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
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 encouraged and supported me to accomplish this work. I am grateful to my mentor Dr. Björn Grüning who entrusted me with the task of building a recommendation system for Galaxy. He facilitated this work by providing me with all the indispensable means. Being specific, his pragmatic suggestions concerning the Galaxy tools and workflows helped me discern them better and improve the performance. His advice to create a visualizer for showing the similar tools worked wonders as it enabled me to find and rectify a few bugs which were tough to establish. For the next task, creating a separate visualizer for looking through the next predicted tools was conducive in all merits. I appreciate and thank Eric Rasche who extracted the workflows for me from the Galaxy Freiburg server. I offer thanks to Dr. Mehmet Tekman and Joachim Wolff for their expert feedback, insights and general advice. At length, 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 Galaxy tools for each tool and another is to predict a set of possible next tools in Galaxy workflows.

To find similarities among tools, we need to extract information about each tool from its attributes like name, description, input and output file types and help text. We take into account these attributes one by one and compute similarity matrices. We compute three similarity matrices, one each for input and output file, name and description and help text attributes. Each row in a similarity matrix holds similarity scores of one tool against all the other tools. These similarity scores depend on the similarity measures (jaccard index and cosine similarity) used to compute the score between a pair of tools. To combine these matrices, one simple solution is to compute an average. But, assigning equal importance weight 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 for optimization, we use a true similarity value based on the similarity measures. The similarity scores are positive real numbers between 0 and 1. We take an array of 1.0 as the true value.

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 connected. 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 based on the previously connected tools. To preprocess the workflows to make them usable by downstream machine learning algorithms, we compute all the paths bridging the starting and end tools in all workflows. We follow a classification approach to predict the next tools and use LSTM (long short-term memory), a variant of recurrent neural networks. It performs well for learning long range, sequential and time-dependent data (tools connections) [1, 2]. We report the accuracy as precision.

Zusammenfassung

Contents

1 Introduction

1.1 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].

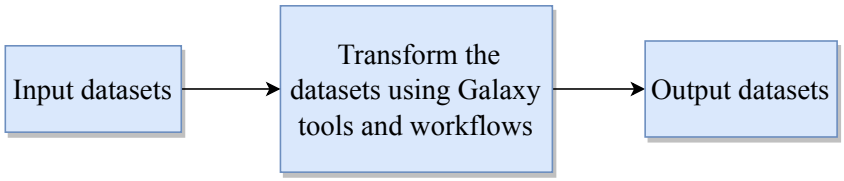

Figure 1: Basic flow of dataset transformation: it shows the basic flow of dataset transformation using Galaxy tools and workflows

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 into 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 with 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 end tools.

[1] `https://usegalaxy.eu/`

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

1.2 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):

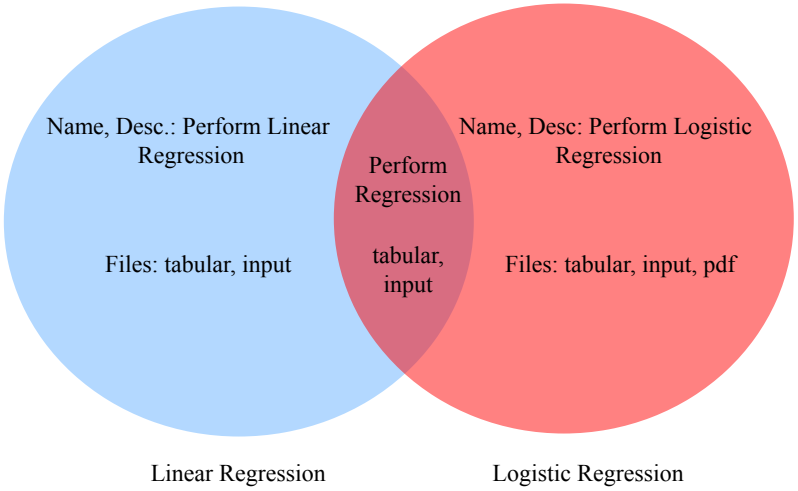

Figure 2: Venn diagram: it shows common features extracted from multiple attributes for the two tools

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.

1.3 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 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.

