Technology Review
CS 410 Text Information Systems
LDA

Introduction:

In natural language processing, the latent Dirichlet allocation (LDA) is a generative statistical model that allows sets of observations to be explained by unobserved groups that explain why some parts of the data are similar. LDA is an example of a topic model and belongs to the machine learning toolbox and in wider sense to the artificial intelligence toolbox. In other words, LDA is a generative probabilistic model for natural texts. It is used in problems such as automated topic discovery, collaborative filtering, and document classification.

Following Tech reviews the Apache MADlib LDA implementation. Apache MADlib is an open-source library for scalable in-database analytics. It provides data-parallel implementations of mathematical, statistical, graph and machine learning methods for structured and unstructured data.

Body:

The LDA model posits that each document is associated with a mixture of various topics (e.g., a document is related to Topic 1 with probability 0.7, and Topic 2 with probability 0.3), and that each word in the document is attributable to one of the document's topics. There is a (symmetric) Dirichlet prior with parameter \(\alpha \) on each document's topic mixture. In addition, there is another (symmetric)

Dirichlet prior with parameter β on the distribution of words for each topic.

The following generative process then defines a distribution over a corpus of documents:

- Sample for each topic i, a per-topic word distribution ϕ_i from the Dirichlet (β) prior.
- For each document:
    - Sample a document length N from a suitable distribution, say, Poisson.
    - Sample a topic mixture θ for the document from the Dirichlet (α) distribution.

    - For each of the N words:
        - Sample a topic z_n from the multinomial topic distribution θ.
        - Sample a word w_n from the multinomial word distribution ϕ_{z_n} associated with topic z_n.

In practice, only the words in each document are observable. The topic mixture of each document and the topic for each word in each document are latent unobservable variables that need to be inferred from the observables, and this is referred to as the inference problem for LDA. Exact inference is intractable, but several approximate inference algorithms for LDA have been developed.

This implementation provides a parallel and scalable in-database solution for LDA based on Gibbs sampling. It takes advantage of the shared-nothing MPP architecture and is a different implementation than one would find for MPI or map/reduce.

Training Function

The LDA training function has the following syntax:

```
lda_train( data_table,
        model_table,
        output_data_table,
        voc_size,
        topic_num,
        iter_num,
        alpha,
        beta,
        evaluate_every,
        perplexity_tol
      )
```

Arguments:

data_table - Name of the table storing the training dataset. Each row is in the form <docid, wordid, count> where docid, wordid, and count are non-negative integers.

model_table - This is an output table generated by LDA which contains the learned model.

output_data_table - The name of the table generated by LDA that stores the output data.

voc_size - Size of the vocabulary.

topic_num - Desired number of topics.

iter_num - Maximum number of iterations. If a 'perplexity_tol' is set, LDA may train for less than the maximum number of iterations if the tolerance is reached.

alpha - Dirichlet prior for the per-document topic multinomial

beta - Dirichlet prior for the per-topic word multinomial

evaluate_every (optional) - default: 0. How often to evaluate perplexity. Set it to 0 or a negative number to not evaluate perplexity in training at all. Evaluating perplexity can help you check convergence during the training process, but it will also increase total training time.

perplexity_tol (optional) - default: 0.1. Perplexity tolerance to stop iteration. Only used when the parameter 'evaluate_every' is greater than 0.

Conclusion:

This implementation provides a parallel and scalable in-database solution for LDA based on Gibbs sampling.
It takes advantage of the shared-nothing MPP architecture and is a different implementation than one would find for MPI or map/reduce.
This library can be used with databases like PostgreSQL and also MPP (Massively Parallel processing) Analytical database like Greenplum database.

Also, in addition to the training function mentioned above, this LDA implementation provides few helper functions that are helpful to interpret the output from LDA training and LDA prediction.

Topic modeling is often used as part of a larger text processing pipeline, which may include operations such as term frequency, stemming and stop word removal. Apache MADlib also provides a utility function **Term Frequency** to generate the required vocabulary format from raw documents for the LDA training function.

References:

https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation
https://madlib.apache.org/docs/latest/group__grp__lda.html