YesWorkflow cases- topic modeling in Text Mining area

Abstract

Yesworkflow is a toolkit which can offer workflow results to users with different platforms, including bash, Python, R and Matlab, Java, C/C++. YesWorkflow will generate workflow from the annotations made by users. The annotates are very easy to put into use. And here the case is to use Gensim LDA models to do topic modeling. Before the very topic modeling apart, I need to do data cleaning and preprocess the text data and transform text data into vectors. Then through tuning the hyper-parameter of the LDA model, I can find the converge point for the model. Using log file with the log tag in the YesWorkflow, I can track the parameters which I'm interested in. And the whole steps can turn the models from black box into white one. Finally, YesWorkflow implement reconstructed provenance script, which will be used in the querying step. Here I use the example workflow-structure queries that are supported by YesWorkflow. And use ProvToolbox to generate the gv, png and pdf file to show the outputs.

Keyword: YesWorkflow, ProvToolbox, Query, topic modeling

1.Yesworkflow

[1]Yesworkflow is a User-Oriented toolkit, which can generate the workflow from the YW(Yesworkflow) annotations. YW is like a black box, which is combined with ProvToolbox. Users are not required to know how to use this work engine nor know how to put code into effectively use in such a system. There mainly three tags in pairs here: @Begin and @End represent start this workflow and end this workflow, and here normally we will use program block as a sub-workflow. Specially, if there is some invokes in the program block itself, YW will generate a double border to express “there is a sub-workflow here”; @IN and @OUT represent input the data and output the products; @param represents the parameters input from outside, for @Log tag, YW use parenthesis “{}” to get the parameters recorded in the log file which interests the users. And the very last @uri , record the locations for all of the inputs and outputs.

After all these annotations have been defined, using YW command to generate the recon file for later use:

Predefine YW command: $alias yw= 'java -jar ~/yesworkflow-0.2.2.0-SNAPSHOT-jar-with-dependencies.jar'

Generate recon file: $ yw recon topic_modeling.py -c recon.comment='#' -c recon.factsfile=reconfacts.P

Recon file is reconstructed prospective model augmented by retrospective provenance, which is used to generate retrospective provenance. Prospective provenance can be called as what kind of output the users want to generate, on the other hand, retrospective provenance is the actual outputs. That’s also why we want to use YW to help users debug, where YW will try to find the outputs following the @uri, if users find that the final recon file can’t present the uri location, it means that there must something wrong with the original script.

2.Text Mining

Nowadays, NLP (Natural Language Processing) becomes more and more popular. Text Mining and NLP are both tools which can be called as AI (Artificial Intelligence) that empower the users to catch the very key points from the large collection of datasets. Through text mining process, people can generate new information from the large size of written text data, as well as transform the unstructured data into structured data.

Step 1. data cleaning

Here I use Neural Information Processing Systems (NIPS) dataset, ranging from 2000 to 2012 years. It's said that NIPS dataset is one of the top conference in machine learning area. Before any data analysis steps, we all need to do data cleaning, where removing stop words, lower the alphabet, using regular expression to filter the unlawful strings or words. I have more steps for special reasons, including calculate the word frequency and remove the word which only appears one time. When we remove all these "trash" words, then It comes text mining procedure. The very first is to use NLTK library to lemmatize all the words in the documents. And then add the bigrams to the words which appear 20 times or more in the documents.

Step 2. do the transformation and construct corpus

In this step, we use Gensim library, which is one of the most popular text mining libraries. We transfer the words in the documents into dictionary, where Gensim library will associate each word in the corpus with a unique integer ID.

After all this done, we have our own corpus, and we can use the Latent Dirichlet Allocation model to do the following steps.

Step 3. use Gensim Latent Dirichlet Allocation model

Log_perplexity: calculate and return per-word likelihood bound, using the chunk of documents as evaluation corpus. Also output the calculated statistics.

