Speed Up Topic Modeling: Distributed Computing and Convergence Detection for LDA

Kohei Watanabe

Waseda University

May 2023

Topic models have been widely used by political scientists, but the high computational costs of the algorithms often prevent them from analyzing large corpora without aggressive data simplification. Therefore, I have developed the *seededlda* package that has distributed computing and convergence detection mechanisms, both of which dramatically speed up Latent Dirichlet allocation (LDA) by processing small chunks of data in parallel or by terminating Gibbs sampling when topic assignment becomes stable. Through systematic evaluation of the algorithms using a corpus 10,000 news articles, I demonstrate that they not only jointly reduce the time for identifying 100 topics from over one hour to about five minutes but also produce results very similar to that of the standard LDA. I argue that these faster algorithms are useful in not only identifying topics of documents but also generating document vectors as low dimensional representation of their content for other computational analysis.

Introduction

Topic models have been widely used by political scientists to analyze textual data. The algorithms can automatically identify clusters of words as topics in many domains and languages, but it is widely recognized by the users that the algorithms are computationally expensive: Latent Dirichlet allocation (LDA) usually take much longer time to perform the tasks than simpler algorithms such as Latent Semantic Analysis (Deerwester et al., 1990); variants of LDA are usually even more computationally expensive due to the additional parameters that need to be estimated.[1] The long execution time often forces users to perform aggressive feature selection before fitting models, potentially leading to biased results (Denny & Spirling, 2018).

There are at least two ways to speed up topic modeling without aggressive data simplification: the first is employing distributed computing, in which data is split into smaller chunks and allocated to multiple processors; the second is applying convergence detection, with which the iterative Gibbs sampling can be interrupted when parameters become sufficiently stable. Both are very effective ways to speed up topic modeling, but there are only a few software programs that implement these algorithms.

I have implemented distributed computing and convergence detection algorithms in the *seededlda* package to allow political scientists to apply topic modeling to a large corpus of texts. My systematic evaluation shows that the distributed computing alone makes topic modeling significantly faster; the convergence detection also cuts the execution time more than the half; despite the dramatic speed up, the results produced with these algorithms are very similar to that of the standard LDA (Heinrich, 2008).

Algorithms

I have implemented a distributed LDA algorithm that dramatically speeds up topic modeling when multiple processors are available, but it performs Gibbs sampling in the same sequence as the standard LDA algorithm when only a single processor is used. It is also general enough to fit the variants of LDA such as Seeded LDA (Lu et al., 2011), which can classify texts into pre-defined topics, and Sequential LDA (Du et al., 2012), which can classify individual sentences of texts. Following the notation in Watanabe & Baturo (2023), I explain the algorithm in concise manner with the aid of the pseudocode (Figure 1). I implemented the algorithm in C++ using the Intel Thread Building Block (TBB) library for parallel computing.

Distributed Computing

Among several distributed LDA algorithms that have been proposed, the Approximately Distributed LDA (Newman et al., 2009) is chosen for its simplicity and generalizability. It assigns topics of words in the same way as the standard LDA but splits data into smaller chunks and performs Gibbs sampling on multiple processors in parallel.

In inferring the document-topic distribution θ and topic-word distribution ϕ, topics assigned by the Gibbs sampler are saved in M_{dk} and N_{kv}. The former is the frequency of topic k to be found in document d; the latter is the frequency of topic k to be assigned to unique word v; α and β are the hyper-parameters for smoothing:

$$\theta_{dk} = \mathrm{P}(Z = k|d) = \frac{M_{dk} + \alpha}{M_{d.} + \alpha|K|}$$

$$\phi_{kv} = \mathrm{P}(W = v|k) = \frac{N_{kv} + \beta}{N_{.v} + \beta|V|}$$

While N_{kv} and M_{dk} are global variables shared by all the processors, each subprocess $p = \{1,2,\cdots,P\}$ has a local variable to record topics assignment $\acute{N}_{kvp}$. In every 10 iterations, these local counts are added to the global variable to synchronize the inference between processors.

