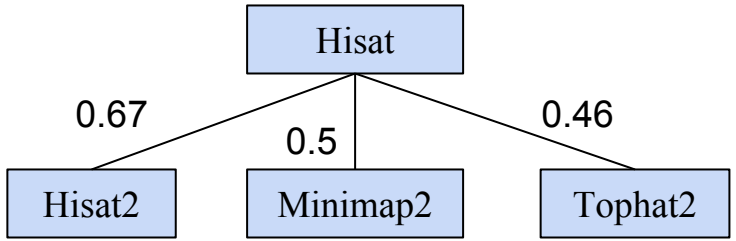
Recommendation system for scientific tools and workflows

Anup Kumar

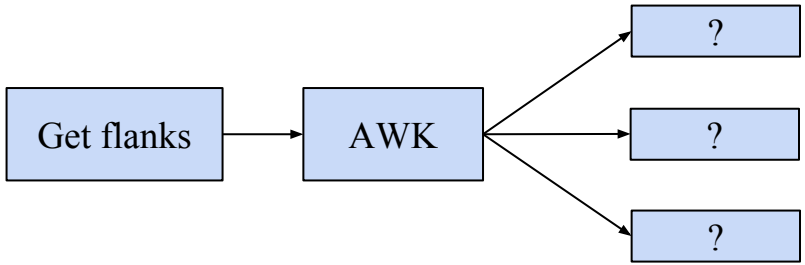
Adviser: Dr. Björn Grüning

Recommendation system

- Find similar scientific tools



- Predict tools for workflows



Find similar scientific tools

- Compute similarity among tools
- Recommend similar tools
- Replace a tool by its similar tool
- Provide more options for data processing

Approach

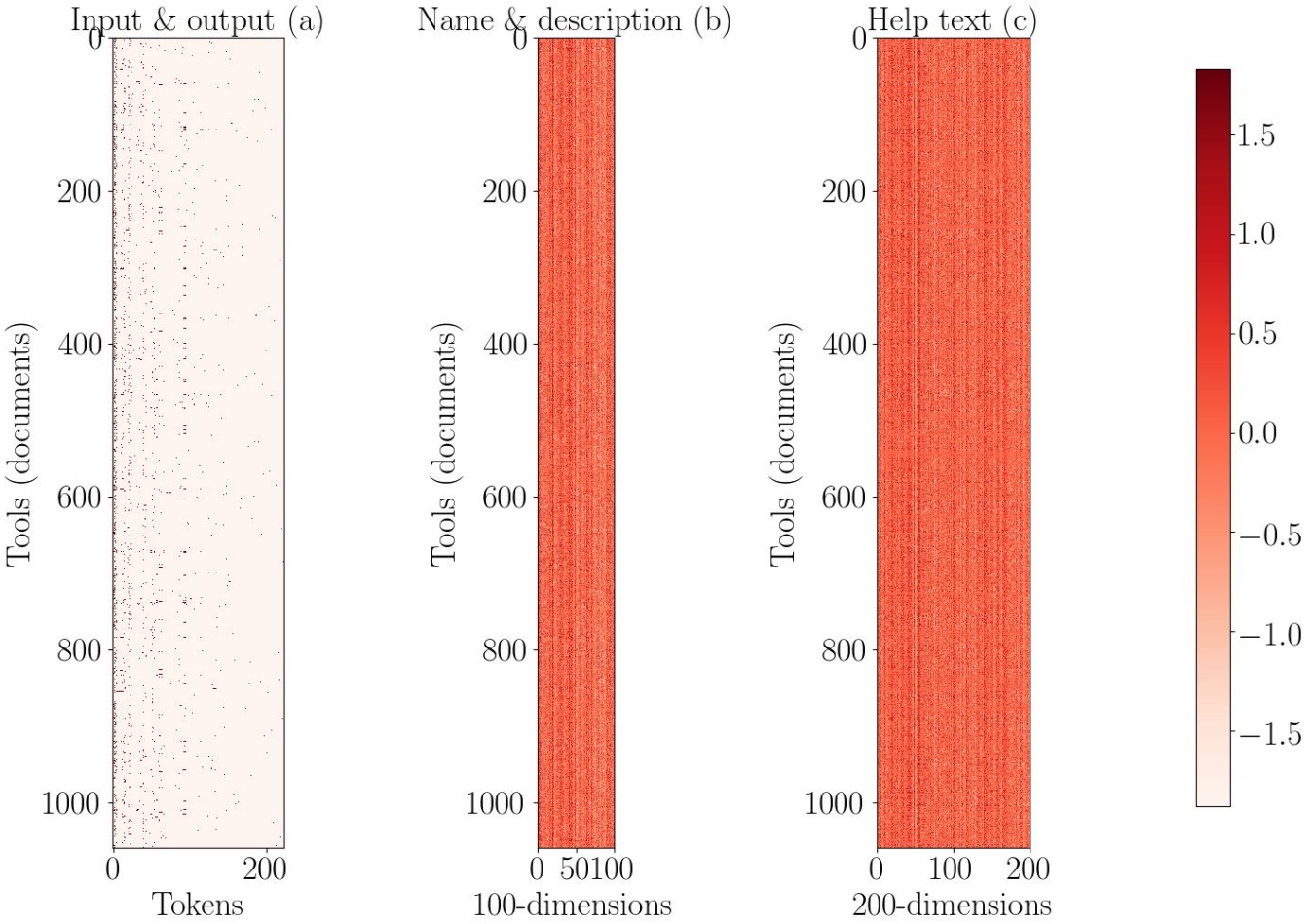
- Collect tools metadata
- Three attributes - input and output files, name and description and help text
- Clean metadata by stemming
- Learn vectors for tools (tool-token matrix)