In our daily life, there are lots of text generated from every area, like communication, production, etc. Then follows a lot of text- mining algorithms to do the clustering or classification. Before talking about the very details in text-mining algorithm, from human beings' perspective, every text can be stored as a document and every document is a bag-of-words. In this way, every document d can be seen as d= (w_1, w_2, w_3, ..., w_n), w is the word in the document sequentially.

“reference”: the purpose for constructing text modeling is to observe how to generate the words sequence of the corpus. So, the statistic is also called "God's Game", where all the corpus generated by human beings can be seen as the results of every time God toss the die, we only get the results of the game by God- words sequence made up of the corpus, and the whole procedure of tossing die is absolutely a black box for us. During the modeling, we hope to predict how God play this game. In more details, we need to solve two problems:

1. What kinds of dies used by god;
2. How to toss these dies by god.

And the first question is to define how many parameters/ hyper-parameters for this model; and the second one is the rules to generate the word sequence.

LDA (Latent Dirichlet Discrimination) is one of the most popular algorithm in text-mining area, where the key point is that the creator thinks that every document is combined with several topics of different weights. And for every topic, it is made up with similar words of probability distribution, like when we think of topic "computer science", we may come up with words like: "object", "memory", "proof" , "induction" , etc. Here we can see that there are two relationships: document to topic, and topic to word. And the tossing procedure can be repeated as two steps: first is to get the topics from documents, and for every topic, we keep tossing and get words one by one. Initially, randomly assign topics to each word in the document, and then count the term frequency under each topics and the inverse document frequency for this topic, for each loop calculations, it will exclude the current topic distribution, and assign the probability of other topics according to the other topic distributions. After we get all of the topic distributions, we will assign this sample a new topic. We will do this procedure recursively until it converges, which means to minimize the similarity of the topics and minimize the log-perplexity difference.
[7]Here we use Gensim LDA library to help do the modeling. Initially, set the hyper-parameter alpha and eta default to a symmetric 1.0/(number of topics prior). Here, alpha and eta are hyper-parameters that affect sparsity of the document-topic (theta) and topic-word(lambda) distributions.
Here we calculate and log perplexity estimate to set as a threshold , which means when the log-perplexity of two neighbor recursion are the same, then it converges and stops.

3.logging file

Using logging file to do tracking, and show every details and recalls through the program. To some extent, information recorded in the logging file is like text-expression of the workflow. Especially for a text mining project, logging file will be very important. The very first reason is that the Latent Dirichlet Allocation Model is like a black box, in which we tune hyper parameter, literately train the models until it converges. Using the logging file can help us do the recording and make sure every pass and iteration work, and get the parameters that we care most, like the number set for topics, we can check this manually and do the evaluation. In this way, we can transfer this black box into grey box, and finally into white box. And another reason for logging file is that we can do query with this file, cause if the logging file is too long, manually do the checking or tracking thing will be very tedious. So, if we can do the annotation and extract the knowledge which we need from the logging file, this will be very useful and valuable.

4.Querying YesWorkflow Models

4.1 Prospective Data Provenance Queries

There are two kinds of queries: the first is just list the whole workflow, and list the subset of workflow. The other is to find the dependency of each data. Given a data item, determine on which inputs it depends. And show the downstream of dependencies of the input data, as well as the upstream of the dependencies of the output results.

4.2 Inference of Retrospective Data Provenance

The difference between the perspective provenance outputs and the retrospective provenance outputs is that for perspective provenance, the outputs is what the programmer expects, However, for retrospective provenance, the outputs is the real outputs.
[6]And here I use the following prospective, retrospective queries for my use case:
- Q1: Render the workflow (sub-workflow) of the product X, where the upstream of the product X.
- Q2: List the workflow outputs and the inputs upstream of the final product X.
- Q3: Render the sub-workflow of the product X, where the downstream of the product X.
- Q4: List the dependencies of the outputs and downstream of the inputs.
- Q5: Render the recon file (reconstructed prospective model augmented by retrospective provenance) of the final products.
- Q6: Render the recon file (reconstructed prospective model augmented by retrospective provenance) of all runtime observables.

