Recommendation system for scientific tools and workflows

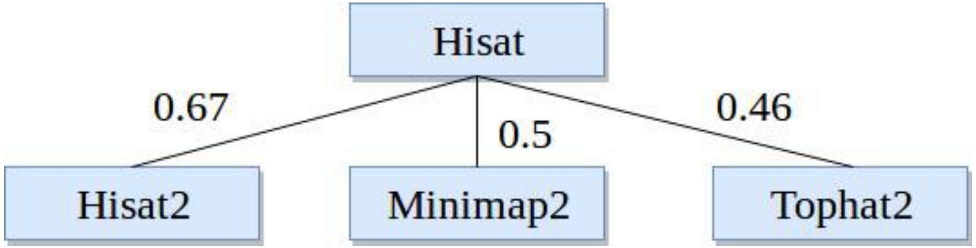
Master thesis

Anup Kumar

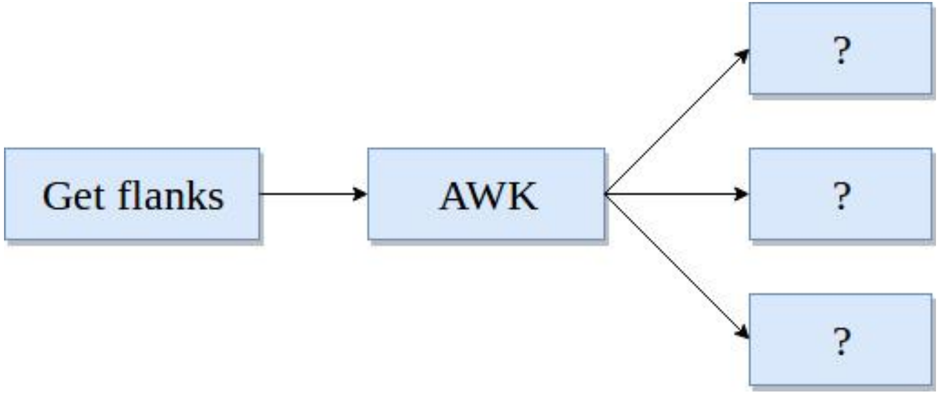
Adviser: Dr. Björn Grüning

Recommendation system

1. Find similar scientific tools



2. Predict tools for workflows



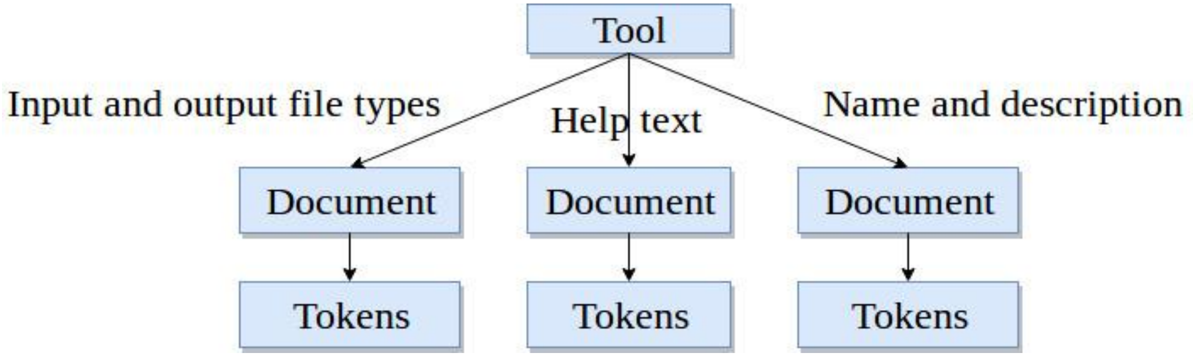
Find similar scientific tools

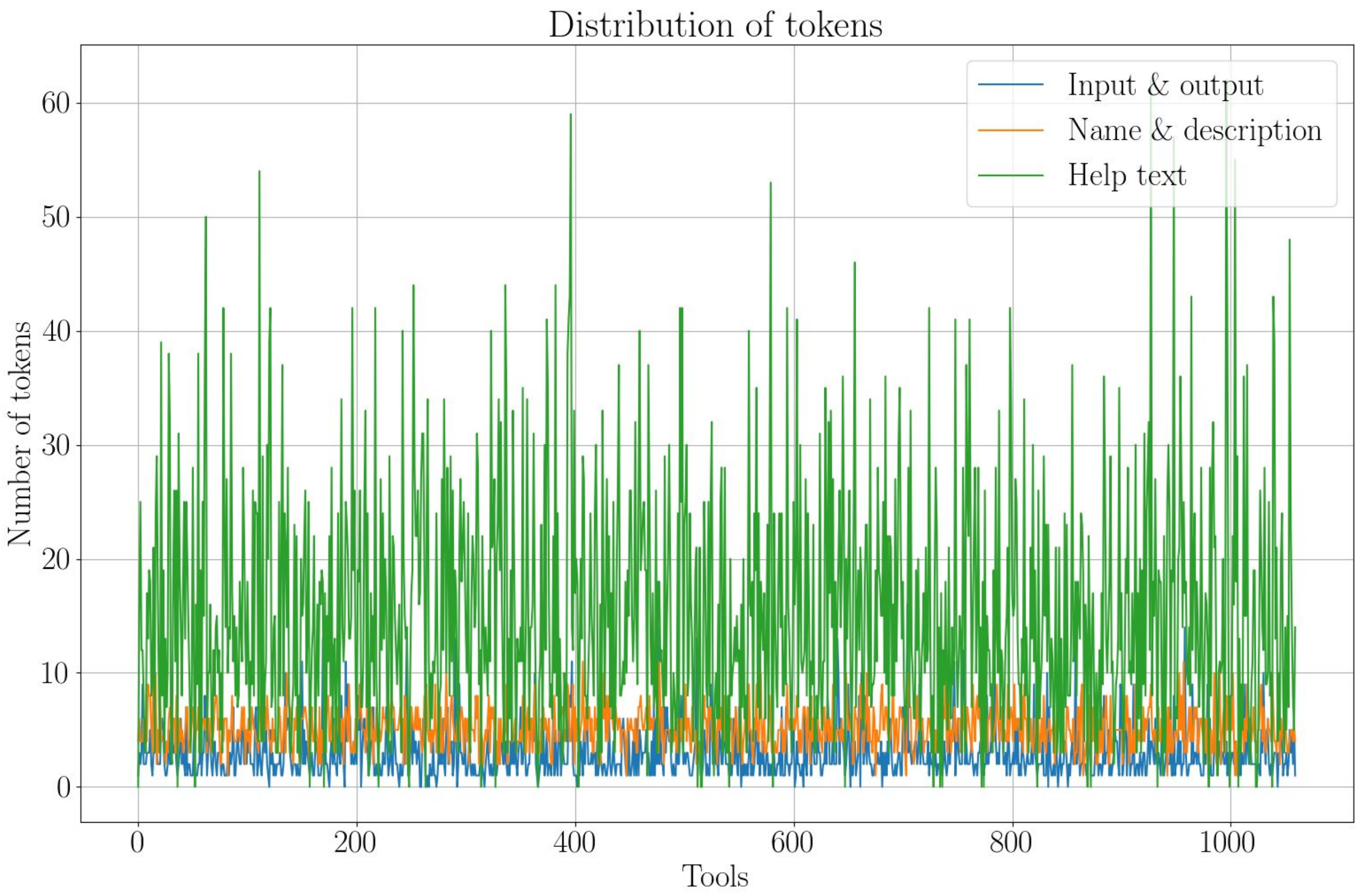
- Compute similarity among tools
- Recommend similar tools
- Replace a tool by its similar tool
- Provide more options for data processing

