Feature Proposal

Latent Dirichlet Allocation for Dask-ML

Christina Lino

1. Context

When dealing with big data, programmers are tasked with writing machine learning algorithms that can automate the process of finding patterns and discovering trends. Python is often used to handle big data because of its accessibility as a high-level programming language and its associated third party libraries that facilitate machine learning. These libraries, such as scikit-learn (sklearn), NumPy, and pandas, focus on offering simple, high-level APIs and largely ignore performance. As datasets become increasingly larger and the machine learning algorithms become more complex, current machine learning libraries are not able to handle the influx of data. Scalable alternatives must be developed. One such alternative called Dask is a free, open source software that aims to parallelize Python and its associated libraries.

Dask is a parallel computing library developed in coordination with the pandas, NumPy and sklearn communities. A subcomponent of Dask called Dask-ML contains modules that scale similarly to sklearn for training and prediction on large models and datasets. Although Dask-ML aims to extend upon the functionality of sklearn, the library has yet to parallelize the full scope of algorithms offered by the API. Dask-ML is missing the Latent Dirichlet Allocation (LDA) algorithm. LDA is a generative probabilistic model for collections of discrete data, such as text corpora. This means that given a set of text documents, the LDA generates a set of topics, and determines if the words in the given documents match the topics.

2. Problem

When conducting research, users want the ability to analyze large swaths of text. LDA is used in legal document searches, content recommendation, search engine optimization, and more. It is used in academic settings to analyze topics across studies, and assess gaps in knowledge based on which topics do (or do not) appear). LDAs are applied in a wide range of use cases because they can automatically annotate unstructured datasets with topics, creating labeled data for supervised learning at scale.

Because the LDA is applied in many use cases, the LDA algorithm should be available as a feature for users to implement and analyze increasingly large datasets. The sklearn LDA is the perfect candidate for parallelization because the LDA provides more accurate results when fed long, cohesive articles. Recent research conducted by Jonnsson and Stolee found that the LDA algorithm is only effective in extracting topics from texts when the documents are at least a few hundred words long.

Researchers often pass tweets and Reddit posts into the sklearn LDA. Social media posts are often short and contain irrelevant detail that gets cleaned, leaving a small portion of the original post for analysis. The LDA was originally designed to work with news articles and book chapters, but existing versions of the LDA in sklearn cannot handle larger datasets. Parallelization would allow users to have the full power of the LDA. The LDA algorithm produces more accurate results when provided more information, and Dask-ML is designed to process high volumes.

3. Solution

The user will be able to interact with the Dask-ML LDA algorithm in a similar manner to the sklearn LDA. The user will have limited interaction with the LDA algorithm. There will be a number of parameters they can pass in to fine-tune the algorithm to their needs. Users can initialize the LDA, run it, and record the results of the algorithm.

Fit(), transform(), fit_transform(), get_params(), score(), and set_params() are all public methods that the user will need. Fit() and fit_transform() are standard for most scikit-learn algorithms that scale the training data and learn the scaling parameters of the data. The fit() method calculates mean and variance of data passed in. The fit_transform() method allows the algorithm to use the same mean and variance that

was calculated on the training data for the test data. Get_parameters() returns the parameters passed into the LDA. Score() returns the approximate log-likelihood of the model.

After it's finished running, the LDA should return a transformed array, where each entry in the array is a list with the probability distributions for each word in the document. The probability distribution is the likelihood that the term is in a topic. The LDA will be added to the The backend of the LDA should be implemented with the Dask Client for parallelization.

4. Deliverables

Sprint 1: Research

- Gain familiarity with probability and statistics concepts for LDA
- Research how new estimator models can be integrated into the Dask client
- Outline development process

Sprint 2: Implement and integrate the LDA

- Write fit(), transform(), and fit_transform() functionality
- Rewrite sklearn LDA methods with parallel implementation
- Integrate with existing Dask-ML environment

Sprint 3: Test and review code

- Conduct code review
- Write comprehensive, automated unit tests
- Demo product towards end of sprint

5. Conclusion

Dask-ML is a library aimed at parallelizing the sklearn models, but it has yet to include the full functionality of the popular machine learning API. One such missing model, the LDA algorithm, should be parallelized and included in Dask-ML. LDA should be a priority because it requires datasets full of full-length articles and papers to run, and can be used in a variety of settings.

6. References

Jonsson, E. ias, Stolee, J. (n.d.). An Evaluation of Topic Modelling Techniques for Twitter

Hoffman, M., Bach, F., Blei, D. (2010). Online learning for latent dirichlet allocation. advances in neural information processing systems, 23, 856-864.

Geletta S, Follett L, Laugerman M. Latent Dirichlet Allocation in predicting clinical trial terminations. BMC Med Inform Decis Mak. 2019 Nov 27;19(1):242. doi: 10.1186/s12911-019-0973-y. PMID: 31775737; PMCID: PMC6882341.

Schwaber, K., Sutherland, J. (2020). The 2020 Scrum Guide. Scrum Guide — Scrum Guides. Retrieved November 18, 2021, from https://scrumguides.org/scrum-guide.html.