Unsupervised Part of Speech Inference with Particle Filters

Gregory Dubbin and **Phil Blunsom**
Department of Computer Science
University of Oxford
Wolfson Building, Parks Road
Oxford, OX1 3QD, United Kingdom
Gregory.Dubbin@wolfson.ox.ac.uk Phil.Blunsom@cs.ox.ac.uk

Abstract

As linguistic models incorporate more subtle nuances of language and its structure, standard inference techniques can fall behind. Often, such models are tightly coupled such that they defy clever dynamic programming tricks. However, Sequential Monte Carlo (SMC) approaches, i.e. particle filters, are well suited to approximating such models, resolving their multi-modal nature at the cost of generating additional samples. We implement two particle filters, which jointly sample either sentences or word types, and incorporate them into a Gibbs sampler for part-of-speech (PoS) inference. We analyze the behavior of the particle filters, and compare them to a block sentence sampler, a local token sampler, and a heuristic sampler, which constrains inference to a single PoS per word type. Our findings show that particle filters can closely approximate a difficult or even intractable sampler quickly. However, we found that high posterior likelihood do not necessarily correspond to better Many-to-One accuracy. The results suggest that the approach has potential and more advanced particle filters are likely to lead to stronger performance.

1 Introduction

Modern research is steadily revealing more of the subtle structure of natural language to create increasingly intricate models. Many modern problems in computational linguistics require or benefit from modeling the long range correlations between latent variables, e.g. part of speech (PoS) induction (Liang et al., 2010), dependency parsing (Smith and Eisner, 2008), and coreference resolution (Denis and Baldridge, 2007). These correlations make inference difficult because they reflect the complicated effect variables have on each other in such tightly coupled models.

Sequential Monte Carlo (SMC) methods, like particle filters, are particularly well suited to estimating tightly coupled distributions (Andrieu et al., 2010). Particle filters sample sequences of latent variable assignments by concurrently generating several representative sequences consistent with a model's conditional dependencies. The sequential nature of the sampling simplifies inference by ignoring ambiguous correlations with unsampled variables at the cost of sampling the sequence multiple times. The few applications of particle filters in computational linguistics generally focus on the online nature of SMC (Canini et al., 2009; Borschinger and Johnson, 2011). However, batch applications still benefit from the power of SMC to generate samples from tightly coupled distributions that would otherwise need to be approximated. Furthermore, the time cost of the additional samples generated by SMC can be mitigated by generating them in parallel.

This report presents an initial approach to the integration of SMC and block sampling, sometimes reffered to as Particle Gibbs (PG) sampling (Andrieu et al., 2010). Unsupervised PoS induction serves as a motivating example for future extensions to other problems. Section 3 reviews the PYP-HMM model used for PoS inference. Section 4 explains the Sequential Importance Sampling (SIS) algorithm, a basic SMC method that generates samples for the block sampler. This approach yields two implementations: a simple sentence-based block sampler (4.1) and a more complicated type-based sampler (4.2). Finally, section 5 evaluates both implementations on a variety of unsupervised PoS inference tasks, analyzing the behavior of the SMC inference and comparing them to state-of-the-art approaches.

2 Background

SMC was introduced in 1993 as a Bayesian estimator for signal processing problems with strong non-linear conditional dependencies (Gordon et al., 1993). Since then, SMC methods have been adopted by many fields, including statistics, biology, economics, etc. (Jasra et al., 2008; Beaumont, 2003; Fernandez-Villaverde and Rubio-Ramirez, 2007). The SMC approach is the probabilisitic analogue of the beam search heuristic, where the beam width can be compared to the number of particles and pruning is analogous to resampling.

The basic SMC approach serves as the basis for several variants. Many SMC implementations resample the population of particles to create a new population that minimizes the effect of increasing sample variance with increasing sequence length (Kitagawa, 1996). Particle smoothing variants of SMC reduce the relative variance of marginals early in the sequence, as well improving the diversity of the final sample (Fearnhead et al., 2008). Particle Markov chain Monte Carlo (PMCMC) formally augments classic Markov chain Monte Carlo (MCMC) approaches, like Gibbs sampling, with samples generated by particle filters (Andrieu et al., 2010).