Tools/Tokens	Regress	Linear	Gap	Mapper	Perform
LinearRegression	5.22	4.1	0.0	0.0	3.84
LogisticRegression	3.54	0.0	0.0	0.0	2.61
Tophat2	0.0	0.0	1.47	1.47	0.0
Hisat	0.0	0.0	0.0	0.0	0.0

Tool vector

1. Latent Semantic Analysis
    - Compute tool-token matrix using bestmatch25
    - Reduce ranks
2. Paragraph vector
    - Neural network approach
    - Similar tools have similar vectors

Document-token and paragraph matrices



Similarity scores

- Jaccard index (input and output files)
- Cosine angle (name and description and help text)
- Compute similarity matrix
- Three similarity matrices

Optimisation

- Learn weights on similarity scores

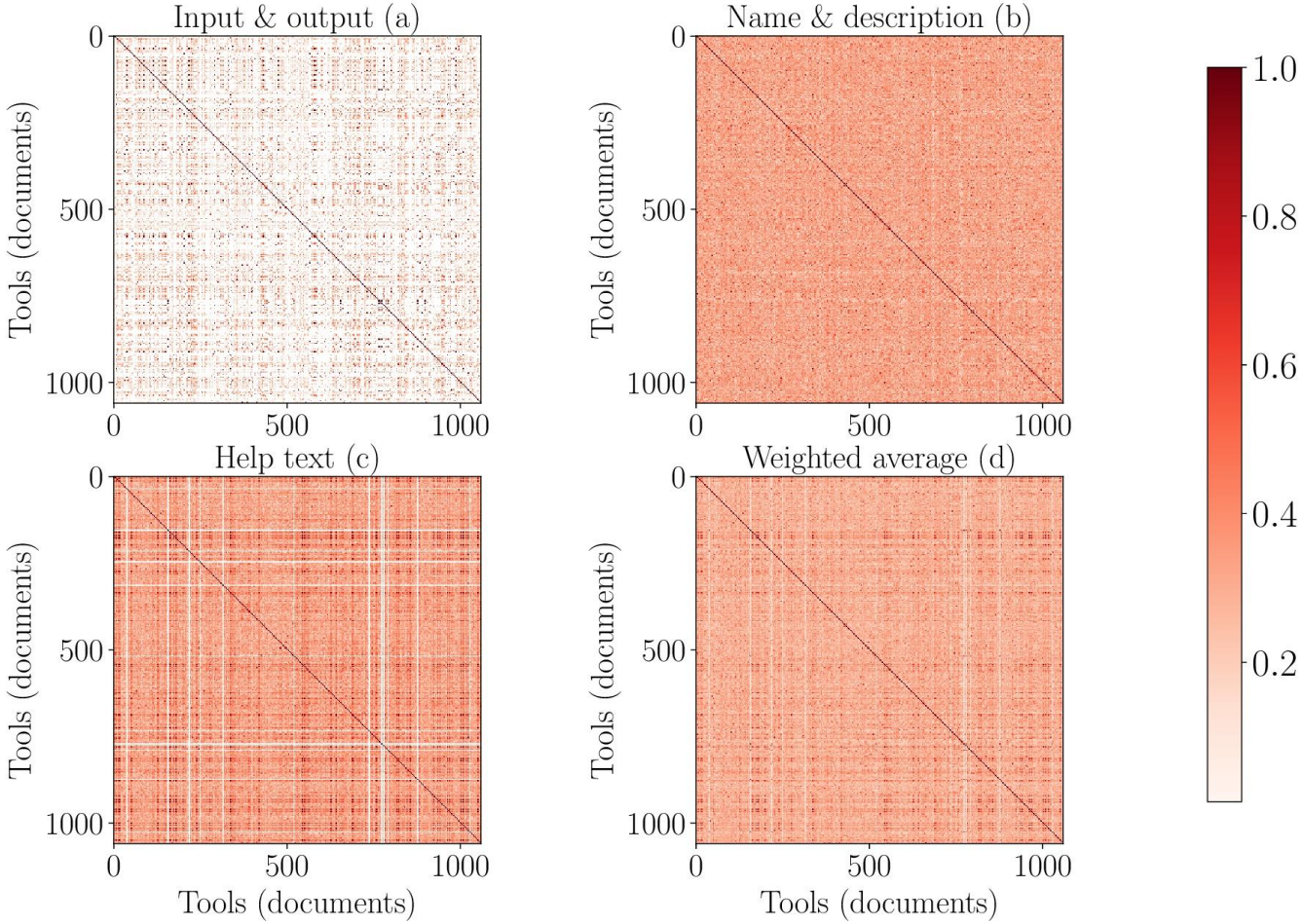
