Distributed MAP Inference for Undirected Graphical Models

Sameer Singh[§] [*] **Amarnag Subramanya**[†] **Fernando Pereira**[†] **Andrew McCallum**[§]
[§] Department of Computer Science, University of Massachusetts, Amherst
[†] Google Research, Mountain View

1 Introduction

Graphical models have widespread uses in information extraction and natural language processing. Recent improvements in approximate inference techniques [1, 2, 3, 4] have allowed exploration of dense models over a large number of variables. These applications include coreference resolution [5, 6], relation extraction [7], and joint inference [8, 9, 10]. But as the graphs grow to web scale, even these inference techniques become infeasible, motivating the need for scalable, parallelizable solutions.

In this work, we distribute the MCMC-based MAP inference using the Map-Reduce framework [11]. The variables are assigned randomly to machines, which leads to some factors that neighbor variables on separate machines. Parallel MCMC-chains are initiated using proposal distributions that only suggest local changes such that factors that lie across machines are not examined. After a fixed number of samples on each machine, we redistribute the variables amongst the machines to enable proposals across variables that were assigned to different machines.

To demonstrate the distribution strategy on a real-world information extraction application, we model the task of cross-document coreference resolution. Given noun phrase mentions from a large document corpus, the problem is to identify clusters of these mentions such that mentions in a cluster refer to the same latent entity. Graphical model representations of this task have set-valued random variables and factors between all pairs of mentions. Scalability results using a random redistribution strategy show improved performance when increasing the number of machines. We also explore a hierarchical representation of our model. Experiments show that the hierarchical model converges much faster than the flat model, even though it contains many more latent random variables.

Related studies include Gonzalez et al. [12], where message passing is distributed by efficiently splitting the graph to minimize communication. This method is not applicable when the graph structure changes with every configuration, or the ground graph[1] is fully connected. Furthermore, set-valued variables are difficult to incorporate into message passing. Asuncion et al. [14] and Smola and Narayanamurthy [15] distribute MCMC-based inference for topic models. Our approach is similar, but we do not calculate probabilities (marginals). This allows us to specify non-random redistribution and customized proposal functions, which can lead to faster convergence. Low et al. [16] introduce the *GraphLab* library that distributes computation when specified on nodes and edges. It is not straightforward to express our model as computations on nodes and edges because of set-valued variables and hierarchical models. In comparison to the above approaches, we allow the most flexibility in specifying structure, redistribution strategy, and the proposal function.

[*]This work was done when the author was an intern at Google Research.

[1]In our work, as in much of the previous work [13, 4], the model is specified implicitly. The *ground* graph is the result of instantiating the implicit specification fully into a possibly very large graph-structured model.

2 Proposed Method

Undirected graphical models can be used to represent a probability distribution over output variables Y given observed variables x. When represented as a *factor graph*, this distribution is given by $p(y|x) \propto \exp \sum_{y_c \subseteq y} \psi_c(y_c)$, where y is an assignment to Y, y_c is the neighborhood of factor c, and ψ_c is the factor potential for the assignment y_c. Note that the set of factors (i.e. the graph structure) may be different for every assignment to Y. In most real-world applications, a small change to the assignment y involves changes only to the local factors.

Given this representation of the distribution, we are interested in calculating the *maximum a posteriori* (MAP) configuration, i.e. configuration with the highest probability according to the model: $\hat{y} = \arg\max_y p(y|x) = \arg\max_y \exp \sum_{y_c \subseteq y} \psi_c(y_c)$.