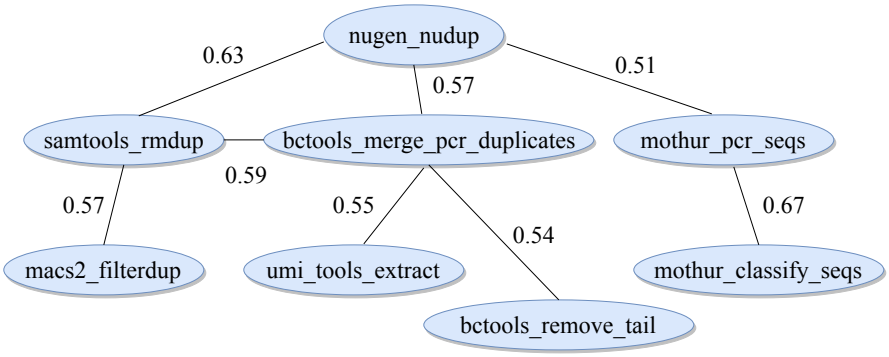

Figure 3: Similarity knowledge network: it shows how the tools in the network are related. The real numbers show the relation strength

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 "connectedness" among tools. The strength of this relation may vary from being small to large. To ascertain that, this study learns a continuous representation of the relation strength. Figure 2 shows how this knowledge 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 Extract data

There are multiple repositories of Galaxy tools stored at GitHub [1]. In each of the tool repository, there are *xml* files starting with a $< tool >$ tag. We read all of these *xml* files, extract information from a few of the attributes and collect them in a *tabular* file.

2.1.1 Select tools attributes

A tool has multiple attributes like input and output file types, help text, name, description, citations and some more. But all of these attributes are not important and do not generally identify a tool exclusively. We consider some of these attributes:

- Input and output file types
- Name and description
- Help text

Moreover, we combine the input and output file types and name and description respectively as they are of similar nature. These two combined attributes give complete information about a tool file types and its functionality. We also consider help text attribute which is larger in size compared to the previous two. At the same time, they are empty for few tools. Apart from being large in size, this attribute is noisy as well. It provides more information about the usage of a tool. Generally, in the first few lines, it gives a detailed explanation of the tool functions. Further, it explains how the input data should be supplied to a tool or how an input data looks like. Hence, much of the information present in this attribute is not important. Because of noise present in this attribute, we decide to use only upto first 4 lines which illustrates the core functionality of the tool. The decision to select only first 4 lines is empirical. The rest of the information in help text is discarded.

[1] One example:`https://github.com/galaxyproject/tools-iuc/tree/master/tools`

2.1.2 Clean data

Remove duplicates and stopwords

The collected data for tools is raw containing lots of commonplace and duplicate items which do not add value. These items should be removed to get *tokens* which are unique and useful. For example, a tool $bamleftalign$ has input files as $bam, fasta$ and output file as bam. While combining the file types, we discard the repeated file types and in this case, we consider file types as $bam, fasta$. The other attributes we deal with are different from the file types. The files types are discrete items but in attributes like name and description and help text, the account is in a human language. The explanation contains complete or partially complete sentences in *English*. Hence, to process this information, we need startegies that are prevalent in natural language processing [2]. The sentences we write in *English* contain many words and has different parts. These parts include subject, object, preposition, interjection, verbs, adjectives, adverbs, articles and many others. For our processing, we need only those tokens (words) which categorize a tool uniquely and do away with multiple parts of speech present in the statements. For example, a tool named *tophat* has name and description as "TopHat for Illumina Find splice junctions using RNA-seq data". The words like for, *using* and *data* do not give much value as they must be present for many tools. These words are called as "stop words" [3] and we selectively discard them. In addition, we remove numbers and convert all the tokens to lower case.

Use stemming

After removing duplicates and stop words, our data is clean and contain tokens which uniquely identify corresponding tools. When we frame sentences, we follow grammar which constrains us to use different forms of the same word in varying contexts. For example, a word *regress* can be used in multiple forms as *regresses* or *regression* or *regressed*. They share the same root and point towards the same concept. If many tools use this word in varying forms, it is beneficial to converge all the different forms of a word to one basic form. This is called stemming [4]. We use nltk [5] package for stemming.