Approach

1. Collect tools metadata (~ 1,050 tools)
2. Three attributes - input and output files, name and description and help text
3. Clean metadata - stemming and stopwords
4. Learn vectors for tools
5. Find similarity among vectors - similarity matrix
6. Combine similarity matrices

Tool, documents and tokens




Distribution of tokens
Input & output
Name & description
Help text
Number of tokens
0
10
20
30
40
50
60
0
200
400
600
800
1000
Tools

Paragraph (document) vector

- Neural network approach [1]
- A dense vector for each paragraph
- Similar tools have similar vectors
- Encode variable length paragraphs
- Each tool has three paragraphs (documents)
- Input/output file types - Bestmatch25 [2]

1. https://cs.stanford.edu/~quocle/paragraph_vector.pdf
2. https://dl.acm.org/citation.cfm?id=1704810

Similarity scores

- Jaccard index (input and output file types) $j = \frac{A \cap B}{A \cup B}$
- Cosine angle (name and description and help text) $x \cdot y = |x| \times |y| \times \cos\theta$
- Compute similarity matrix
- Three similarity matrices
- Simple - average the matrices
- Better - learn weights

Optimisation

- Gradient descent
- 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

SM_{ideal} is 1.0 as the maximum value of similarity measures can be at most 1.0

Optimisation

Input and output (a)

1	0.34	0.65	0.44
0.34	1	...	...
0.65	...	1	...
0.44	...	...	1

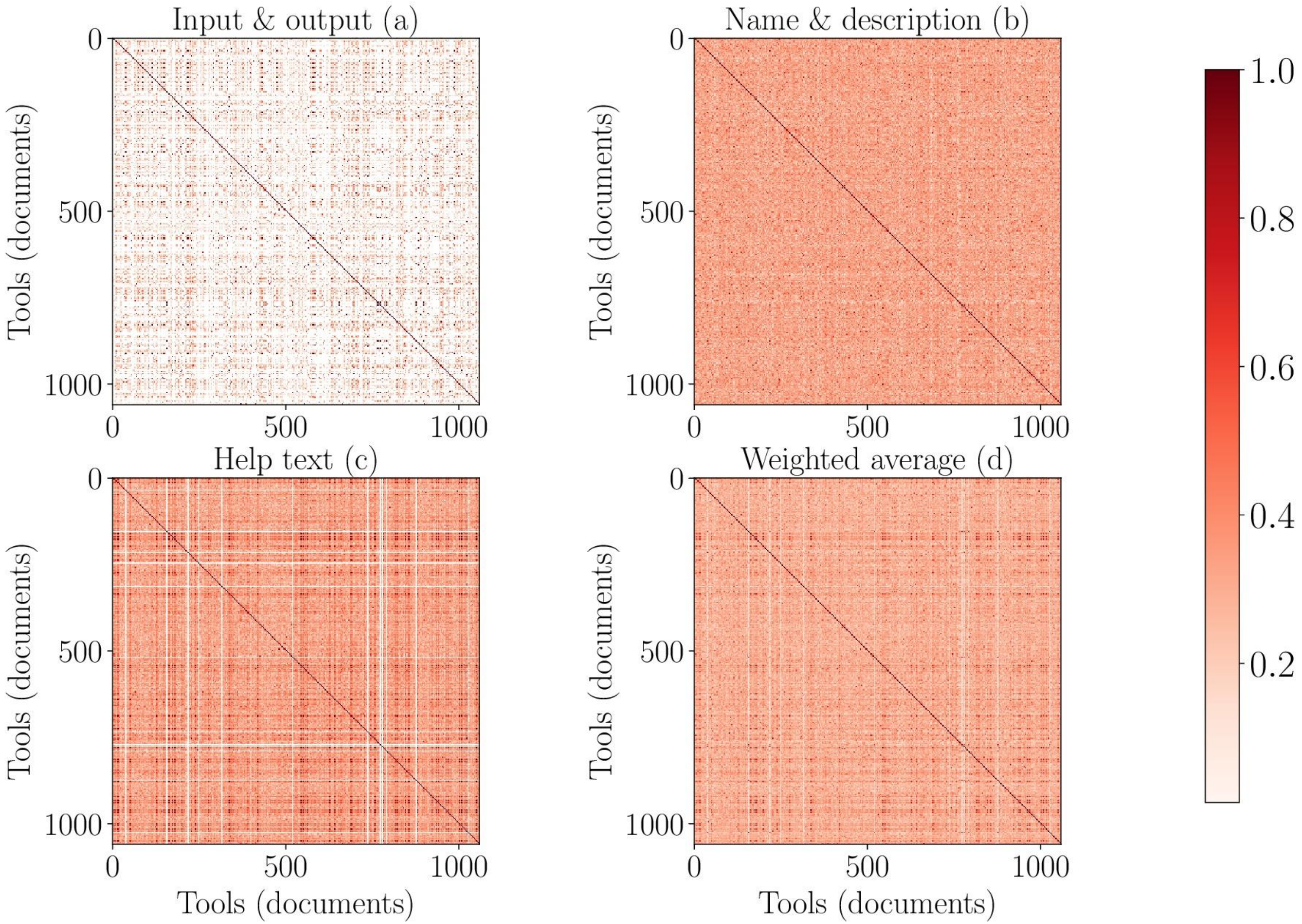
Name and description (b)