$$Error(w^k) = \frac{1}{N} \times \sum_{j=1}^{N} [w^k \times SM^k - SM_{ideal})^2]_j$$

$$Gradient(w^k) = \frac{\partial Error}{\partial w^k} = \frac{2}{N} \times ((w^k \times SM^k - SM_{ideal}) \cdot SM^k)$$

$$w^k = w^k - \eta \times Gradient(w^k)$$

- Obtain a weighted average similarity matrix

Similarity matrices



Summary

- Collect tools metadata
- Clean metadata
- Learn vector for each tool
- Get similarity matrix by applying similarity measures on vectors
- Combine similarity matrices using optimisation

Conclusion and future work

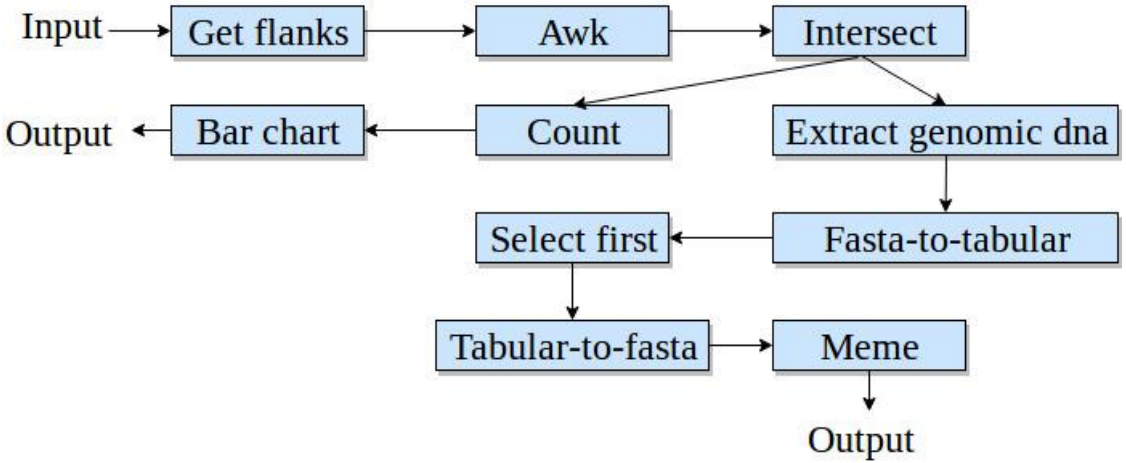
- Paragraph vector better than latent semantic analysis
- No true similarity
- Less data for name and description attribute