Convergence Detection

Due to the difficulty in detecting the convergence of parameters in Gibbs sampling (Gelman & Rubin, 1992), it is common to set a fixed number of iterations in topic modeling. The numbers should be high enough (usually between 1,000 and 3,000) to make the topics in ϕ distinctive from each other, but the iteration could be interrupted earlier if topic assignment become stable enough. There are multiple methods to measure quality of topics (e.g., perplexity and divergence), they are unsuitable for this purpose due to its computational costs.

I propose the *delta statistic* as an efficient convergence detection criterion for LDA. It measures stability of topic assignment by comparing current and previous topics $\delta_i = |z_j \neq z_{j-1}|$ and continues Gibbs sampling as long as the statistic is decreasing $\delta_i \leq \delta_{i-1}$.[2] Since the statistic tend to fall quickly in the first a few hundred iterations, it can reduce the computational cost even if a single processor is available. In parallel computing, the local delta $\acute{\delta}_{ip} = |z_j \neq z_{j-1}|$ can be computed in the last sub-iteration $j = 10$ and added to the global delta δ_i to detect convergence.

[2] After convergence, δ tend to fluctuate around zero because words that only have weak semantic association with others receive different topics each time.

for $(1 \leq i \leq max\_iter/10)$ {
  assign $D \times batch\_size$ documents to processor p
  parallel_for $(1 \leq j \leq 10)$ {
    sample topic for words in the batch: $z_j \leftarrow \text{gibbs\_sample}(N_{kv}, M_{dk}, \alpha, \beta)$
    if $(j = 10)$ {
      count topic changes: $\acute{\delta}_{ip} \leftarrow |z_j \neq z_{j-1}|$
    }
    return $\acute{N}_{kvp}, \acute{\delta}_{ip}$
  }
  synchronize topic-word count: $N_{kv} \leftarrow \acute{N}_{kv1} + \acute{N}_{kv2} + \cdots + \acute{N}_{kvP}$
  aggregate topic changes: $\delta_i \leftarrow \acute{\delta}_{i1} + \acute{\delta}_{i2} + \cdots + \acute{\delta}_{iP}$
  if $(\delta_i \leq \delta_{i-1})$ {
    exit
  }
}

Figure 1: Pseudocode of the proposed algorithms. Topics are sampled max_iter times if is convergence detection is disabled; $batch\_size$ determines the sizes of the batches; D is the total number of documents in the corpus.

Evaluation

I evaluated the distributed computing and the convergence detection mechanisms in terms of how much they can speed up topic modeling and how much they change the result of inference by fitting models on a sample of 10,000 the Guardian newspapers articles.[3] I fitted the models with 100 topics on a computer using between 1 and 16 physical processors, recorded their execution time and compared their results with that of the standard LDA model.[4] The convergence detection mechanism and the sequential classification algorithm (Sequential LDA) were enabled or disabled; when the sequential algorithm is enabled, the documents are 391,395

[3] I preprocess the textual data using the quanteda package (Benoit et al., 2018). I segmented texts based on the ICU rules; removed punctuation, symbols, numbers, email addresses; remove grammatical words and single letters; remove tokens that occur less than 10 times in the corpus. The resulting document-feature matrix contains 4.7 million tokens (27,532 unique types).

[4] I used Ubuntu 22.04 on Microsoft Azure Virtual Machine (F16s v2, 16 CPUs, 32 GB RAM).

sentences from the articles; Gibbs sampling was limited to 1,000 iterations; a small number of seed words are given to the models to align topics for direct comparison.[5] In addition to these, I fitted benchmark models by disabling the distributed computing and the convergence detection.

Figure 2 shows that the average execution time is around 3,300 seconds in the non-sequential models and 4,500 seconds in the sequential models when only one processor is used, but it becomes around 7 times shorter as the number of processors increases to 8. The rate of speed up decreases after 8 processors, but the execution time becomes 8 times shorter than the original in both models when all the 16 processors are used. When the convergence detection is enabled, the execution time becomes an additional 2 to 3 times shorter in the non-sequential model, but it does not affect the sequential models.