[2] `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3168328/`
[3] `https://www.ranks.nl/stopwords`
[4] `https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html`
[5] `http://www.nltk.org/`

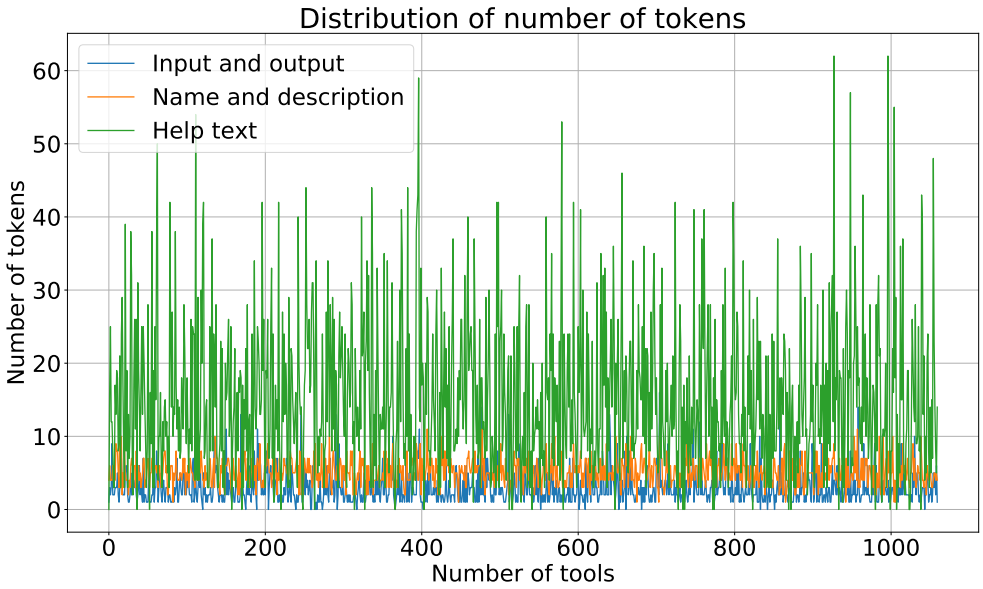

Figure 4: Distribution of tokens: the plot shows the distribution of tokens for all the attributes. The help text has the most number of tokens for all tools and input and output has the least number of tokens.

Refine tokens

At this stage of tools preprocessing, we have a set of good tokens for each attribute whic are input and output file types, name and description and help text. Let's call these sets as *documents*. The tokens present in these documents do not carry equal importance. Some tokens are more relevant to the document and some not so relevant . We need to find out importance factor for all tokens in a document. Using these factors, we can arrange them in big, sparse documents-tokens matrix. In this matrix, each row represents a document and each column belongs to one token. To compute these relevance scores, we use bestmatch25. Let's associate some variables to be used in explaining this algorithm.

- Token frequency [6] tf
- Inverted document frequency idf
- Average document length $|D|_{avg}$
- Number of documents N

[6] `https://nlp.stanford.edu/IR-book/pdf/06vect.pdf`

- Size of a document $|D|$

First of all, token frequency (tf) specifies the count of a token's occurrence in a document. If a token *regress* appears twice in a document, its tf is 2. This can also be understood as a weight given to this term. Inverted document frequency for a token is defined as:

$$idf = \log \frac{N}{df} \tag{1}$$

where df is the count of the documents in which this token is present and N is the total number of documents. If we randomly sample a document, then the probability of this token to be present in this document is $p_i = \frac{df}{N}$. From information theory, we can say that the information contained by this event is $-\log p_i$. The entity idf is higher when a token appears less number of documents which means that this token is a good candidate for representing the document and possesses higher power to distinguish between documents. The tokens which appear in many documents are not good representatives. Average document length is the average number tokens for all the documents. Size of a document is the count of all the tokens for that document [3].

$$\alpha = (1 - b) + \frac{b \cdot |D|}{|D|_{avg}} \tag{2}$$

$$tf^* = tf \cdot \frac{k + 1}{k \cdot \alpha + tf} \tag{3}$$

$$BM25_{score} = tf^* \cdot idf \tag{4}$$

where k and b are hyperparameters. Using the equation 4, we compute the relevance score for each token in all the documents. Table 1 shows some sample scores for a few documents where the tokens are present with their respective relevance scores. In this way, we arrange document-tokens matrix for all the attributes of tools. For input and output file types, these matrix entries will have only two value, 1 if a token is present for a document and 0 if not. For other attributes, relevance scores are positive real numbers. This strategy of representing documents with their tokens is called vector space model as each document represents a vector of tokens.

Figure 4 shows the heatmaps for documents-tokens matrices that belong to input and output file types and name and description. We can see that these plots are sparse. Each entry in these matrices contain BM25 score for each token in every document.