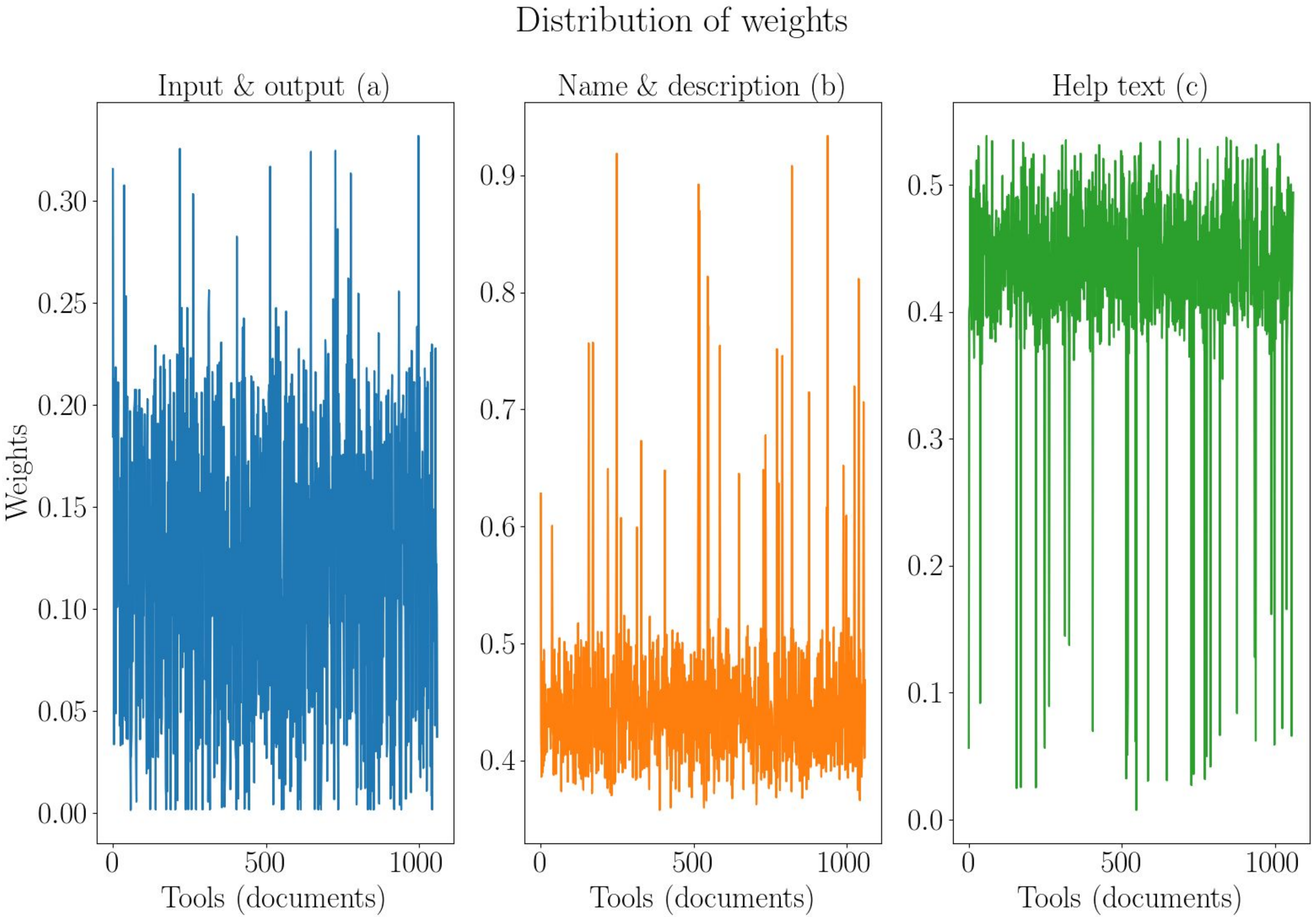
1	0.76	0.63	0.85
0.76	1	...	...
0.63	...	1	...
0.85	...	...	1

Help text (c)

1	0.06	0.1	0.17
0.06	1	...	...
0.1	...	1	...
0.17	...	...	1

Similarity matrices




Distribution of weights
Input & output (a)
Name & description (b)
Help text (c)
Weights
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.4
0.5
0.6
0.7
0.8
0.9
0.0
0.1
0.2
0.3
0.4
0.5
0
500
1000
Tools (documents)
Tools (documents)
Tools (documents)

Summary

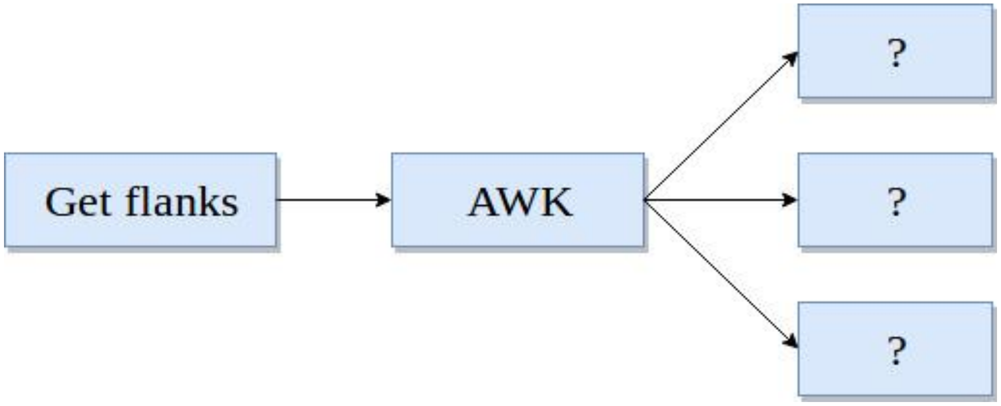
- Collect tools metadata
- Clean metadata
- Learn vector for each tool
- Get similarity matrix
- Combine similarity matrices

Conclusion and future work

- Paragraph vector
- Less data for name and description attribute
- No true similarity
- Create sets of similar tools
- Exclude low similarity values
- Run analysis on larger set of tools
- Compute similar tools using workflows

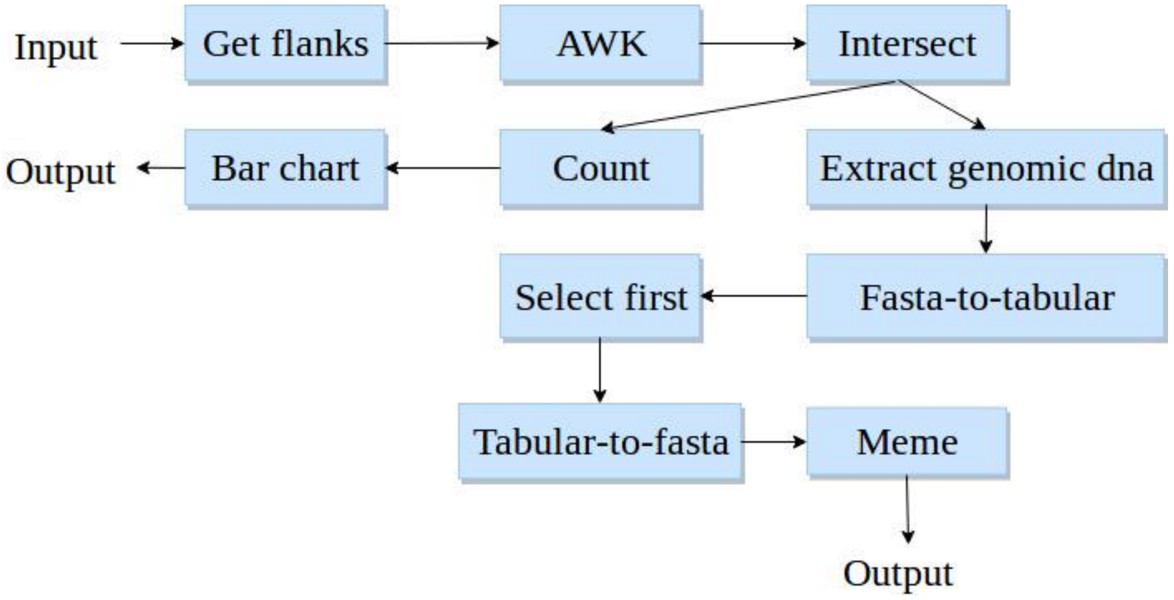
Predict tools in workflows

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



Workflows

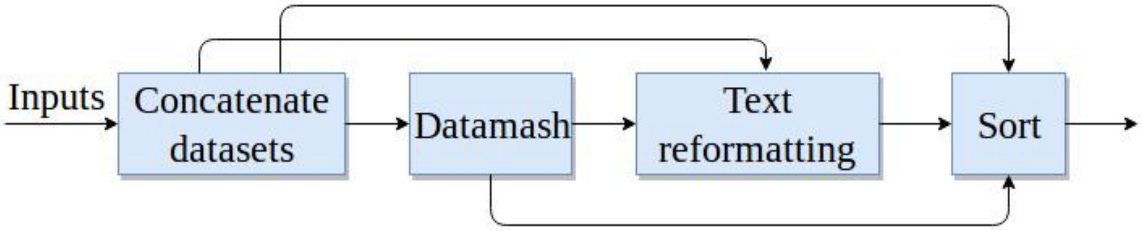
- Workflow - a directed acyclic graph
- Paths in a workflow



