Text Classification with Parallel Latent Dirichlet Allocation

Angelica Feng, Judy Kong

INTRODUCTION

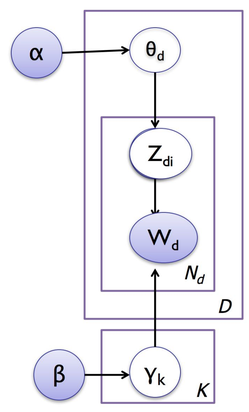
Latent Dirichlet Allocation (LDA) is a widely used algorithm in text classification, which clusters word occurrences into latent classes (i.e. topics) after iterations of parameter learning. As the sampling process is extremely computation-heavy for large data sets, we implemented parallel LDA in this project and experimented with its performance.

METHOD

The William Randolph Hearst Foundation will give $1.25 million to Lincoln Center, Metropolitan Opera Co., New York Philharmonic and Juilliard School. "Our board felt that we had a real opportunity to make a mark on the future of the performing arts with these grants an act every bit as important as our traditional areas of support in health, medical research, education and the social services," Hearst Foundation President Randolph A. Hearst said Monday in announcing the grants. Lincoln Center's share will be $200,000 for its new building, which will house young artists and provide new public facilities. The Metropolitan Opera Co. and New York Philharmonic will receive $400,000 each. The Juilliard School, where music and the performing arts are taught, will get $250,000. The Hearst Foundation, a leading supporter of the Lincoln Center Consolidated Corporate Fund, will make its usual annual $100,000 donation, too.

"Arts" "Budgets" "Children" "Education"

Figure 1. (above) *LDA's view of a document*
Figure 2. (right) *LDA as a graphical model*



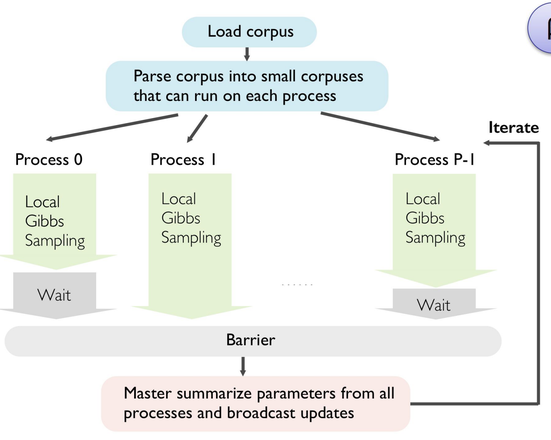


Figure 3. (left)
Parallel LDA workflow: split corpus by document; each process runs local Gibbs sampling; master gathers updates and send back to each process

IMPLEMENTATION

We implemented two variations of parallelism LDA - the synchronous version and the asynchronous version. We also added staleness in our implementation.

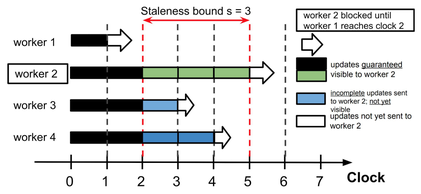


Figure 4. *Synchronous Parallel LDA (updates broadcasted to all processes)*

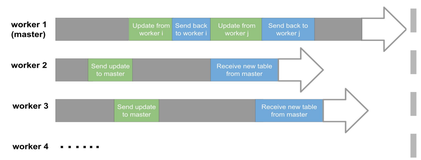


Figure 5. *Asynchronous Parallel LDA (each process send local updates to master after s iters, keep on sampling, and updates upon receiving new table)*

RESULT

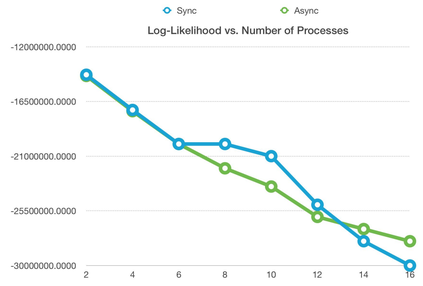


Figure 6. *Log-Likelihood vs. # of Processes*

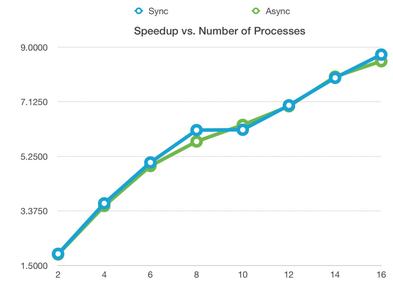


Figure 7. *Speedup vs. # of Processes*

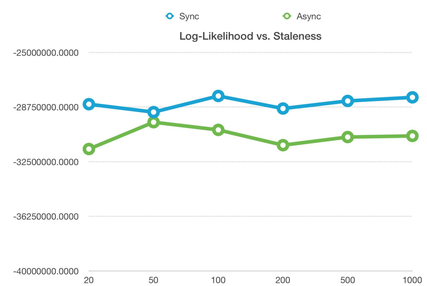


Figure 8. *Log-Likelihood vs. Staleness*

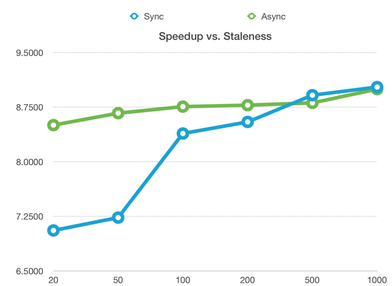


Figure 9. *Speedup vs. Staleness*

DISCUSSION

➡ How does the parallelism scale

Close to linear speedup until ~6 procs, gets smaller after but still achieves 9x with 16 procs.

➡ Tradeoff between log-likelihood & speed

Learned parameters will be more "stale" as the staleness gets larger. This gets us faster speed, but sometimes might decrease the objective function (log-likelihood) as a tradeoff.

➡ Speedup comparison of sync vs. async

We expected async version to be faster, but since there's not much work imbalance in sync version, and for async the master takes on more work updating the tables based on messages from each process, the speedup for both are about the same.

CONCLUSION

Our implementation of parallel Latent Dirichlet Allocation achieves an overall good speedup compared to the sequential version. For future work, better asynchronous update rules might be designed for better work balance to further improve performance.

REFERENCE

Blei, David M., Andrew Y. Ng, and Michael I. Jordan. "Latent dirichlet allocation." Journal of machine Learning research 3.Jan (2003): 993-1022.
Newman, David, et al. "Distributed algorithms for topic models." Journal of Machine Learning Research 10.Aug (2009): 1801-1828.
10-605 Lecture Notes by William W. Cohen