Master Thesis

Recommender System for Galaxy Tools and Workflows

(Find similar tools and predict next tools in workflows)

Anup Kumar

Examiners:	Prof. Dr. Rolf Backofen and Prof. Dr. Wolfgang Hess
Adviser:	Dr. Björn Grüning

University of Freiburg
Faculty of Engineering
Department of Computer Science
Chair for Bioinformatics

July, 2018

Thesis period

10. 01. 2018 – 09. 07. 2018

Examiners

Prof. Dr. Rolf Backofen and Prof. Dr. Wolfgang Hess

Adviser

Dr. Björn Grüning

Declaration

I hereby declare, that I am the sole author and composer of my thesis and that no other sources or learning aids, other than those listed, have been used. Furthermore, I declare that I have acknowledged the work of others by providing detailed references of said work.
I hereby also declare, that my Thesis has not been prepared for another examination or assignment, either wholly or excerpts thereof.

Place, Date

Signature

Acknowledgment

I express my gratitude to all the people who helped me finishing this work. I thank Dr. Björn Grüning who entrusted me with the task of building a recommendation system for Galaxy. His pragmatic suggestions concerning the tools and workflows helped me to understand them better and improve the results. His advise to create a visualizer for showing the similar tools did wonders as it helped me find and rectify few bugs which would have been difficult to identify. Similar trick to create a visualizer for browsing the results worked with the next task of predicting future tools in workflows as well. I thank Eric Rasche who extracted the workflows for me from the Galaxy Freiburg server. I thank Dr. Mehmet Tekman and Joachim Wolff for their expert feedback and insights. In the end, I wish to thank all the other members of Freiburg Galaxy team for their continuous support and help.

Abstract

The study explores two concepts to devise a recommendation system for Galaxy. One idea is to find similar tools for each tool and another is to predict a set of possible next tools in workflows.

To find similarities among tools, we need to extract information from each tool under multiple categories like a tool's name, description, input and output file types and helptext. We take into account these categories one by one and compute similarity matrices. Each row in a similarity matrix keeps similarity scores of one tool against all other tools. These similarity scores depend on the similarity measure used to compute the score between a pair of tools. We compute three such similarity matrices, one each for input and output file types, name and description and helptext. To combine these matrices, one simple solution is to compute an average. But, assigning equal importance weights to each matrix might be sub-optimal. To find an optimal combination, we use optimization to learn importance weights for the corresponding rows for each tool in the similarity matrices. To define a loss function, we use a true similarity value based on the similarity measures. We take an array of 1.0 as true value because we use jaccard index and cosine similarity as similarity measures which give a positive real number between 0 and 1.

Next task analyzes workflows to predict a set of next tools at each stage of creating workflows. While creating workflows, it would be convenient to leaf through a set of next possible tools as a guide. It can assist the less experienced (Galaxy) users in creating workflows when they are unsure about which tools can further be joined. In addition, it can curtail the time taken in creating a workflow. To achieve that, we need to learn the connections among tools in order to be able to predict the next possible ones. To preprocess each workflow, we compute all the paths bridging the starting and ending tools. These paths contain a set of connected tools. We follow a classification approach to predict the next tools. For the classification task, we use a variant of neural networks (long short-term memory). It performs well for learning long range dependencies (tools connections). We report the accuracy as precision.

Zusammenfassung

Contents

1 Introduction 1

1.0.1 Galaxy 1

1.0.2 Galaxy tools 2

1.0.3 Motivation 3

2 Approach 4

2.1 Data preprocessing 4

2.1.1 Tools attributes 4

2.1.2 Data extraction 4

2.1.3 Clean data 4

2.2 Word embeddings 4

2.2.1 Latent semantic indexing 4

2.2.2 Paragraph vectors 4

2.3 Similarity measures 4

2.4 Optimization 4

3 Experiments 5

4 Results and Analysis 6

5 Conclusion 7

6 Future Work 8

Bibliography 8

1 Introduction

1.0.1 Galaxy

Galaxy [1] is an open-source biological data processing and research platform. It supports numerous types of extensively used biological data formats like FASTA, FASTAQ, GFF, PDB and many more. To process these datasets, Galaxy offers tools and workflows which either transform these datasets from one type to another or manipulate them. A simple example of data processing is to merge two compatible datasets to make one. Another example can be to reverse complement a sequence of nucleotides [2].