- 193,000 workflows
- 900,000 paths
- Max. 25 tools in a path

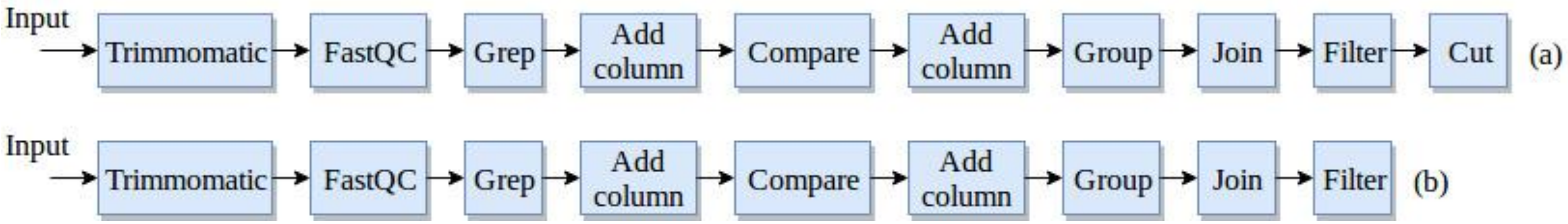
Higher order dependency

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



Workflow preprocessing

- No decomposition



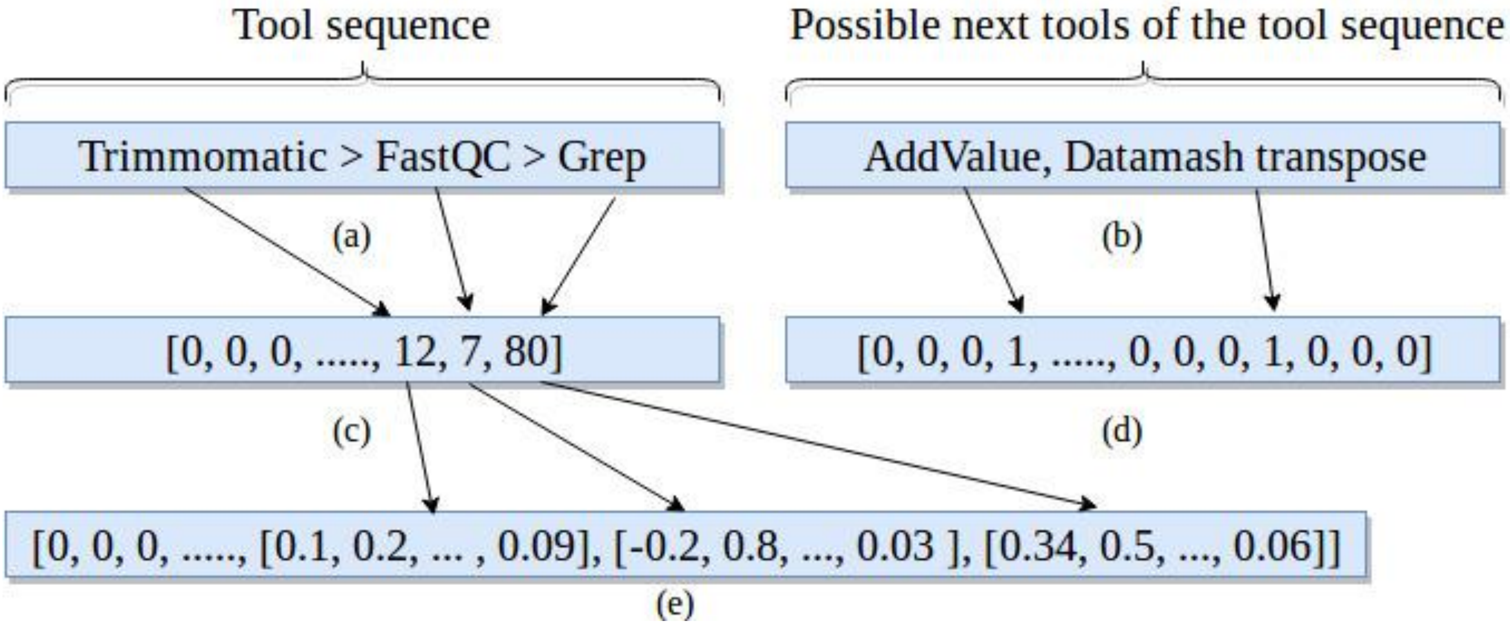
Workflow preprocessing

- Keep first tool fixed

Recurrent neural network

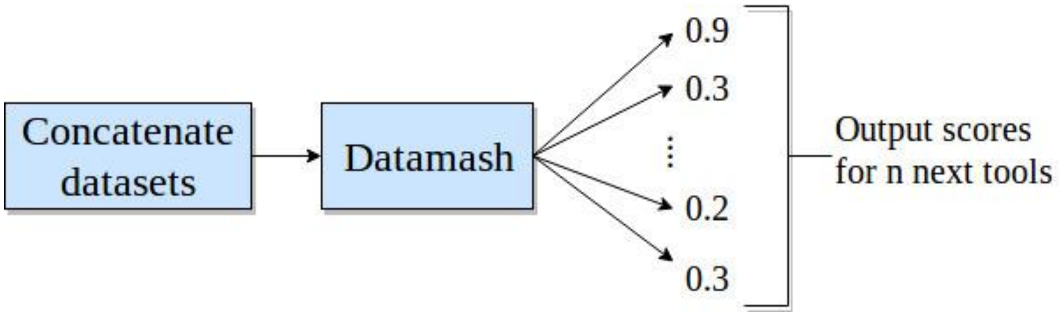
- Classifier to learn on sequential data
- Multilabel, multiclass classification
- Recurrent neural network - Gated recurrent units
- Learn long-range dependencies
- Two hidden layers, one embedding layer and one output layer
- Sigmoid output activation
- Cross-entropy loss
- Root mean square propagation (rmsprop) optimiser