2.1 MCMC-based MAP Inference

With the exception of some simple models, computing $\hat{y}$ exactly is intractable due to the exponential number of configurations. In such cases, MCMC-based sampling can be used to discover the MAP configuration. Here a proposal function q is used to propose a small change y' to the current configuration y. This jump is accepted with the following Metropolis-Hastings acceptance probability:

$$\alpha(y, y') = \min \left(1, \left(\frac{p(y')}{p(y)}\right)^{1/t} \frac{q(y)}{q(y')}\right); \quad \frac{p(y')}{p(y)} = \exp\left\{\sum_{y'_c \subseteq y'} \psi_c(y'_c) - \sum_{y_c \subseteq y} \psi_c(y_c)\right\} \tag{1}$$

where t is a temperature parameter. Note that if the difference between configurations y and y' is small, for most models the set of factors instantiated for the two configurations will have a high overlap, i.e. only a few factors are required to compute Eq. 1.

MCMC chains efficiently explore the high-density regions of the probability distribution. By reducing the temperature, we can decrease the entropy of the distribution to encourage convergence to the MAP configuration. This MAP inference technique offers several advantages – (a) we do not need the full ground graph, as would be required for most message-passing-based techniques, (b) we do not examine all the factors for a single configuration to evaluate a proposal but only the ones that are not common to the two configurations, (c) inference can be stopped at any point to obtain the current best configuration, which can be used to trade-off accuracy with time, and (d) there exist efficient training algorithms such as SampleRank [4, 17] that are used for parameter estimation *during* inference.

2.2 Parallelizing MAP Inference

The key observation to enable distribution is that the acceptance probability computation for a jump only examines a few factors that are not common to the previous and next configurations (Eq 1). If jumps are proposed such that the factors used to evaluate them are mutually exclusive, then these jumps can be proposed and evaluated simultaneously. Using this insight, we distribute variables amongst multiple machines, and propose only those jumps that can be evaluated using factors present on a single machine[2]. To enable exploration of the complete configuration space, rounds of sampling are interleaved by *redistribution* stages, where the variables are randomly redistributed amongst the workers. We use the Map-Reduce [11] framework as the underlying parallelization architecture, where in the *map* stage each worker performs inference in parallel on its variables, and the *reduce* stage redistributes the variables amongst the workers.

This approach to distribution is similar to inference with all variables on a single machine, but is faster since it exploits independencies. By restricting the jumps as described above, the acceptance

[2]Note that this restriction does not require that all the neighbors of a variable v for the ground graph be on the same machine as v; it only requires that the neighbors of v whose factor potentials are affected for a specific change to v be present on the same machine.

probability calculation is exact and the same as it is on a single machine. Partitioning the variables and proposing local jumps are restrictions to the single-machine proposal distribution, however redistribution stages ensure the equivalent Markov chains are still irreducible. Calculating the forward-backward ratio is difficult and is required when estimating probabilities [14, 15]; however we set it to 1.

To attain good mixing between the parallel MCMC chains, we use a random redistribution step. This can result in slow convergence since unrelated variables may be assigned to the same machine. Since the MAP inference does not rely on good mixing amongst the parallel chains, customized redistribution methods that utilize domain knowledge can be employed to speed-up convergence. Futhermore, redistribution can also be encoded as additional latent variables in the model, an example of which will be described in Section 3.2.

3 Cross-document Coreference

For evaluation on a real-world task, we consider the problem of cross-document coreference resolution. This involves partitioning noun phrase mentions in a collection of documents into clusters such that mentions within each cluster refer to the same underlying entity. Cross-doc coreference is used in many downstream information extraction applications [18, 19]. Within-document coreference resolution is a closely related problem where the goal is to resolve noun phrases (and referring expressions) within a document [20] and is not the focus of our work here.