Predict tools in workflows

- Tough to assemble workflows
- Tools compatibility issues
- Loss of computation time
- Thousands of tools

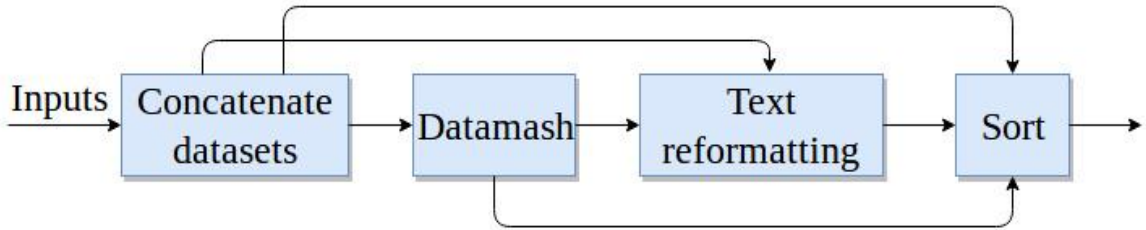
Workflow

- Workflow - directed acyclic graph
- Paths in workflow



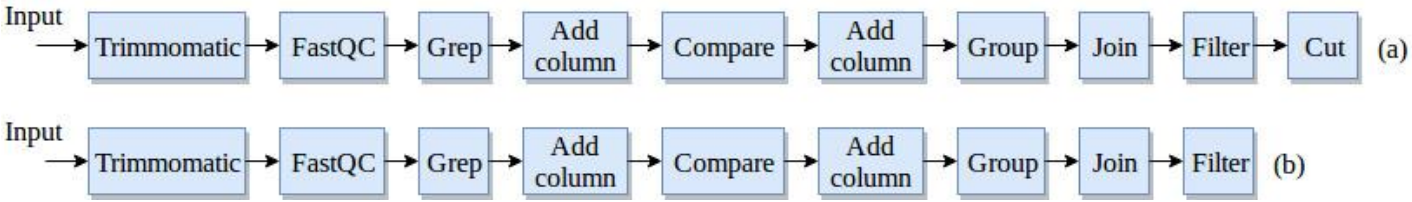
Higher order dependency

- Not dependent only on immediate parent
- Dependent on all previous tools



Workflow preprocessing

- No decomposition



Workflow preprocessing

- Keep first tool fixed

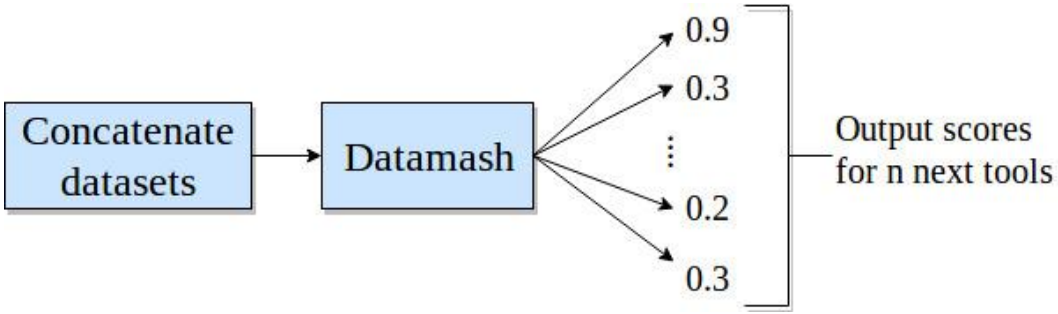
Recurrent neural network

- Need a classifier to consume sequential data
- Multilabel, multiclass classification
- Recurrent neural network - Gated recurrent units
- Learn long-range dependencies