Embedding and label vectors

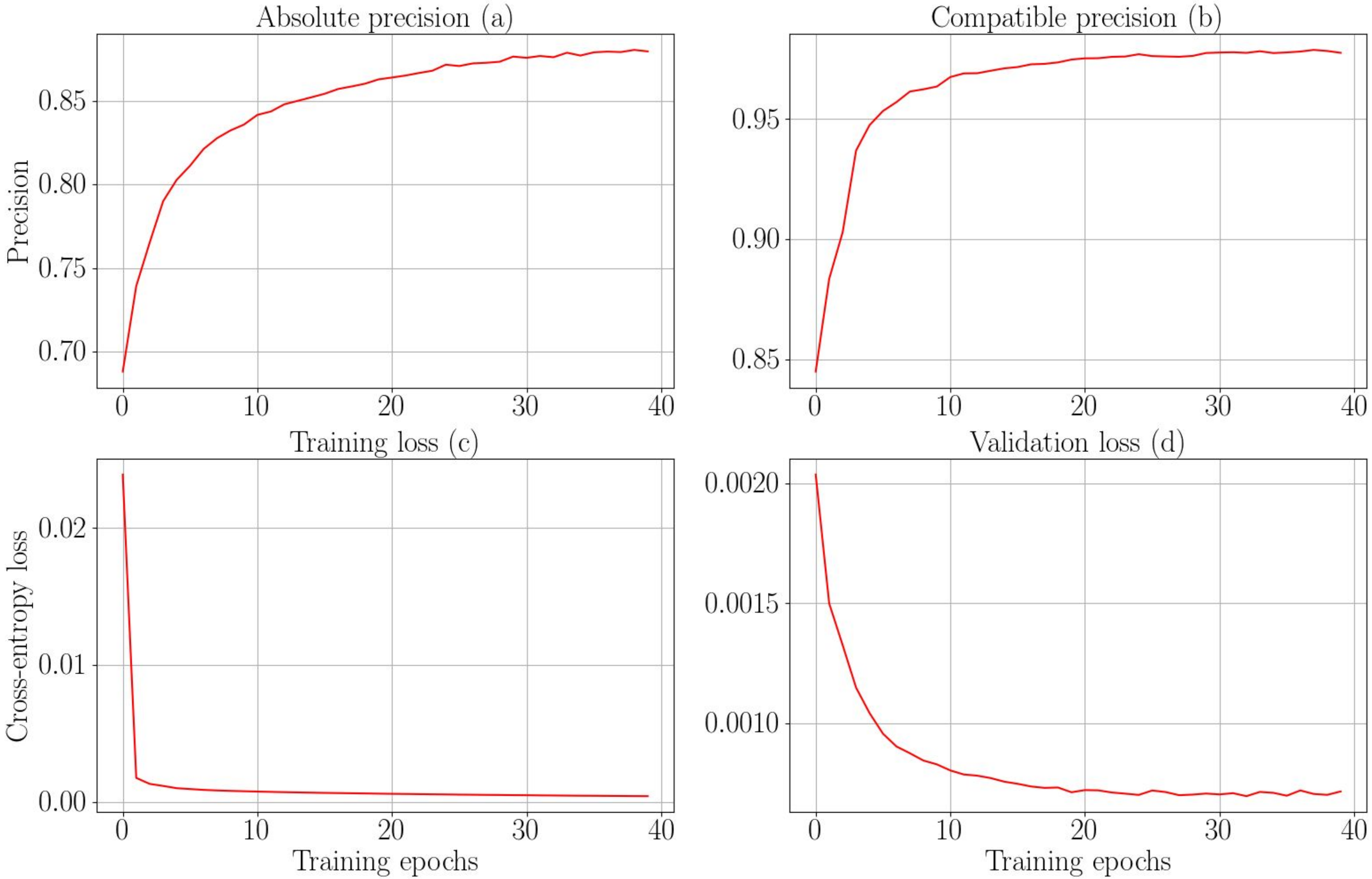


Precision and prediction

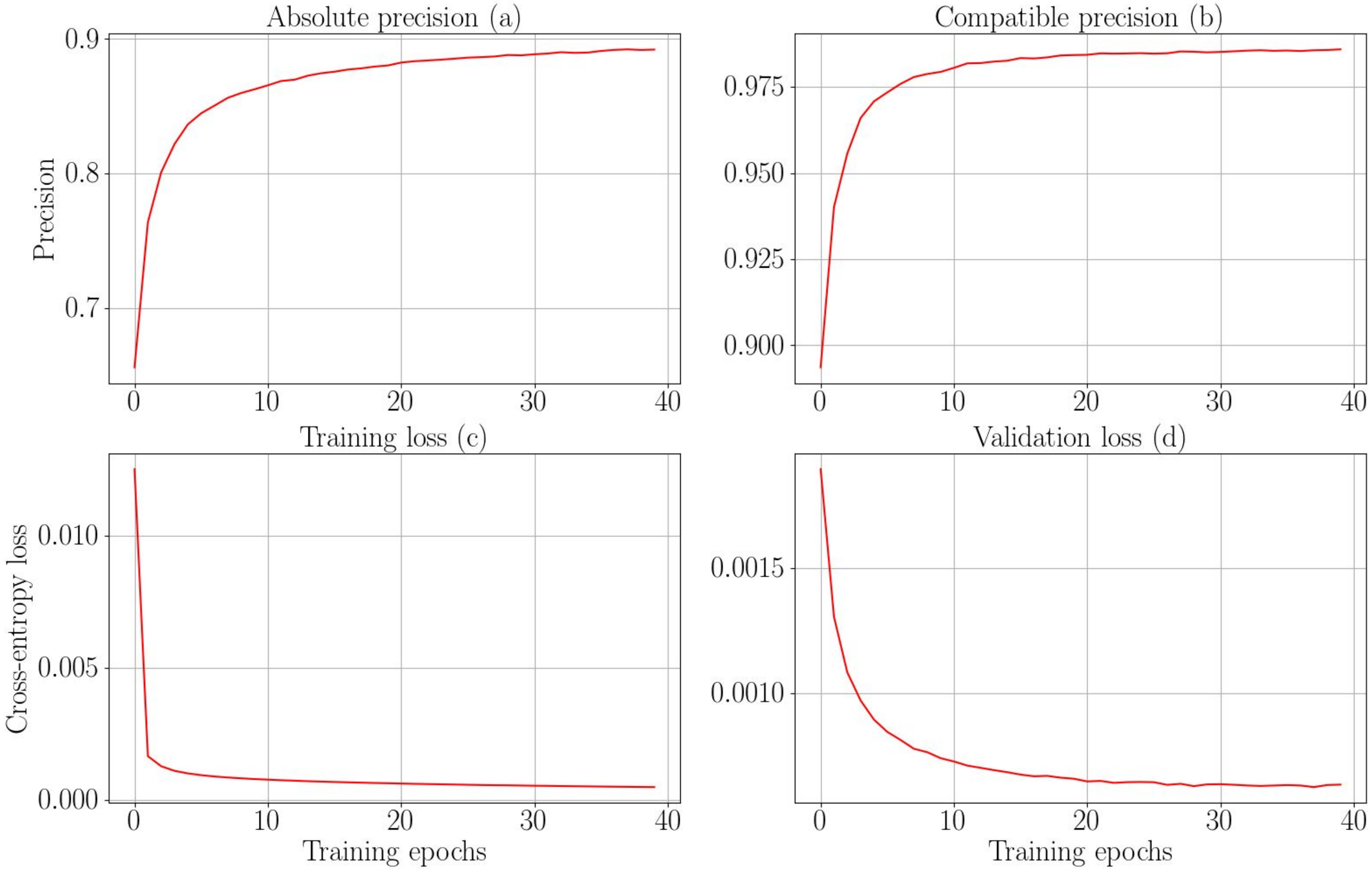
- Absolute precision
- Compatible precision