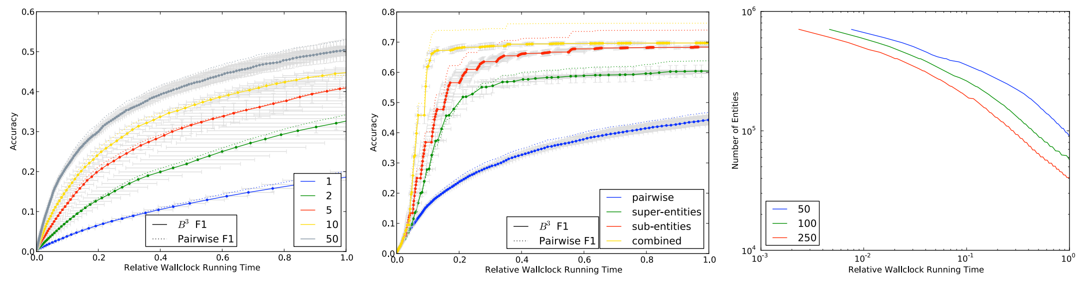
Previous approaches to cross-document coreference have used clustering with a context-similarity based distance function[21, 22, 23, 24, 25]. These approaches are greedy and differ in the choice of the distance function and the clustering algorithm used. Rao et al. [26] have proposed an online deterministic method that uses a stream of input mentions and assigns them to entities incrementally. Daumé III and Marcu [27] proposed a generative approach to supervised clustering.

Representing the problem as an undirected graphical model provides a combination of advantages not available in other approaches. First, most of the methods do not scale to the millions of mentions that are present in real-world applications. Approaches that use clustering are limited to using pairwise distance functions for which additional supervision and features are difficult to incorporate. Also, many of the approaches are greedy and do not revisit earlier decisions. We address these problems by applying our method of distributed inference to a graphical model, whose parameters can be learned using supervised or semi-supervised techniques. However, in this work we only address the problem of MAP inference, and leave learning for future work.

3.1 Pairwise Model

Our model for cross-document coreference contains mentions ($\vec{M}$) and entities ($\vec{E}$) as random variables, where each mention variable can take an entity as its value, and each entity takes a set of mentions as its value. Each mention also has a set of features associated with the observed text mention. The probability of a configuration (partitioning) $\vec{E} = \vec{e}$ is defined by $p(\vec{e}) \propto \exp\left\{ \sum_{m_i \sim m_j} \psi_a(m_i, m_j) - \sum_{m_i \nsim m_j} \psi_r(m_i, m_j) \right\}$, where ψ_a represent *affinity* between mentions that are coreferent according to $\vec{e}$, while ψ_r represents *repulsion* between mentions that are not coreferent. Thus different factors are instantiated for different configurations. We use cosine similarity of mention context pairs (ϕ_{ij}) as the factor potentials such that $\psi_a(m_i, m_j) = \phi_{ij} - b$ and $\psi_r(m_i, m_j) = \phi_{ij} - b$, where b is a bias factor.

For MAP inference on a single machine, we start with the initial configuration of all singleton entities. The proposal function moves a random mention to a random entity. To score the proposal, only the factors that neighbor mentions in the previous and current entity need to be scored. In the distributed inference case, we can exploit this to ensure that entities are not divided across machines, and the proposals move mentions between entities on the same machine.



(a) Varying the Number of Machines (b) Hierarchical Models (c) Large-Scale Inference

Figure 1: **Distributed Inference for Cross-Document Coreference**: (a) Person-X Evaluation for the pairwise model on 25k mentions, (b) Hierarchical models for Person-X on 50 machines, and (c) Number of predicted entities for 1 million mentions

3.2 Hierarchical Model

When applying cross-document coreference to a real-world task, the number of mentions and the entities is typically very large, which can lead to inefficiencies that slow inference. We mention two of the challenges, and attempt to address them by adding *hierarchies* to the model.

Consider the task of proposing moves. Given a large number of mentions and entities, the probability of randomly picking a mention and moving it to its correct entity is very small. If such a move is made, ideally we would also like to move a group of similar mentions simultaneously, instead of randomly discovering these moves. We introduce latent *sub-entity* variables that represents groups of similar mentions within an entity, where the similarity is defined by the model. For inference, we have stages of sampling sub-entities (moving individual mentions) interleaved with stages of entity sampling (moving sub-entities).

