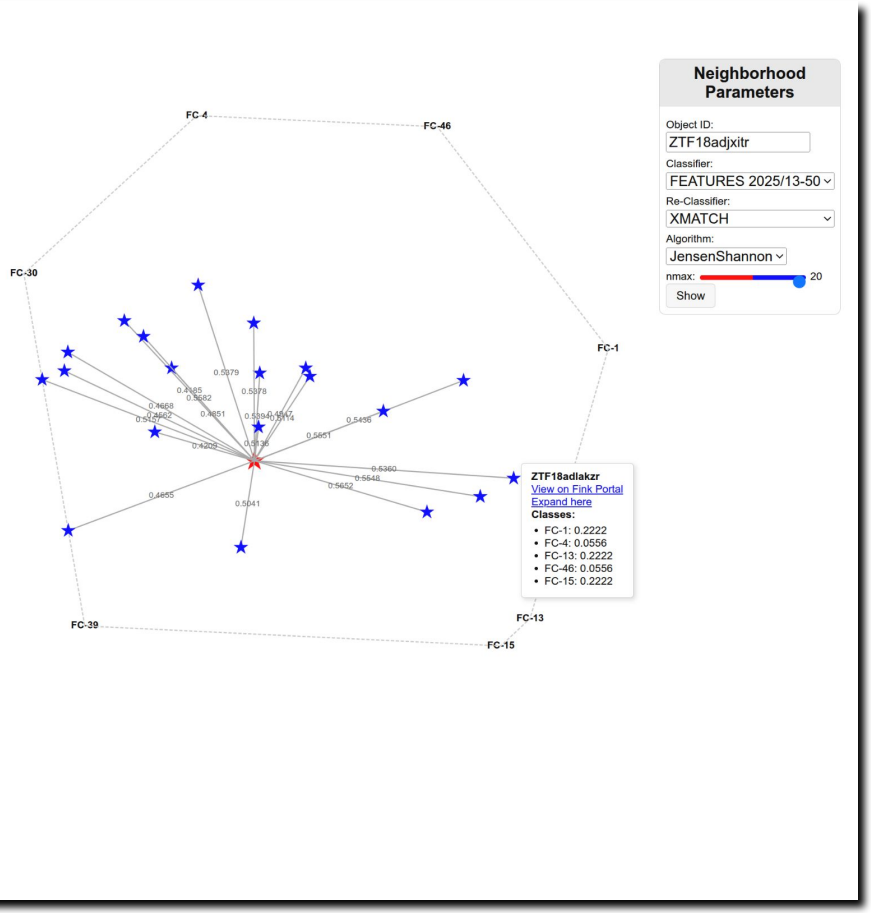
- One classification schema is able to reproduce (to some extent) classification of another schema without directly using it
  - for some classes
  - for groups of classes
    - can try to use AI to find good candidates for classes groups

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

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

// classification from FEATURES interpreted by FINK classification
//   using correlations between two classifications
gr.reclassification('ZTF25aaksfzy', 'FEATURES', 'FINK')
==>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
```

Visual Search



- Distances between classes correspond to their overlap
- Distances between objects (stars) correspond to their similarities
- Distances between objects and classes correspond to objects classification
- Projecting n-dim space into 2-dim plane => approximative distances

Multi-language & Multi-database

Languages

- **Java**
  - Most of the core
- **Groovy**
  - Scripting, application routines
  - Completely interoperable with Java (Java scripting extension)
- **Gremlin**
  - Access and algorithms on Graph databases
  - Groovy extension
- **Python**
  - API to Fink
  - Transparent availability via Py4J access server
- **Scala**
  - Some functional (Spark) access to databases and file
  - Completely interoperable with Java
- **JavaScript**
  - Web service graphical interface
  - Interfaced via JSP
  - Written with the help of AI
- **Native code** (C/C++)
  - Indirectly (via DL4J) to access optimised DL (TensorFlow)
- **Ant, Maven, Gradle**
  - Java - Groovy based build, configuration and deployment

```
#!/usr/bin/env groovy
// converting SQL into XML with Groovy
// either run as a shell script or compiled
// ------------------------------------------------------
sql = Sql.newInstance("jdbc:mysql://localhost/Tuples",
                      "org.gjt.mm.mysql.Driver")
xml = new MarkupBuilder(new File("Tuples.xml"))
xml.tagSet() {
  sql.eachRow("select * from tuple where run > 2") {
    row -> xml.tag(Run:row.run, Event:row.event)
    }
  }
```

```
g.V(object0).inE().
             as('e').
             filter(and(outV().values('classifier').is(eq(cf[0])),
                        outV().values('flavor'    ).is(eq(cf[1])),
                        outV().values('cls'       ).is(within(classes0)))).
             project('cls', 'w').
             by(select('e').outV().values('cls')).
             by(select('e').values('weight')).
             each {it -> m0[it['cls']] = it['w']}
```

```
from py4j.java_gateway import JavaGateway
gateway = JavaGateway()
jc = gateway.jvm.com.Lomikel.Januser.JanusClient("/blabla/IJCLab.properties");
gr = gateway.jvm.com.astrolabsoftware.FinkBrowser.Januser.FinkGremlinRecipiesG(jc);
results = gr.sourceNeighborhood('ZTF25aaqyygm', 'FINK', 0.5)
```

```
@GrabResolver(name='hrivnac', root='https://raw.githubusercontent.com/hrivnac/Maven/main/')
@GrabResolver(name='central', root='https://repo1.maven.org/maven2/')
@Grab(group='com.hrivnac', module='Lomikel', version='03.08.00')
```

Databases

- Hadoop/**HBase**
  - 3D celular database
    - Unit: cell = row * column * timestamp
    - Core of the Fink data
- **Parquet** files
  - Source data (used to DL)
- **Janus Graph**
  - Classification graphs
  - Gremlin (functional) API
  - Stores data in HBase, indices in Elastic Search
- **Elastic Search**
  - Indexes
- **Vector DB**
  - I.e. approximative search (neighborhood)
  - **Effectively implementing proper Vector DB itself**
  - No ‘standard’ API exists (yet)