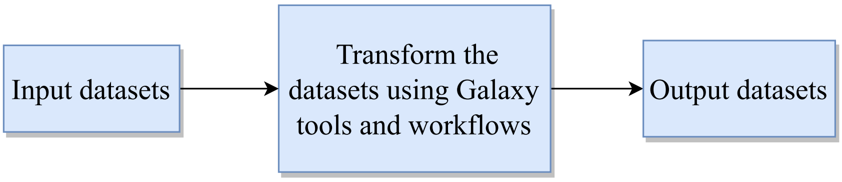

Figure 1: Basic flow of dataset transformation: it shows the basic flow of dataset transformation using Galaxy tools and workflows

A tool is a data-transforming entity which allows one or more types of datasets, transforms these datasets and produces output datasets. The tools are classified under multiple categories based on their functions. For example, the tools which manipulate text like replacing texts and selecting first lines of a dataset are grouped together under "Text Manipulation" group.

These tools form the building blocks of workflows. The workflows are data processing pipelines where a set of tools are joined one after another. The connected tools need to be compatible to each other which means the output types of one tool should be present in the input types of the next tool. A workflow can have one or more starting and ending tools.

[1] https://usegalaxy.eu/

[2] https://usegalaxy.eu/?tool_id=MAF_Reverse_Complement_1&version=1.0.1&__identifer=zmk9dx9ivbk

1.0.2 Galaxy tools

A tool entails a specific function. It consumes datasets, brings about some transformations and produces output datasets which can be fed to other tools. A tool has multiple attributes which include its input and output file types, name, description, help text and so on. They carry more information about a tool. When we look at the collective information about all these attributes for multiple tools, we find that some tools have similar functionalities based on their similarities in their corresponding attributes. For example, there are tools which share similarities in their respective functions and the input and output dataset types they are glued to. For example, a tool "hicexplorer hicpca" [3] has an output type named "bigwig". Hence, if there is a tool or a set of tools which also has "bigwig" as their input and/or output type, we consider there could be some similarity between "hicexplorer hicpca" and the other tools as they do transformations on similar types of datasets. In addition, we can find similar functions of tools by analyzing their "name" and "description" attributes. Let's take an example of two tools (Figure 2):

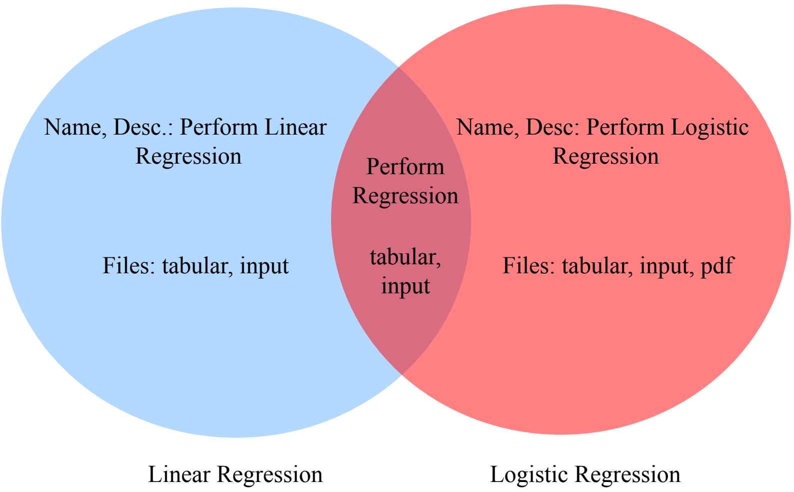
Name, Desc.: Perform Linear Regression
Perform Regression
Name, Desc: Perform Logistic Regression
Files: tabular, input
tabular, input
Files: tabular, input, pdf
Linear Regression
Logistic Regression