Documents/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	1.2	1.47	0.0
Hisat	0.0	0.0	0.0	0.0	0.0

Table 1: Document-tokens matrix: it stores relevance scores for each token for all the documents

The representation shows how to find tokens which are good representatives of documents with a weighted by their relevance factors. But, they do not tell un anything about the co-occurence of a few tokens in a document. It tells us that a token is important for a document if the BM25 score is higher but it does not tell us anything about its relation to other tokens. Due to this shortcoming, it does not acknowledge the presence of "concepts" or "context" hidden in a document. A concept in document can be realised when we see the relation among a few words. To illustrate this idea, let's take an example of three words - "New York City". These three words mean little or point to different things if looked at separately. But, if we see them together, it points towards a concept. This vector space model lacks the ability to find the correlation among tokens. To enable the vector space model to learn this hidden concepts and find correlation among multiple tokens, we explore two ideas

- Latent Semantic Indexing/Analysis
- Paragraph Vectors

Using these approaches, we learn dense, n dimensional vector for each document instead of using sparse vectors as shown in figure 5.

2.2 Document embeddings

2.2.1 Latent semantic indexing

It is statistical way to learn the hidden concepts in documents by computing a low-rank representation of a documents-tokens matrix. This low-rank matrix is dense (figure 6). We use scalar value decomposition (SVD) to decompose the full-rank

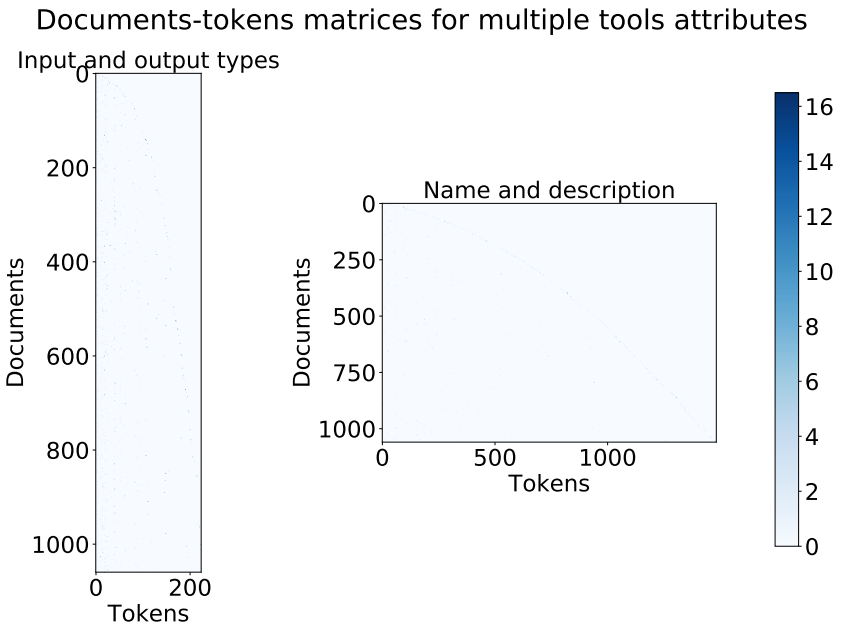

Figure 5: Documents-tokens matrices: the heatmap shows the documents-tokens sparse matrices for two attributes.

matrix into a significantly lower rank matrix. The optimal rank to which a matrix to be decomposed is empirical in nature. This decomposition follows the equation:

$$X_{nxm} = U_{nxn} \cdot S_{nxm} \cdot V^T_{mxm} \tag{5}$$

where n is the number of documents and m is the number of tokens. S is a diagonal matrix containing the eigen values in descending order. It contains the weights of the concepts present in the matrix. The matrices U and V and orthogonal matrices which satisfy:

$$U^T \cdot U = I_{nxn} \tag{6}$$

$$V^T \cdot V = I_{mxm} \tag{7}$$

The matrix U contains information about how the tokens are mapped to concepts and matrix V stores information about how the concepts are mapped to documents.

The low-rank approximated matrix X_k is computed as:

$$X_{nxm} = U_k \cdot S_k \cdot V^T_k \tag{8}$$

where U_k is the first k columns of U, V_k is the first k rows and S is the first k eigen values. k is an empirical parameter. X_k is called as the rank-k approximation of the full rank matrix X.