3 The PYP-HMM

The PYP-HMM model of PoS generation demonstrates the tightly coupled correlations that complicate many standard inference methods (Blunsom and Cohn, 2011). The model applies a hierarchical Pitman-Yor process (PYP) prior to a trigram hidden Markov model (HMM) to jointly model the distribution of a sequence of latent word classes, $\mathbf{t}$, and word tokens, $\mathbf{w}$. This model performs well on corpora in multiple languages, but the lack of a closed form solution for the sample probabilities makes it a strong canditate for PG sampling. The joint probability defined by a trigram HMM is

$$P_\theta(\mathbf{t},\mathbf{w}) = \prod_{n=1}^{N+1} P_\theta(t_l|t_{n-1},t_{n-2})P_\theta(w_n|t_n)$$

where $N = |\mathbf{t}| = |\mathbf{w}|$ and the special tag \$ is added to the boundaries on the sentence. The model defines transition and emission distributions,

$$\begin{aligned} t_n|t_{n-1},t_{n-2},T &\sim T_{t_{n-1},t_{n-2}} \\ w_n|t_n,E &\sim E_{t_n} \end{aligned}$$

The PYP-HMM smoothes these distributions by applying hierarchical PYP priors to them. The hierarchical PYP describes a back-off path of simpler PYP priors,

$$\begin{aligned} T_{ij}|a^T,b^T,B_i &\sim \mathrm{PYP}(a^T,b^T,B_i) \\ B_i|a^B,b^B,U &\sim \mathrm{PYP}(a^B,b^b,U) \\ U|a^U,b^U &\sim \mathrm{PYP}(a^U,b^U,\mathrm{Uniform}). \\ E_i|a^E,b^E,C &\sim \mathrm{PYP}(a^E,b^E,C_i), \end{aligned}$$

where T_{ij}, B_i, and U are trigram, bigram, and unigram transition distributions respectively and C_i is either a uniform distribution (PYP-HMM) or a bigram character language model emission distribution (PYP-HMM+LM, intended to model basic morphology).

Draws from the posterior of the hierarchical PYP can be calculated with a variant of the Chinese Restaraunt Process (CRP) called the Chinese Restaurant Franchise (CRF) (Teh, 2006; Goldwater et al., 2006). In the CRP analogy, each latent variable in a sequence is represented by a customer entering a restaurant and sitting at one of an infinite number of tables. A customer chooses to sit at a table in a restaurant according to the probability

$$P(z_n = k|\mathbf{z}_{1:n-1}) = \begin{cases} \frac{c_k^- - a}{n-1+b} & 1 \le k \le K^- \\ \frac{K^- a + b}{n-1+b} & k = K^- + 1 \end{cases} \tag{1}$$

where z_n is the index of the table chosen by the nth customer to the restaurant, $\mathbf{z}_{1:n-1}$ is the seating arrangement of the previous $n-1$ customers to enter, c_k^- is the count of the customers at table k, and K^- is the total number of tables chosen by the previous $n-1$ customers. All customers at a table share the same dish, representing the value assigned to the latent variables. When customers sit at an empty table, a new dish is assigned to that table according to the base distribution of the PYP. To expand the CRP analogy to the CRF for hierarchical PYPs, when a customer sits at a new table, a new customer enters the restaurant representing the PYP of the base distribution.

4 Sequential Monte Carlo

While MCMC approximates a distribution as the average of a sequence of samples taken from the posterior of the distribution, SMC approximates a distribution as the importance weighted sum of several sequentially generated samples, called particles. This article describes two SMC samplers that jointly sample multiple tag assignments: a sentence based block sampler (`sent`) and a word type based block sampler (`type`). The basics of particle filtering are outlined below, while the implementation specifics of the `sent` and `type` particle filters are described in secions 4.1 and 4.2, respectively.

SMC is essentially the probabilistic analogue of the beam search heuristic. SMC stores P sequences, analogous to beam width, and extends each incrementally according to a proposal distribution q_n, similar to the heuristic cost function in beam search. Many particle filtering implementations also include a resampling step which acts like pruning by reducing the number of unlikely sequences.

We implemented Sequential Importance Sampling (SIS), detailed by Doucet and Johansen (2009), to approximate joint samples from the sentence and word type distributions. This approach approximates a target distribution, $\pi_n(x_{1:n}) = \frac{\gamma_n(x_{1:n})}{Z_n}$, of the sequence, $x_{1:n}$, of n random variables, that is $\gamma_n(x_{1:n})$ calculates the unnormalized density of $x_{1:n}$.