Figure 3 illustrates the average Jansen-Shannon divergence between corresponding topic vectors in the ϕ parameter (topic-word association) of the fitted models. Despite the much shorter execution time, the divergence scores are consistently around 0.14 in the sequential models and 0.10 in the non-sequential models. Importantly, the divergence scores are roughly the same even when Gibbs sampling is terminated earlier by the convergence detection mechanism in the non-sequential models. This suggests that the results produced by the faster models are very similar to those by the benchmark models.[6]

---

[5] I fitted the standard LDA to identify 100 topics and extracted top 10 words from each topic as seed words. These seed words add very little computational costs, but they align the topics in all the fitted models for direct comparison between them.

[6] These divergence scores are also much lower than the distribution of non-self-pairs of topics, which ranges between 0.32 and 0.65, in the benchmark models.

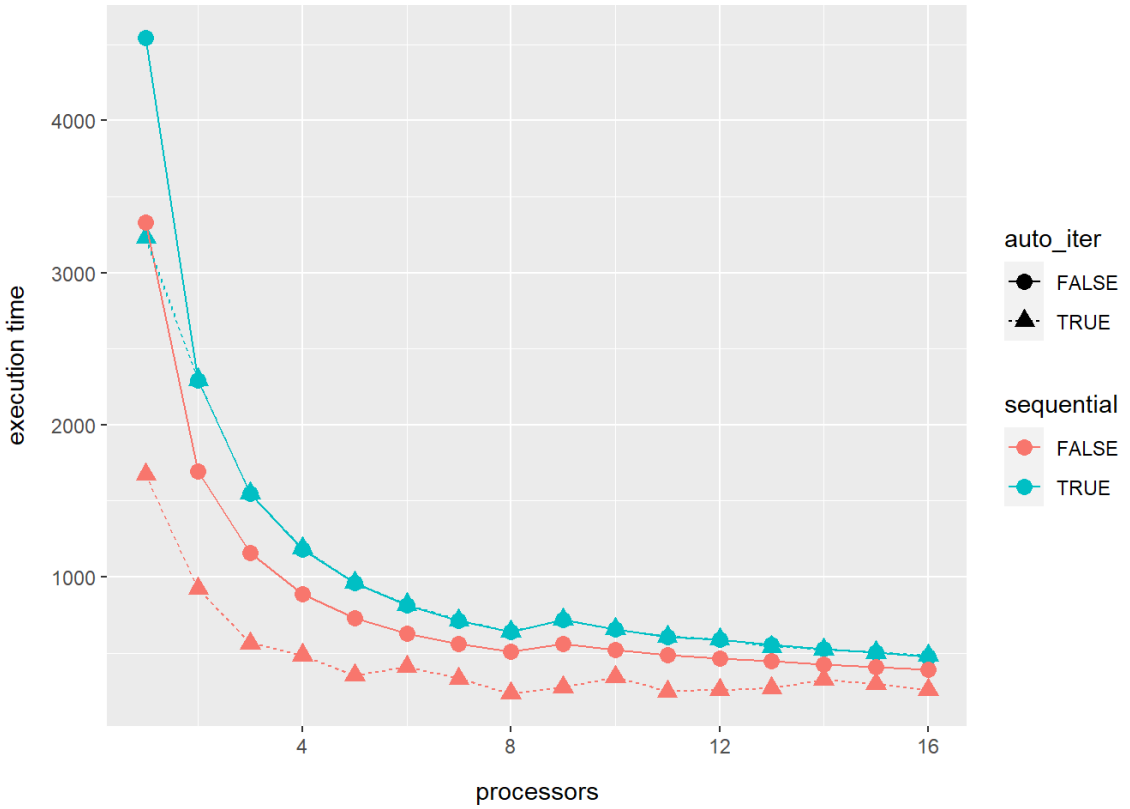

Figure 2: The average execution time for sequential and non-sequential models. Sequential classification is enabled when "sequential" is true (blue). Convergence detection is enabled when "auto_iter" is true (triangles). The average execution time is computed by fitting the same model 10 times.