Figure 2: Venn diagram: it shows common features extracted from multiple attributes for the two tools

In figure 2, we take two tools - "Linear Regression" and "Logistic Regression" and collect their respective information from their input, output file types, name and description attributes. We see that these tools share features in the venn diagram. They both do regression and few file types are also common. In the same way, if we

[3] `https://usegalaxy.eu/?tool_id=toolshed.g2.bx.psu.edu/repos/bgruening/hicexplorer_hicpca/hicexplorer_hicpca/2.1.0&version=2.1.0&__identifer=5kcqmvb71gx`

extrapolate this venn diagram and match one tool against all other tools, we hope to find a set of tools similar in nature to the former tool. While searching for the related tools for a tool, it is possible that we end up with an empty set.

1.0.3 Motivation

From figure 2, we see that there can be tools which share characteristics. Galaxy has thousands of tools having a diverse set of functions. Moreover, new tools are keep getting added to the older set of tools. From a user's perspective, it is hard to keep knowledge about so many tools. It is important to make a user aware of the presence of new tools added. If there is a model which dispenses a clue that there is a set of say n tools which are similar to a tool, it would give more options to a user for her/his data processing.

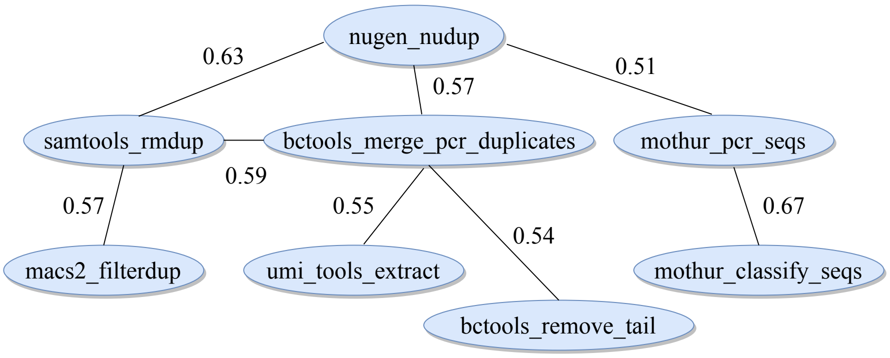

Figure 3: Similarity knowledge network: it shows how the tools in the network are related. The real numbers show the relation strength

To elaborate it more, let's take an example of a tool "nugen nudup" [4]. It is used to find and remove PCR duplicates. The similar tools for it can be "samtools rmdup" and "bctools merge pcr duplicates" which work on related concepts. These similar tools would further have their respective set of similar tools thereby making a network of related entities (tools). This "knowledge network" can help a user find multiple ways to process her/his data and exhibits a exhibits "connectedness/relation" among tools. The strength of this relation may vary from being small to large. We learn a continuous representation of the relation strength. Figure 2 shows how this graph can evolve. First, we find similar tools for "nugen nudup" and connect them to their source tool specifying the similarity values as real numbers at the edges. These similar tools further have their own sets of similar tools and so on.

[4] https://toolshed.g2.bx.psu.edu/repository?repository_id=4f614394b93677e3

2 Approach

2.1 Data preprocessing

2.1.1 Tools attributes

2.1.2 Data extraction

2.1.3 Clean data

Refine tokens

2.2 Word embeddings

2.2.1 Latent semantic indexing

2.2.2 Paragraph vectors

2.3 Similarity measures

Jaccard index

Cosine similarity

2.4 Optimization

Gradient descent

Backtracking line search

3 Experiments

4 Results and Analysis

5 Conclusion

6 Future Work