5. Use-case: topic modeling

5.1 Introduction

The most absorbing feature for Yesworkflow software is that it can fully summarize the whole details in the procedure, where people who are not the expert in this area will save more time understanding and apply in practice. Yesworkflow is much more than a black box, it constructs a bridge between the developers and the users. The reason that I apply this software in text mining area is that I find that topic modeling is also a kind of tool, which are used to summarize/cluster the text data. In this way, they share common features, reproducibility and DRY (don't repeat yourself), both are of high efficiency and using Yesworkflow Log file can help trace the topic modeling training procedure.

5.2 Topic Modeling

Nowadays, there are many trails from which people can obtain the very key events from the real worlds, whether they are text or non-text data. And mostly we get the very important events from the text data. Reading the whole bunch of text will cost a lot of time and so there occurs many tools to help people to get to where they want to reach. And topic modeling is one of the tools which are used to discover the topics, to improve the accuracy and efficiency of the topic modeling methods, [3]we need to consider three more factors:

1. Consider the related words
2. Consider the ambiguous words.
3. How to express the topics clearly and accurately, where we need to reduce the repeatability and using combined words to describe the topics.

All these three problems are about expressive and representative of the core meaning for summarizing the whole text data. We know that usually, the single word or the single existing word in the text data itself can not represent much about what it express actually.
So to solve these three problems, instead of setting topic as word by word, people use word distribution to represent topics. In details, people introduce word weights, and broaden the border from the current document to "background corpus",[5] this very "corpus" can be called as "generative model", which can be used to predict the probability of the words that we may observe the words.

5.3 YW(Yesworkflow) result Analysis

5.3.1 Run Yesworkflow on the topic_modeling.py script

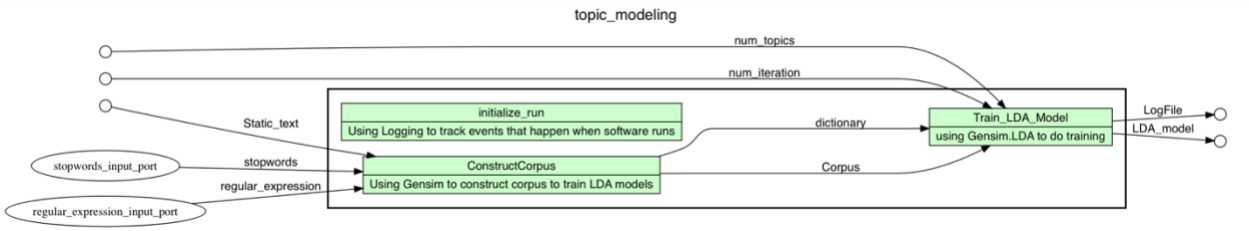

Figure 1. the whole workflow for topic modeling procedure

Analysis:

[4]The green blocks represent all of the stages in the computation performed by this script. The labels on the arrows, like “stopwords”, ”Static_text”, represent the input, intermediate, and final data products of the script.

Firstly, there are three inputs for stage “ConstructCorpus”: port “Static_text”, port “stopwords”, and port “regular_expression”. Then it generate intermediate two inputs for the next “Train_LDA_Model” stage, including “dictionary” and “Corpus”.

Secondly, for the “Train_LDA_Model” stage. Here it introduces the “outside” port: port “num_topics” and port “num_iteration”, combined with the former step outputs: “dictionary” and “Corpus”, finally, generate the LDA_model.

Obviously, there is another stage “initialize_run” , which seems that it is independent with other stage, but when we see the whole box and the outputs, we find that stage “initialize_run” and “Train_LDA_Model” are parallel level , and merge these two outputs product “LogFile”.

Using the YW running command, it clearly describes every input port and output port, input data and output product. It shows the dependencies between the data, the flow direction and to some extent, there will be no data-lose, and this achieves data reproductivity.