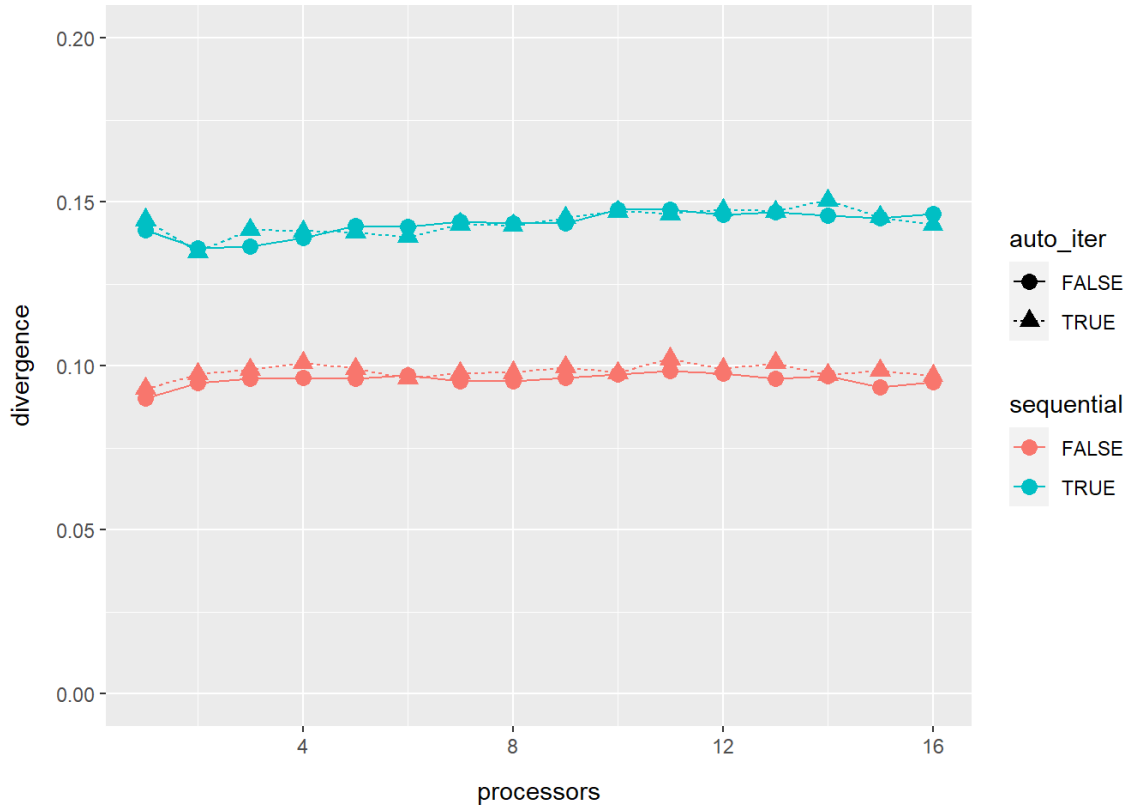

Figure 3: The average Jansen-Shannon divergence of the ϕ parameter. Sequential classification is enabled when "sequential" is true (blue). Convergence detection is enabled when "auto_iter" is true (triangles). Divergence is computed by repeating the same model X times and comparing with the ϕ parameter from the standard LDA model. The average divergence is computed by fitting the same model 10 times.

Conclusions

I have demonstrated that distributed computing and convergence detection can slash LDA's execution time to identify 100 topics in a corpus of 10,000 news articles from over one hour to about five minutes. This dramatic speed up allows users to perform topic modeling on large and complex corpora without oversimplification of the data though aggressive feature selection. I have also shown that, despite the dramatic reduction in the execution time, topics in the models are very similar to those in the standard models. This allows users to replace the standard models with the proposed models without the fear of producing invalid results in topic modeling.

The distributed algorithm accelerated roughly proportionally to the number of processors only until it became eight. This is because the execution time within the parallel-for loop became smaller compared to the overall execution time, and the overhead involving the parallel computing such as synchronization and task scheduling became dominant. However, models with larger number of topics would be benefited from the additional processors because a larger K increases the execution time within the sub-processes.

The distributed algorithm is also useful in searching the optimal number of topics based on the divergence measure proposed by Deveaud et al. (2014) because the algorithm has almost no effect on the divergence of topics within the model. User should fit LDA models with varying K to maximize the average divergence score in this optimization, but they are recommended to disable convergence detection to gain a smooth curve.