SIS initilizes each particle $p \in [1, P]$ by sampling from the initial proposal distribution $q_1(x_1^p)$, where x_n^p is the value assigned to the n-th latent variable for particle p. The algorithm then sequentially extends each particle according to the conditional proposal distribution $q_n(x_n^p|x_{1:n}^p)$, where $x_{1:n}^p$ is the sequence of values assigned to the first n latent variables in particle p. After extending a particle p, SIS updates the importance weight $\omega_n^p = \omega_{n-1}^p * \alpha_n(x_{1:n}^p)$. The weight update, defined as

$$\alpha_n(x_{1:n}) = \frac{\gamma_n(x_{1:n})}{\gamma_{n-1}(x_{1:n-1})q_n(x_n|x_{1:n-1})}, \qquad (2)$$

accounts for the discrepancy between the proposal distribution, q_n, and the target distribution, π_n, without normalizing over $x_{1:n}$, which becomes intractable for longer sequences even in discrete domains. The normalizing constant of the target distribution is approximately $Z_n \approx \sum_{p=1}^{P} \omega_n^p$ and the unnormalized density is $\gamma_n(x_{1:n}) \approx \sum_{p=1}^{P} \omega_n^p \text{if} x_{1:n}^p = x_{1:n}$. The particles can also be used to generate an unbiased sample from π_n by choosing a particle p proportional to its weight ω_n^p.

Andrieu et al. (2010) shows that to ensure the samples generated by SMC for a Gibbs sampler has the target distribution as the invariant density, the particle filter must be modified to perform a *conditional SMC update*. This means that the particle filter guarantees that one of the final particles is assigned the same values as the previous Gibbs iteration. Our implementation of the conditional SMC update reserves one special particle, 0, for which the proposal distribution always chooses the previous iteration's value at that site.

4.1 Sentence Sampling

The `sent` particle filter samples blocks of tag assignments $\mathbf{t}_{1:n}^S$ for a sentence, S, composed of tokens, $\mathbf{w}_{1:n}^S$. Sampling an entire sentence minimizes the risk of assigning a tag with a high probability given its local context but minimal probability given the entire sentence. Sentences can be sampled by ignoring table counts while sampling a proposal sentence, incorporating them after the fact with a Metropolis-Hastings acceptance test (Gao and Johnson, 2008). The Metropolis-Hastings step simplifies the sentence block particle filter further by not requiring the conditional SMC update.

While there is already a tractable dynamic programming approach to sampling an entire sentence based on the Forward-Backward algorithm, particle filtering the sentences PYP-HMM model should prove beneficial. For the trigram HMM defined by the model, the forward-backward sampling approach has time complexity in $O(NT^3)$ for a sentence of length N with T possible tag assignments at each site. Particle filters with P particles can approximate these samples in $O(NTP)$ time, which becomes much faster as the number of tags, T, increases.

Sampling of sentence S begins by removing all of the transitions and emitions in S from the table counts, $\mathbf{z}$, resulting in the table counts $\mathbf{z}^{-S}$ of tag assignments $\mathbf{t}^{-S}$ the values assigned to the variables outside of S. For each site index $n \in [1, N]$ in the sentence, the particle filter chooses the new tag assignment, $t_n^{S,p}$, for each particle $p \in [1, P]$ from the sentence proposal distribution,

$$q_n^S(t_n^{S,p}|\mathbf{t}_{1:n-1}^{S,p}) \propto P(t_n^{S,p}|t_{n-2}^{S,p}, t_{n-1}^{S,p}, \mathbf{t}^{-S}, \mathbf{z}^{-S}) \\ \times P(w_n^{S,p}|t_n^{S,p}, \mathbf{t}^{-S}, \mathbf{z}^{-S}, \mathbf{w}^{-S}).$$

After each new tag is assigned, the particle's weight is updated according to equation (2). The simplicity of the proposal density hints at the advantage of particle filtering over forward-backward sampling: it tracks only P histories and their weights rather than tracking the probability of over all possible histories. Once each particle has assigned a value to each site in the sentence, one tag sequence is chosen proportional to its particle weight, $\omega_N^{S,p}$.