5.3.2 Topic_modeling case workflow (sub-workflow)

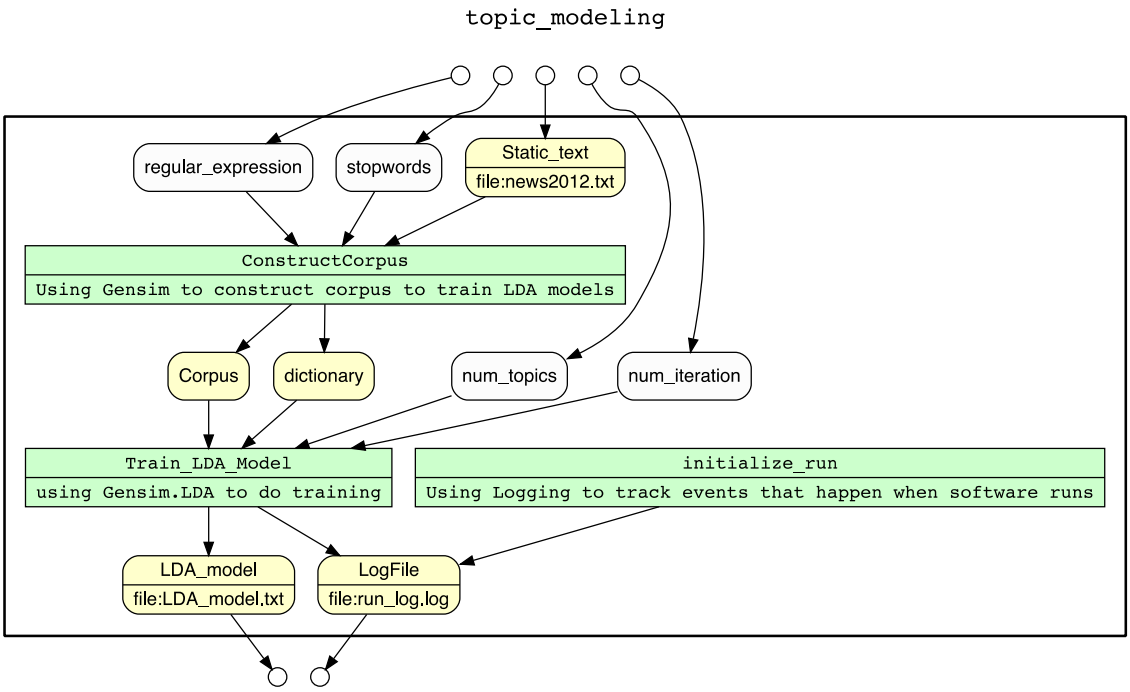

Figure 2. Complete Workflow with File URI (Uniform Resource Identifier)

There are three kinds of data formats above: @param which stands for the input parameter, and expressed with white box; @in which stands for the input data, and expressed with yellow box; @out which stands for the output products, and also expressed with yellow box.
The green box stands for the stages for the whole workflow.
This vertical workflow is pretty similar to the horizontal one above. However, the difference is that