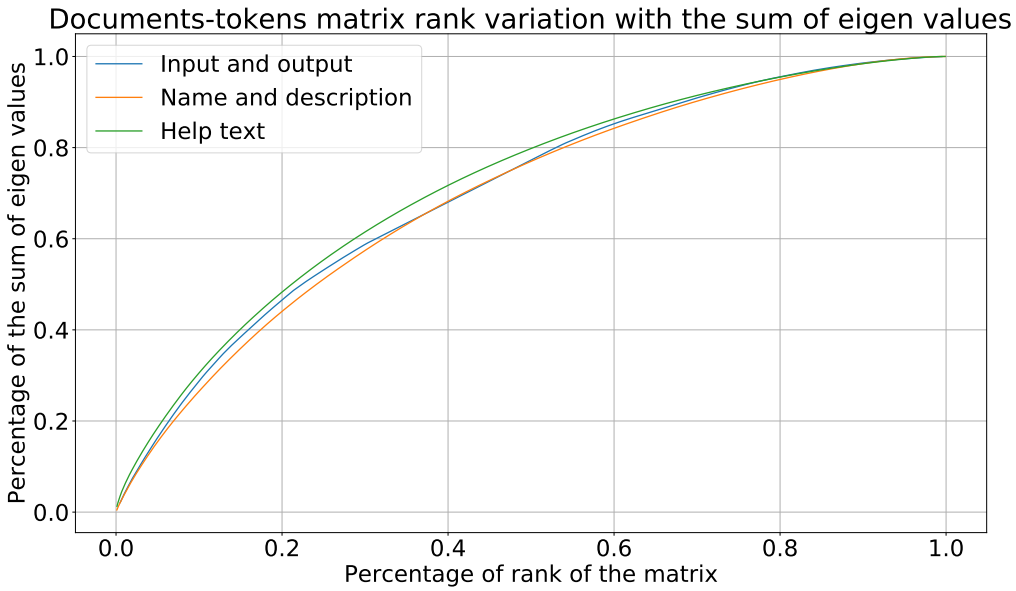

Figure 6: Matrix rank and eigen values: the plot shows how the sum of principal eigen values vary with the rank of the documents-tokens matrix for all the attributes.

2.2.2 Paragraph vectors

2.3 Similarity measures

Jaccard index

Cosine similarity

2.4 Optimization

Gradient descent

Backtracking line search

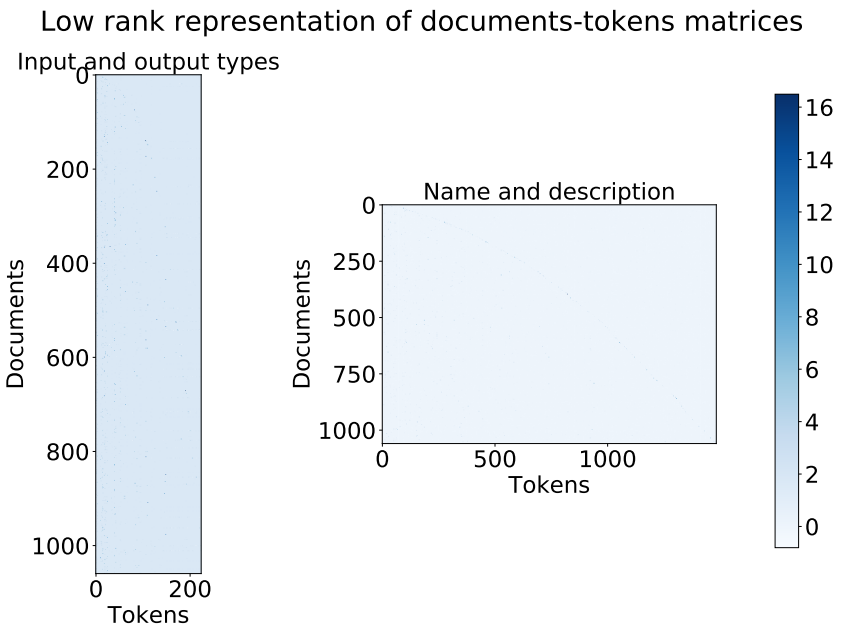

Figure 7: Documents-tokens matrices: the heatmap shows the documents-tokens low rank representation of the matrices.

3 Experiments

4 Results and Analysis

5 Conclusion

6 Future Work

Bibliography

[1] Z. C. Lipton, D. C. Kale, C. Elkan, and R. C. Wetzel, "Learning to diagnose with LSTM recurrent neural networks," *CoRR*, vol. abs/1511.03677, 2015.

[2] H. Sak, A. W. Senior, and F. Beaufays, "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition," *CoRR*, vol. abs/1402.1128, 2014.

[3] S. Robertson and H. Zaragoza, "The probabilistic relevance framework: Bm25 and beyond," *Found. Trends Inf. Retr.*, vol. 3, pp. 333–389, Apr. 2009.