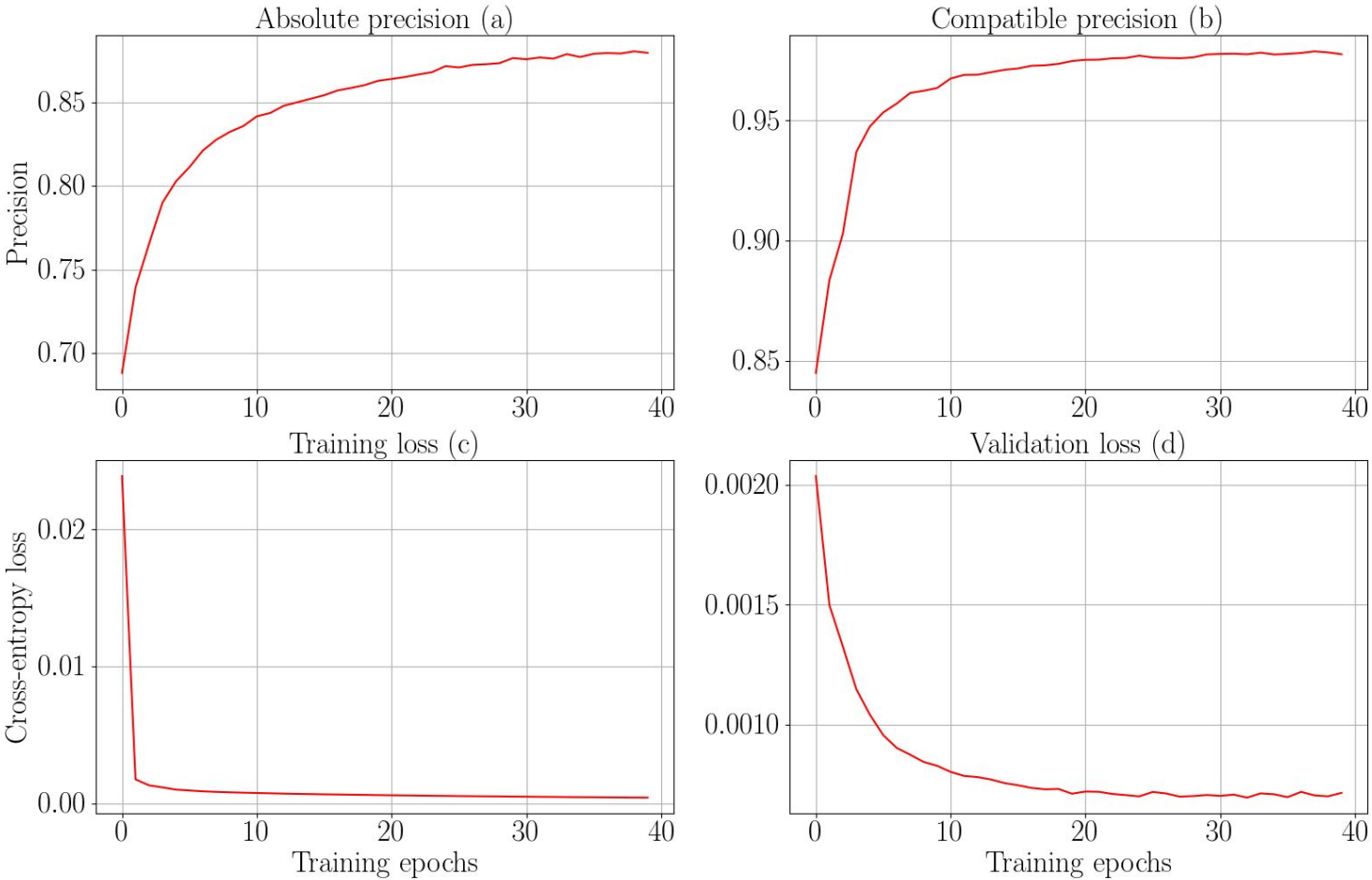
$$h_t = (1 - z_t) \times h_{t-1} + z_t \times h_t^{'}$$