The sequential and non-sequential models are affected by the convergence detection mechanism differently when the maximum number of iterations is limited to 1,000. This difference can be explained by the greater complexity of the sequential models, which would require more than 1,000 iterations to converge. This suggests that the best practice with the sequential models is setting the maximum number of iterations to 2,000 or more while enabling convergence detection to avoid excessive iterations.

Distributed computing is particularly beneficial in fitting the sequential model because it can generate sentence vectors as a low dimensional representation of their content, which could be used for other text analysis purposes (e.g., supervised machine learning). Depending on the complexity of the data, the number of topics for sentence vectors would be much greater than 100 but the distributed algorithm is the most efficient with a large K.

I have implemented the proposed algorithms in the *seededlda* package and published it on CRAN for easy access, but users should be aware that the distributed computing requires physical processors instead of virtual processors. Although modern processors often have multithreading capabilities (e.g., Intel Hyper-Threading), only an increase in the number of physical processors can accelerate topic modeling. Further, although I split the data into chunks that contain about 1% of the documents (i.e., each contains 100 articles or 3,913 sentences), the batch size should be larger for smaller corpora to ensure that enough documents are allocated to sub-processes to infer topics locally.

Finally, the distributed algorithm implemented in the package achieved speed up equivalent to the level reported by Newman et al. (2007), but it could be even faster if the overhead involving synchronization and task scheduling is minimized. More specifically, the frequency of the synchronization could be less frequent, and the sizes of the batches could be smaller to make the overall process faster, but these settings may impact the quality of the topic inference. This should be investigated in future works.

References

Benoit, K., Watanabe, K., Wang, H., Nulty, P., Obeng, A., Müller, S., & Matsuo, A. (2018). quanteda: An R package for the quantitative analysis of textual data. *Journal of Open Source Software*, *3*(30), 774. https://doi.org/10.21105/joss.00774

Deerwester, S. C., Dumais, S. T., Landauer, T. K., Furnas, G. W., & Harshman, R. A. (1990). Indexing by latent semantic analysis. *JASIS*, *41*(6), 391–407.

Denny, M. J., & Spirling, A. (2018). Text Preprocessing For Unsupervised Learning: Why It Matters, When It Misleads, And What To Do About It. *Political Analysis*, *26*(2), 168–189. https://doi.org/10.1017/pan.2017.44

Deveaud, R., Sanjuan, E., & Bellot, P. (2014). Accurate and Effective Latent Concept Modeling for Ad Hoc Information Retrieval. *Revue Des Sciences et Technologies de l'Information - Série Document Numérique*, 61–84. https://doi.org/10.3166/DN.17.1.61-84

Du, L., Buntine, W., Jin, H., & Chen, C. (2012). Sequential latent Dirichlet allocation. *Knowledge and Information Systems*, *31*(3), 475–503. https://doi.org/10.1007/s10115-011-0425-1

Gelman, A., & Rubin, D. B. (1992). Inference from Iterative Simulation Using Multiple Sequences. *Statistical Science*, *7*(4), 457–472.

Heinrich, G. (2008). *Parameter estimation for text analysis*. http://www.arbylon.net/publications/text-est.pdf

Lu, B., Ott, M., Cardie, C., & Tsou, B. K. (2011). Multi-aspect sentiment analysis with topic models. *2011 IEEE 11th International Conference on Data Mining Workshops*, 81–88.

Newman, D., Asuncion, A., Smyth, P., & Welling, M. (2009). Distributed Algorithms for Topic Models. *The Journal of Machine Learning Research*, *10*, 1801–1828.

Newman, D., Smyth, P., Welling, M., & Asuncion, A. (2007). Distributed Inference for Latent Dirichlet Allocation. *Advances in Neural Information Processing Systems*, *20*, 1081–1088.

Watanabe, K., & Baturo, A. (2023). Seeded Sequential LDA: A Semi-Supervised Algorithm for Topic-Specific Analysis of Sentences. *Social Science Computer Review*. https://doi.org/10.1177/08944393231178605