Another problem with the distributed pairwise model is that random redistribution is wasteful. For a large number of entities and machines, the probability that similar entities will be assigned to the same machine is very small. To alleviate this problem, we introduce *super-entities* that represent multiple similar entities. During redistribution, we ensure all entities in the same super-entity are assigned to the same machine. As in sub-entities above, inference switches between regular sampling of entities and sampling of super-entities.

Note that each of the individual levels of the hierarchy are similar to our original model, e.g. mentions/subentities have the same structure as the entities/super-entities, and are modeled using similar factors. To represent the "context" of a sub-entity we take the union of the bag-of-words of the constituent mention contexts. Similarly, we take the union of sub-entity contexts to represent the context of an entity. The factors are instantiated same as in Sec 3.1 except that we change the bias factor b for each level (increasing it for sub-entities, and decreasing it for super-entities).

Since these two levels of hierarchy are independent of each other, we can combine them into a single hierarchical model that contains both sub- and super-entities. The inference for this model takes a round-robin approach by fixing two of the levels of the hierarchy and sampling the third, cycling through the three levels.

4 Experiments

There is a severe lack of labeled corpora for cross-document coreference due to the effort required for humans to evaluate the coreference decisions. Related approaches have used automated *Person-X* evaluation [23], where unique person-name strings are treated as the true entity labels for the mentions. Each mention string is then replaced with an "X" before being sent to the coreference resolution system. We use this evaluation by taking 25k person-name mentions from the New York Times corpus [28] with 50 unique mention strings. For inference, we use rounds of 1 million samples each, followed by random redistribution of entities in the flat pairwise model, and super-entities in the hierarchical model. Results are averaged over five runs.

Fig 1a shows F1 accuracy according to two metrics (B^3 and Pairwise) compared to relative wall clock running time for the pairwise model. Performance improves as additional machines are added, but larger number of machines lead to diminishing returns for this small dataset. The hierarchical model is evaluated in Fig 1b against the pairwise model from 1a. We see that the individual hierarchical models perform better than the pairwise model; moreover, the combined model outperforms both of the individual hierarchical models.

To explore the application of the proposed approach to a larger dataset, we use the million most frequently occurring person-name strings in the New York Times. Given the nature of such data, we expect the number of true entities to be a few orders of magnitude smaller than 10^6. Since our model is initialized such that each mention forms its own cluster, the number of predicted entities by a model can be used as a measure of its rate of convergence. The plot in Fig 1c shows the number of predicted entities against the running time (both in log-scale). Note that the initial number of predicted entities is 10^6. It can be seen that using more machines leads to improved rate of convergence, demonstrating utility of our proposed approach.

References

[1] Alexander M Rush, David Sontag, Michael Collins, and Tommi Jaakkola. On dual decomposition and linear programming relaxations for natural language processing. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1–11, Cambridge, MA, October 2010. Association for Computational Linguistics.

[2] Andre Martins, Noah Smith, Eric Xing, Pedro Aguiar, and Mario Figueiredo. Turbo parsers: Dependency parsing by approximate variational inference. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 34–44, Cambridge, MA, October 2010. Association for Computational Linguistics.

[3] Hoifung Poon, Pedro Domingos, and Marc Sumner. A general method for reducing the complexity of relational inference and its application to MCMC. In *AAAI Conference on Artificial Intelligence*, 2008.

[4] Andrew McCallum, Karl Schultz, and Sameer Singh. FACTORIE: Probabilistic programming via imperatively defined factor graphs. In *Neural Information Processing Systems (NIPS)*, 2009.

[5] Hoifung Poon and Pedro Domingos. Joint unsupervised coreference resolution with markov logic. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 650–659, 2008.

[6] Aron Culotta, Michael Wick, and Andrew McCallum. First-order probabilistic models for coreference resolution. In *North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL HLT)*, 2007.

[7] Sebastian Riedel, Limin Yao, and Andrew McCallum. Collective cross-document relation extraction without labelled data. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2010.