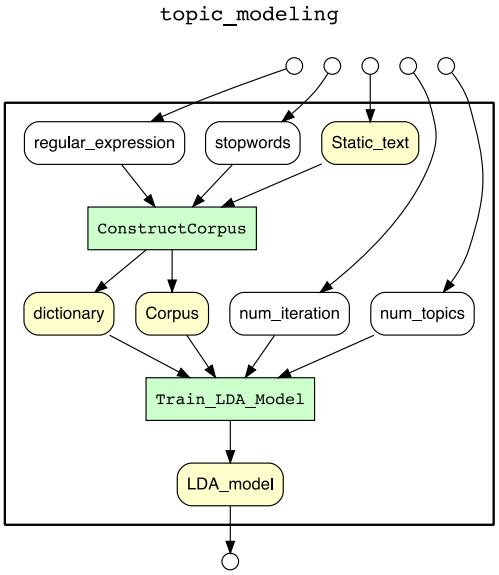

Figure 3. sub-workflow for LDA_model

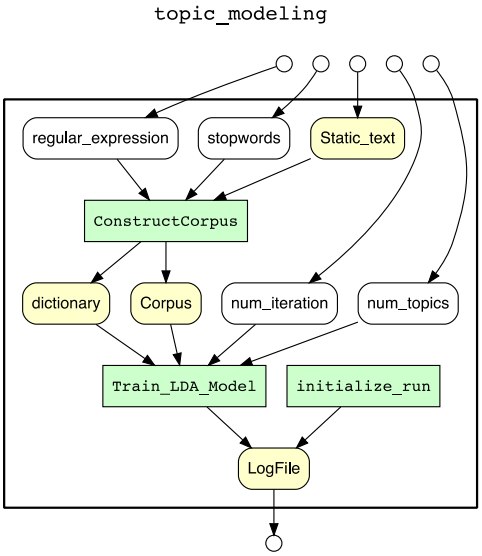

Figure 4. Sub-workflow for LogFile

Analysis:
Here, we use query to list the upstream/downstream of the output/input data, the YW will analysis the comment in the script, and traverse the whole procedure as well as tracking the data flowing path.
First is about the output product LDA_model, from the above, we know that this model is based on two stages: Construct the corpus and then Train the LDA model; and there are two hyper-parameters: number of iteration and number of topics, combined with the output data from the first stage: dictionary and corpus. These four parameters directly influence the final output LDA model. And if we continue to find upstream indirect parameters: “dictionary” and “Corpus” are produced by the first stage: “ConstructCorpus”, thus, there are three more indirect parameters influence the final product, including stopwords, regular expression and Static_text.
Second is about the output product LogFile, from the above, we know that this product record two stages: “Train_LDA_Model” and “initialize_run”; here in the initialize_run, I use @Log tag to record the time stamp and important parameter: log perplexity mentioned previously. This log file can help users know clearly how this training model works, and the whole procedure during the training.

5.3.3 Recon File for topic_modeling workflow (sub-workflow)

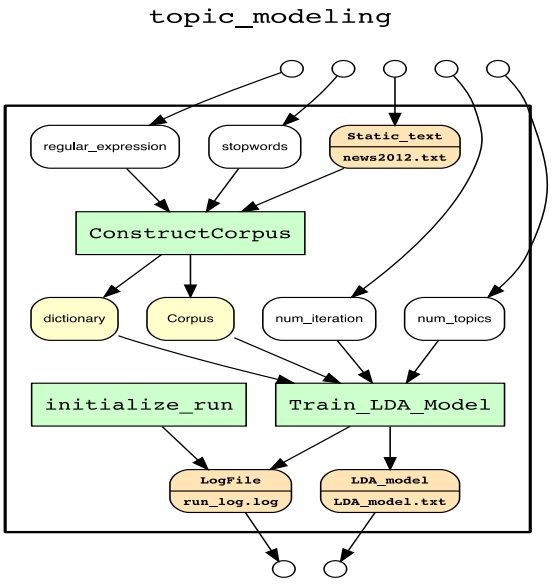

Figure 5. Recon complete workflow

Analysis:

This recon file (reconstructed retrospective provenance) shows that the whole program has been perfectly generating the output products, where YW has found all of the uri.

