YesWorkflow cases- topic modeling in text mining area

Friday, April 27, 2018

10:04 PM

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 tracing 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.

Keyword: YesWorkflow, ProvToolbox , Query, topic modeling

2.text mining

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. Then to filter the extremes which is between 0.5-20. 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=(w1, w2, w3,.., wn), 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.

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, iteratelly 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.

5. use-case : topic\_modeling

1.Yesworkflow annotation.

[1]YesWorkflow: A User-Oriented, Language-Independent Tool

for Recovering Workflow Information from Scripts

[2]A Brief Tour through Provenance in Scientific Workflows and Databases