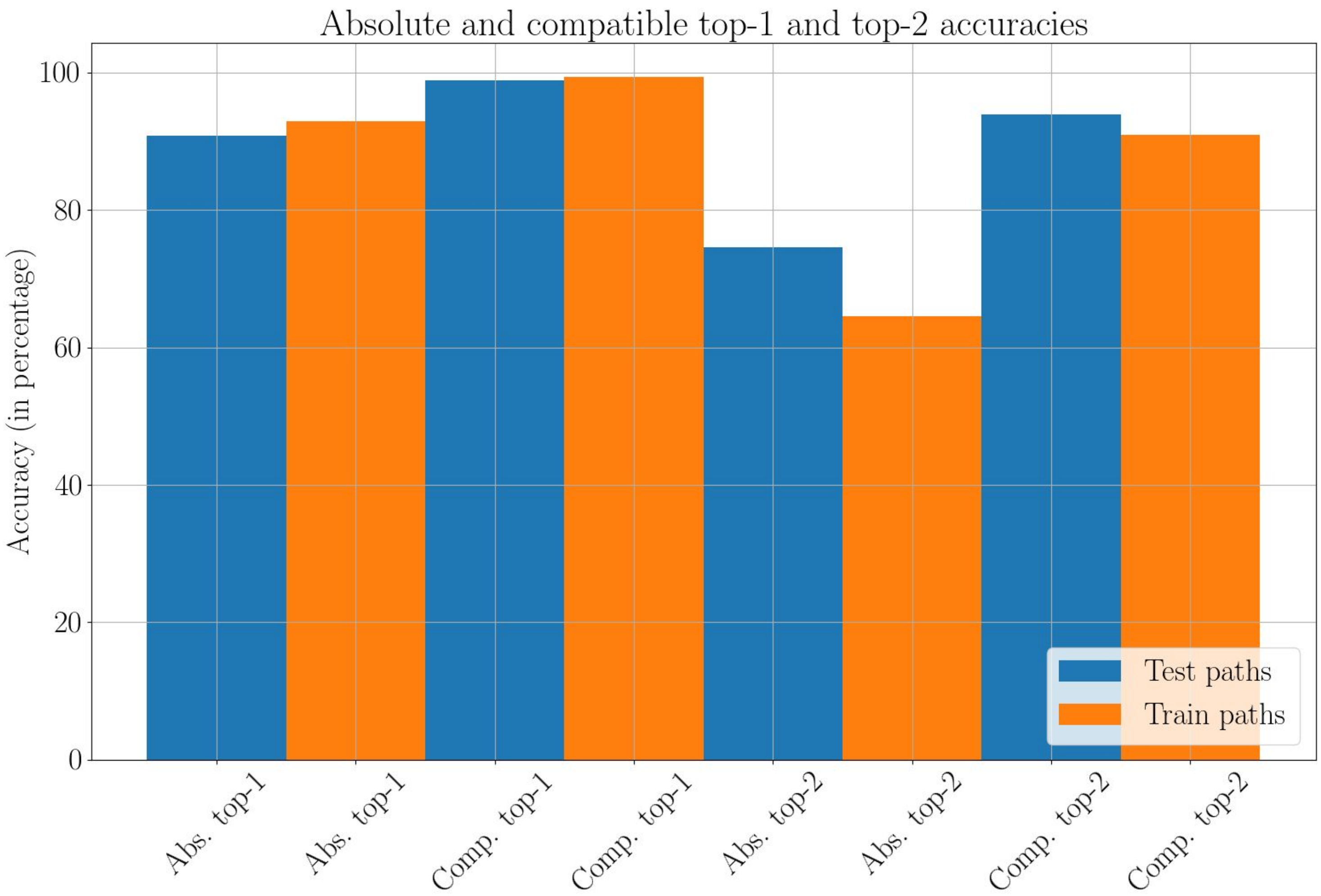


Precision and loss for no decomposition of paths



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 (193, 000 workflows)
- Extract paths from workflows
- Learn long-range dependencies
- Use recurrent neural network (gated recurrent units) as a classifier
- Multilabel, multiclass classification
- Absolute and compatible precision

Conclusion and future work

- Absolute precision ~90%, compatible precision ~99%
- Recurrent neural network suitable for sequential learning
- More workflows, better results
- Restore original distribution
- Decay prediction over time
- Integrate into Galaxy

Thank you for your attention

Questions?

References

Stemming and stopwords

- Stemming - converge all forms of a word into one basic form
- "*Operate, operating, operates, operation, operative, operatives, operational*" into "*oper*" [1]
- Stopwords - "*a, about, above, would, could …*" [2]

1. https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html
2. https://www.ranks.nl/stopwords

Bestmatch25 (bm25)

- Token frequency (tf)
- Document and inverted document frequency (idf)