Precision and prediction

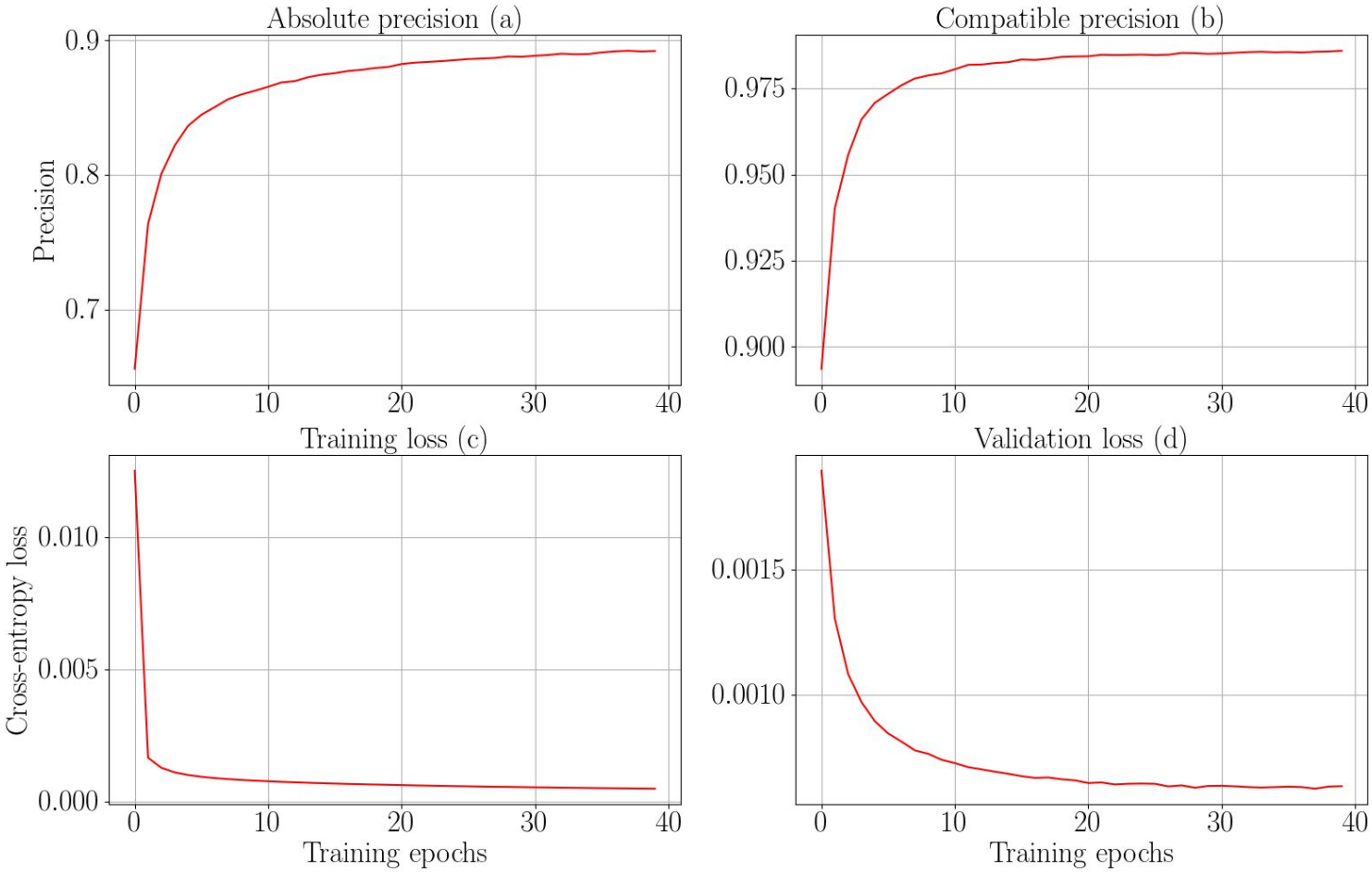
- Absolute precision
- Compatible precision