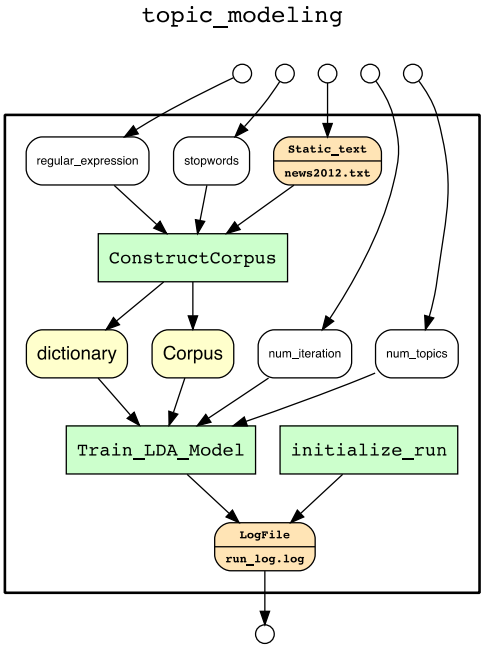

Figure 6. Recon LogFile sub-workflow

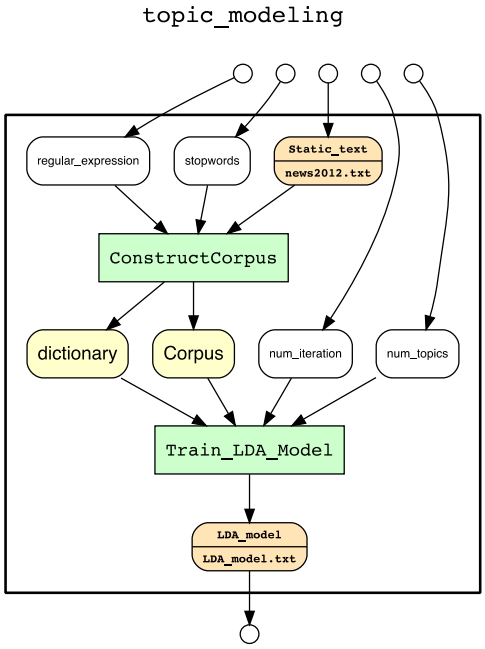

Figure 7. Recon LDA_model sub-workflow

Analysis:
These two recon sub-workflows capture the data dependency on the actual input file. Both are coming from the Static text file.

6.Conclusion

In this independent study, I manage to combine the YW (Yesworkflow) with topic modeling, where using the YW to do prospective provenance and retrospective provenance querying. The predefined queries help me to record the lineage of the data, and capture the data dependency on the actual input data files. The generated workflow can help me understand much more clearly for the whole procedure. Especially for the modeling part, I use YW log tag to help record the change of parameters, which can represent the complete training process.
I learnt a lot from Professor Bertram and his team, they give me a lot of help and help me understand clearly about the Yesworkflow toolkit, the deep cognition on provenance of the data.

Reference

[1] T. McPhillips, T. Song, T. Kolisnik, S. Aulenbach, K. Belhajjame, R.K. Bocinsky, Y. Cao, J. Cheney, F. Chirigati, S. Dey, J. Freire, C. Jones, J. Hanken, K.W. Kintigh, T.A. Kohler, D. Koop, J.A. Macklin, P. Missier, M. Schildhauer, C. Schwalm, Y. Wei, M. Bieda, B. Ludäscher (2015). YesWorkflow: A User-Oriented, Language-Independent Tool for Recovering Workflow Information from Scripts. International Journal of Digital Curation 10, 298-313.
[2] Ludäscher, Bertram. A Brief Tour Through Provenance in Scientific Workflows and Databases. In Building Trust in Information, edited by Victoria L. Lemieux, 103–26. Springer Proceedings in Business and Economics, 2016. doi:10.1007/978-3-319-40226-0_7.
[3] Chengxiang Zhai, Sean Massung. Text Data Management and Analysis A Practical Introduction to Information Retrieval and Text Mining, 2016.
[4] https://github.com/yesworkflow-org/yw-prototypes
[5] Chengxiang Zhai, Sean Massung. Text Data Management and Analysis A Practical Introduction to Information Retrieval and Text Mining, 2016.doi: 10.1145/2915031.2915048
[6] Qian Zhang, Yang Cao, Qiwen Wang, Duc Vu, Priyaa Thavasimani, Timothy McPhillips, Paolo Missier, Peter Slaughter, Christopher Jones, Matthew B.Jones. Revealing the Detailed Lineage of Script Outputs Using Hybrid Provenance,2018.
[7] https://radimrehurek.com/gensim/corpora/dictionary.html