$$idf = \log \frac{N}{df}$$

$$\alpha = (1 - b) + \frac{b \cdot |D|}{|D|_{avg}}$$

$$tf^* = tf \cdot \frac{k+1}{k \cdot \alpha + tf}$$

$$bm25 = tf^* \cdot idf$$

Bestmatch25 (bm25) scores

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

Latent Semantic Analysis

- Document-token matrix (*X*)
- Singular value decomposition

$$X_{n\times m} = U_{n\times n} \cdot S_{n\times m} \cdot V^T_{m\times m}$$

$$U^T \cdot U = I_{n\times n}$$

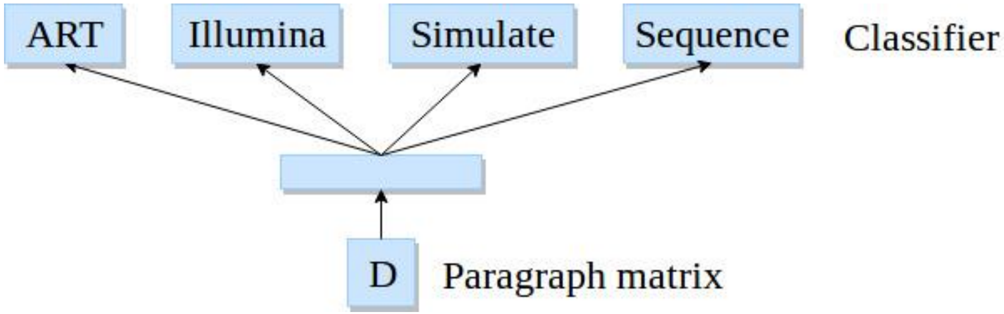
$$V^T \cdot V = I_{m\times m}$$

$$X_{n\times m} = U_k \cdot S_k \cdot V_k^T$$

Paragraph vectors

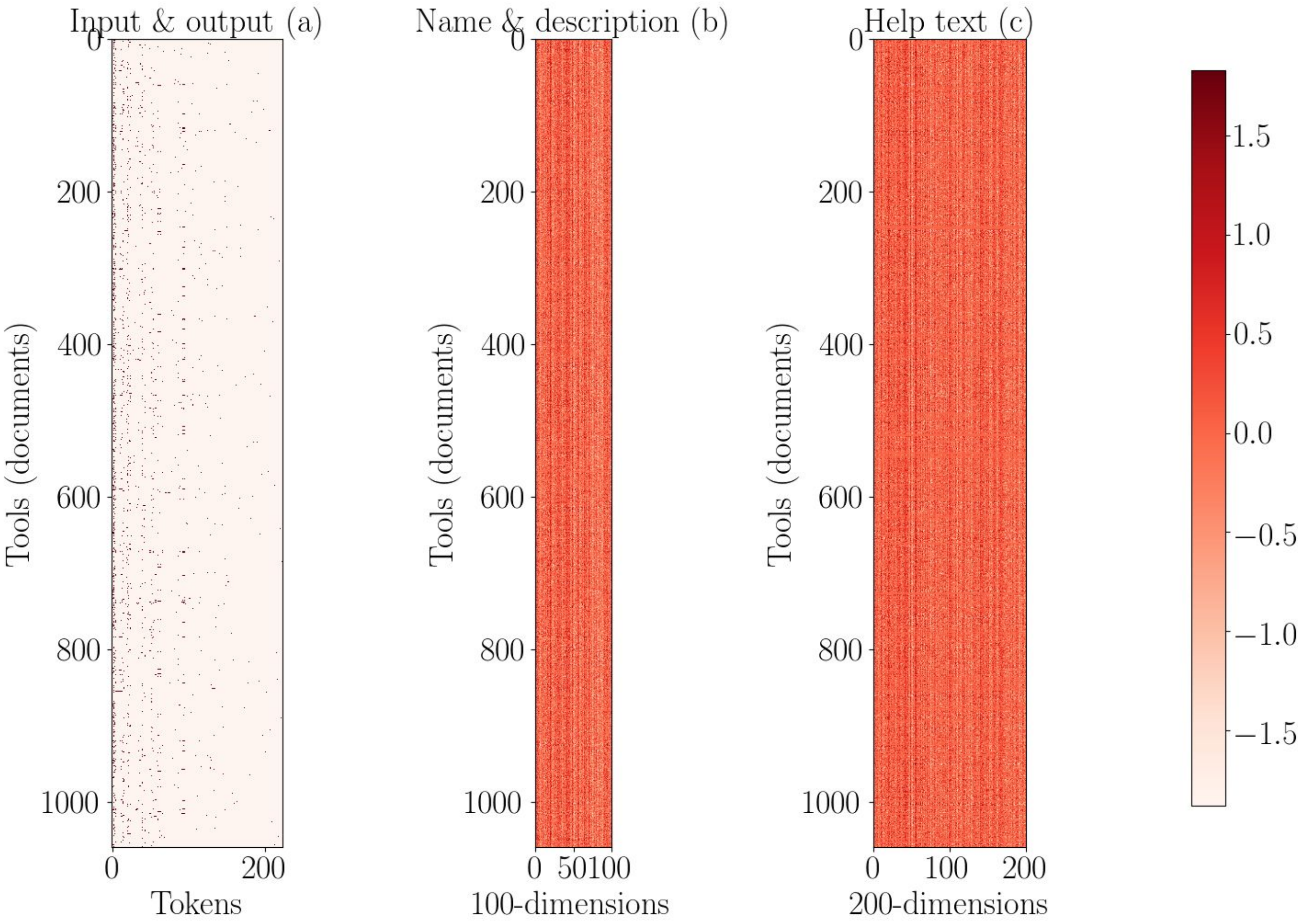
- Softmax classifier
- Backpropagation
- Stochastic gradient descent
- Gensim*

$$\frac{1}{T} \cdot \sum_{t=k}^{T-k} \log p(w_t | w_{t-k}, \ldots, w_{t+k})$$



*https://radimrehurek.com/gensim/models/doc2vec.html
Image adapted from: https://cs.stanford.edu/~quocle/paragraph_vector.pdf

Document-token and paragraph matrices



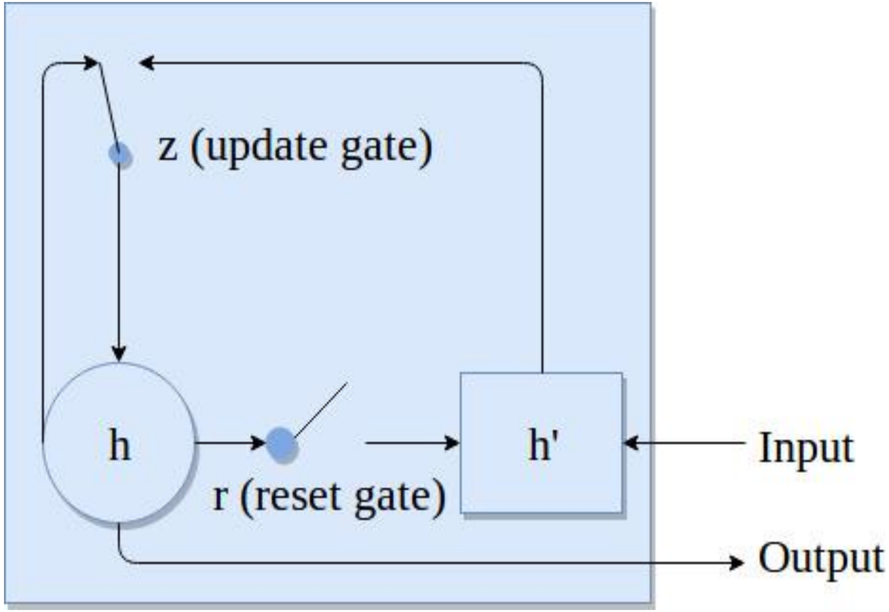
Optimisation

- Gradient descent
- Mean squared error
- Initial learning rate - 0.05
- Decay learning rate
- Nesterov's accelerated gradient

$$update_{t+1} = \gamma \cdot update_t - \eta \cdot Gradient(w_t + \gamma \cdot update_t)$$

$$w_{t+1} = w_t + update_{t+1}$$

Recurrent neural network



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

$$z_t = \sigma(W_z \times x_t + U_z \times h_{t-1})$$

$$r_t = \sigma(W_r \times x_t + U_r \times h_{t-1})$$

$$h_t^{'} = \tanh(W \times x_t + U \times (r_t \odot h_{t-1})$$

$$p(x_T | x_1, x_2, \ldots, x_{T-1})$$

Recurrent neural network

- One embedding layer, two hidden layer and one output layer
- Recurrent layer activation - exponential linear unit

$$f(x) = \begin{cases} x, & \text{if}\, x > 0 \\ \alpha \times (e^x - 1), & \text{if}\, x \leq 0, \alpha > 0 \end{cases}$$

- Output activation is sigmoid $f(x) = \frac{1}{1+e^{-x}}$

Recurrent neural network

- Optimiser - Root mean square propagation (rmsprop)