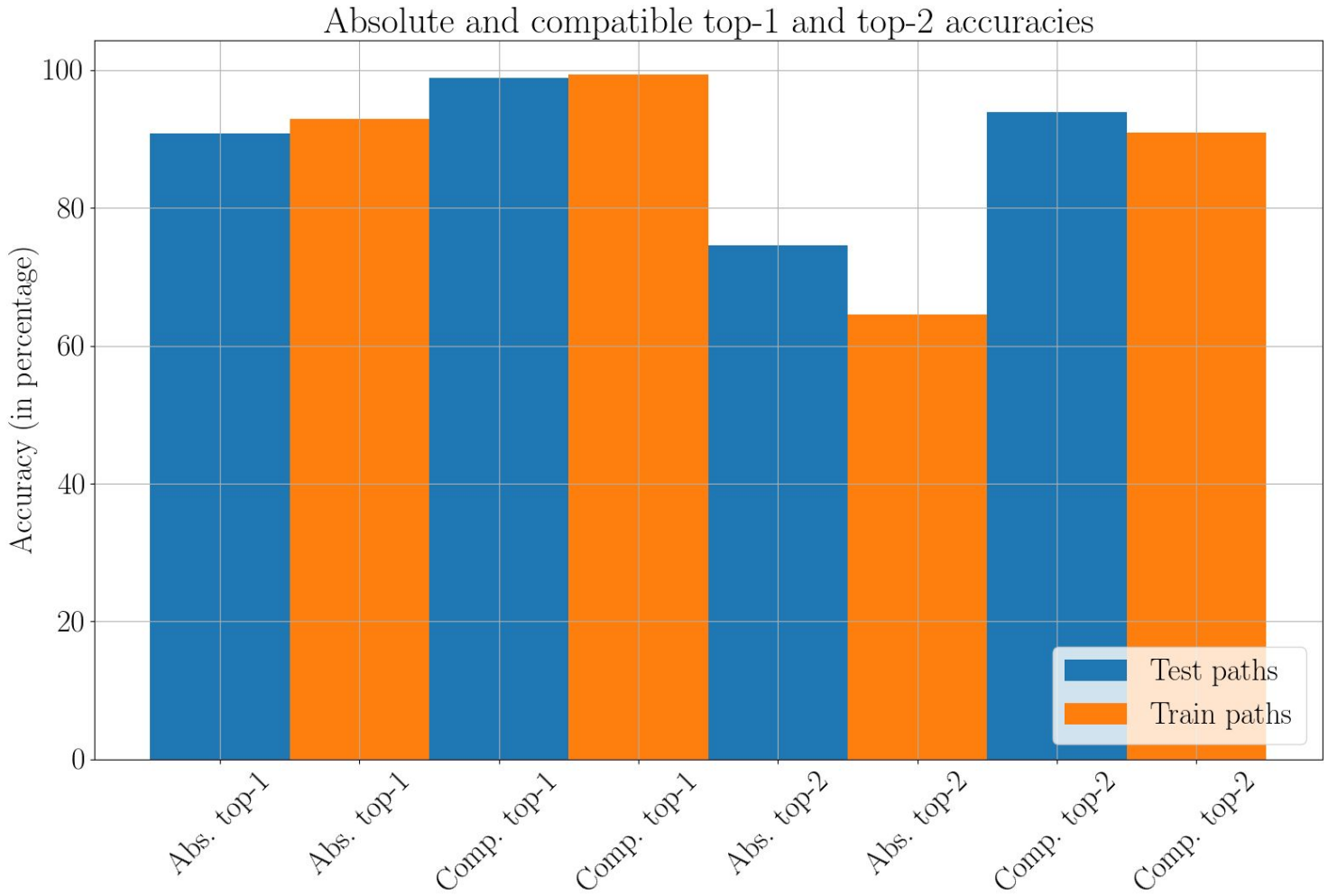


Precision and loss for no decomposition of paths



Precision and loss for decomposition of train and test paths




Absolute and compatible top-1 and top-2 accuracies
Accuracy (in percentage)
0
20
40
60
80
100
Abs. top-1
Abs. top-1
Comp. top-1
Comp. top-1
Abs. top-2
Abs. top-2
Comp. top-2
Comp. top-2
Test paths
Train paths

Summary

- Workflows are directed acyclic graphs
- Extract paths from workflows
- Learn long-range dependencies in these paths
- Use recurrent neural network (gated recurrent units) as a classifier
- Multilabel, multiclass classification
- Absolute and compatible precision

Conclusion and future work

- Restore original distribution
- Decay prediction over time
- Integrate into Galaxy

Thank you for your attention

Questions?