[8] Hoifung Poon and Pedro Domingos. Joint inference in information extraction. In *AAAI Conference on Artificial Intelligence*, pages 913–918, 2007.

[9] Sameer Singh, Karl Schultz, and Andrew McCallum. Bi-directional joint inference for entity resolution and segmentation using imperatively-defined factor graphs. In *Machine Learning and Knowledge Discovery in Databases (Lecture Notes in Computer Science) and European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*, pages 414–429, 2009.

[10] Jenny R. Finkel and Christopher D. Manning. Joint parsing and named entity recognition. In *North American Association of Computational Linguistics (NAACL)*, 2009.

[11] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. *Symposium on Operating Systems Design & Implementation (OSDI)*, 2004.

[12] Joseph Gonzalez, Yucheng Low, Carlos Guestrin, and David OHallaron. Distributed parallel inference on large factor graphs. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2009.

[13] Aron Culotta. *Learning and inference in weighted logic with application to natural language processing*. PhD thesis, University of Massachusetts, 2008.

[14] Arthur Asuncion, Padhraic Smyth, and Max Welling. Asynchronous distributed learning of topic models. In *Neural Information Processing Systems (NIPS)*, pages 81–88, 2009.

[15] Alexander J. Smola and Shravan Narayanamurthy. An architecture for parallel topic models. *International Conference on Very Large Data Bases (VLDB)*, 3(1):703–710, 2010.

[16] Yucheng Low, Joseph Gonzalez, Aapo Kyrola, Danny Bickson, Carlos Guestrin, and Joseph M. Hellerstein. Graphlab: A new parallel framework for machine learning. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, Catalina Island, California, July 2010.

[17] Michael Wick, Khashayar Rohanimanesh, Aron Culotta, and Andrew McCallum. Samplerank: Learning preferences from atomic gradients. In *Neural Information Processing Systems (NIPS), Workshop on Advances in Ranking*, 2009.

[18] J. Mayfield, D. Alexander, B. Dorr, J. Eisner, T. Elsayed, T. Finin, C. Fink, M. Freedman, N. Garera, P. McNamee, et al. Cross-document coreference resolution: A key technology for learning by reading. In *AAAI Spring Symposium on Learning by Reading and Learning to Read*, 2009.

[19] Matthias Blume. Automatic entity disambiguation: Benefits to NER, relation extraction, link analysis, and inference. In *International Conference on Intelligence Analysis (ICIA)*, 2005.

[20] Aria Haghighi and Dan Klein. Coreference resolution in a modular, entity-centered model. In *Human Language Technologies: Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL HLT)*, pages 385–393, 2010.

[21] Amit Bagga and Breck Baldwin. Entity-based cross-document coreferencing using the vector space model. In *International Conference on Computational Linguistics*, pages 79–85, 1998.

[22] A. Baron and M. Freedman. Who is who and what is what: experiments in cross-document coreference. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 274–283, 2008.

[23] Chung Heong Gooi and James Allan. Cross-document coreference on a large scale corpus. In *Human Language Technologies: Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL HLT)*, pages 9–16, 2004.

[24] A. Purandare and T. Pedersen. Word sense discrimination by clustering contexts in vector and similarity spaces. In *Conference on Computational Natural Language Learning (CoNLL)*, pages 41–48, 2004.

[25] Yael Ravin and Zunaid Kazi. Is Hillary Rodham Clinton the president? disambiguating names across documents. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 9–16, 1999.

[26] Delip Rao, Paul McNamee, and Mark Dredze. Streaming cross document entity coreference resolution. In *International Conference on Computational Linguistics (COLING)*, pages 1050–1058, Beijing, China, August 2010. Coling 2010 Organizing Committee.

[27] Hal Daumé III and Daniel Marcu. A Bayesian model for supervised clustering with the Dirichlet process prior. *Journal of Machine Learning Research (JMLR)*, 6:1551–1577, 2005.

[28] Evan Sandhaus. The New York Times annotated corpus. *Linguistic Data Consortium*, 2008.