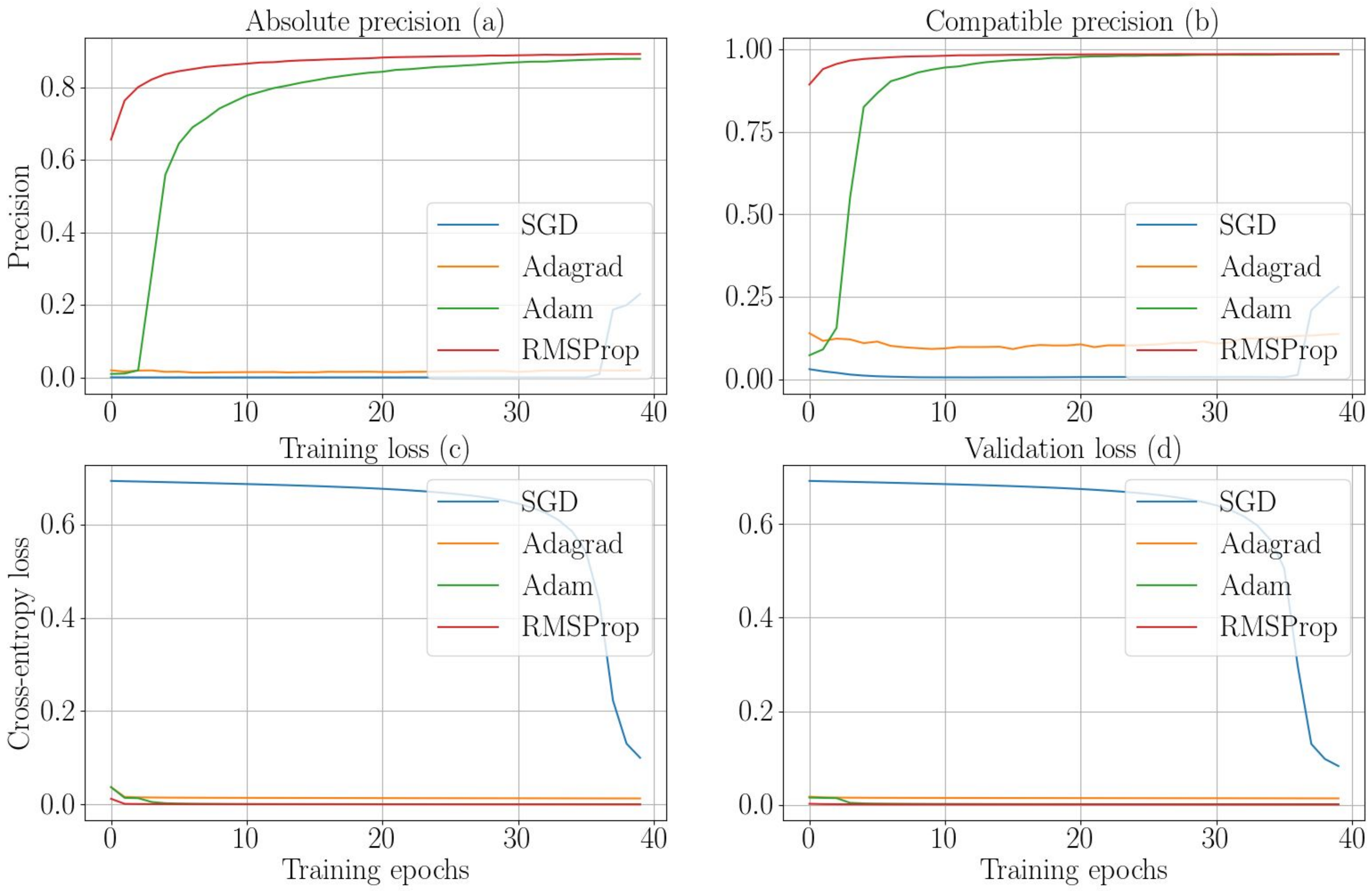
$$MeanSquare(w,t) = 0.9 \times MeanSquare(w,t-1) + 0.1 \times (\frac{\partial E}{\partial w}(t))^2$$

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{MeanSquare(w,t) + \epsilon}} \times \frac{\partial E}{\partial w}(t)$$

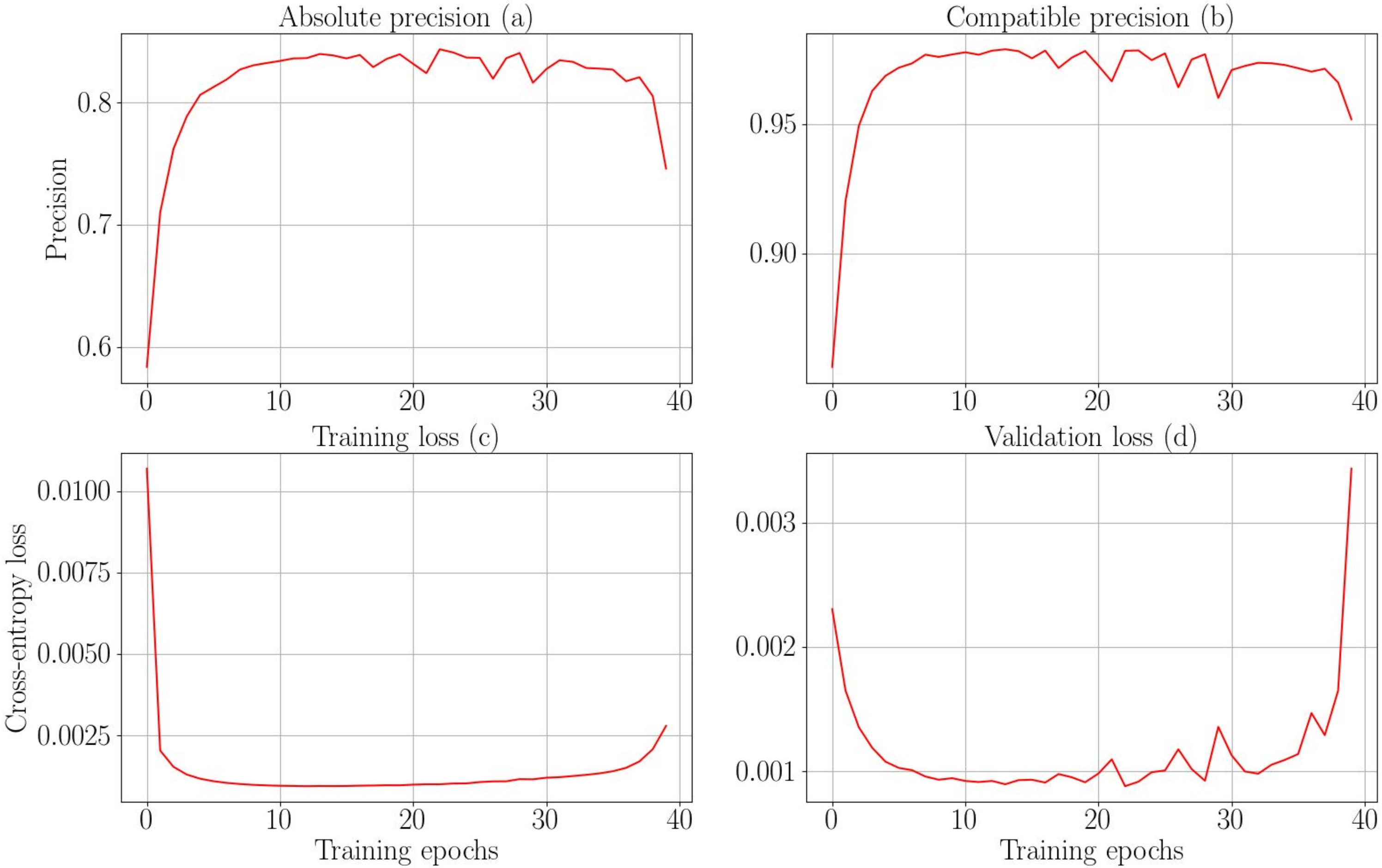
- Binary cross-entropy loss

$$loss_{mean} = -\frac{1}{N}(\sum_{i=1}^{N} y_i \times log(p_i) + (1-y_i) \times log(1-p_i))$$

Precision and loss for various optimisers



Precision and loss using neural network with dense layers



Precision and loss using less data