4.2 Type Sampling

The type sampling case for the PYP-HMM is more complicated than the `sent` sampler. The long-range couplings defined by the hierarchical PYP priors strongly influence the joint distribution of tags assigned to tokens of the same word type (Liang et al., 2010). Therefore, the affects of the seating decisions of new customers cannot be postponed during filtering as in sentence sampling. To account for this, the `type` particle filter samples sequences of seating arrangements and tag assignments jointly, $\mathbf{x}_{1:n}^W = (\mathbf{t}_{1:n}^W, \mathbf{z}_{1:n}^W)$, for the word-type, W. The final table counts are resampled once a tag assignment has been chosen from the particles.

Tracking the seating arrangement history for each particle adds an additional complication to the `type` particle filter. The exchangeability of seating decisions means that only counts of customers are necessary to represent the history. Each particle represents both a tag sequence, $\mathbf{t}_{1:n}^{W,p}$, and the count deltas, $\mathbf{z}_{1:n}^{W,p}$. The count deltas of each particle are stored in a hash table that maps a dish in one of the CRF restaurants to the number of tables serving that dish and the total number of customers seated at those tables. The count delta hash table ensures that it has sufficient data to calculate the correct probabilities (per equation (1)) by storing any counts that are different from the base counts, $\mathbf{z}^{-W}$, and defering to the base counts for any counts it does not have stored.

At each token occurence n, the next tag assignment, $t_n^{W,p}$ for each particle $p \in [1, P]$ is chosen first according to the word type proposal distribution

$$q_n^W(t_n^{W,p}|\mathbf{t}_{1:n-1}^{W,p}, \mathbf{z}_{1:n-1}^{W,p}) \propto \\ P(t_n^{W,p}|c_n^{-2}, c_n^{-1}, \mathbf{t}_{1:n-1}^{-W,p}, \mathbf{z}_{1:n-1}^{-W,p}) \\ \times P(c_n^{+1}|c_n^{-1}, t_n^{W,p}, \mathbf{t}_{1:n-1}^{-W,p}, \mathbf{z}_{1:n-1}^{-W,p}) \\ \times P(c_n^{+2}|t_n^{W,p}, c_n^{+1}, \mathbf{t}_{1:n-1}^{-W,p}, \mathbf{z}_{1:n-1}^{-W,p}) \\ \times P(w_n^W|t_n^{W,p}, \mathbf{t}_{1:n-1}^{-W,p}, \mathbf{z}_{1:n-1}^{-W,p}, \mathbf{w}_{1:n-1}^{-W,p}).$$

In this case, $c_n^{\pm k}$ represents a tag in the context of site t_n^W offset by k, while $\mathbf{t}_{1:n-1}^{-W,p}$, $\mathbf{z}_{1:n-1}^{W,p}$, and $\mathbf{w}_{1:n-1}^{-W,p}$ represent the tag assignments, table counts, and word token values chosen by particle p as well as the values at all of the sites where a word token of type W does not appear. This proposal distribution ignores changes to the seating arrangement between the three transitions involving the site n. The specific seating arrangement of a particle is chosen after the tag choice, at which point the weights are updated by the result of equation (2). As with the `sent` sampler, once all of the particles have been sampled, one of them is sampled with probability proportional to its weight. This final sample is a sample from the true target probability.

As mentioned earlier, the sequence of particle approximations do not have the target distribution as invariant unless they use the conditional SMC update. Therefore, a the special 0 particle is automatically assigned the value from the prior iteration of the Gibbs sampler at each site n, though the proposal probability $q_n^W(t_n^{W,0}|\mathbf{t}_{1:n-1}^{W,p}, \mathbf{z}_{1:n-1}^{W,p})$ still has to be calculated to update the weight $\omega_n^{W,p}$ properly. This ensures that the `type` sampler has a non-zero probability of reverting to the prior iteration's sequence.

5 Experiments and Results

We take two approaches to evaluating the SMC based samplers. The first approach is an analysis of the samplers as inference algorithms. The samplers should tend to maximize the posterior likelihood of the model over iterations, eventually converging to the mode. Section 5.1 analyzes the particle filter based samplers with various numbers of particles in an effort to understand how they behave.

Then, section 5.2 evaluates each of the proposed approaches on PoS inference tasks from several languages. These results allow a practical comparison with other PoS inference approaches.

5.1 SMC Analysis

Before comparing the performance of the SMC block samplers to other inference methods, we wish to learn more about the approaches themselves. It is not clear how well the benefits of block sampling transfer to SMC based approaches. Both the `sent` and `type` samplers are novel approaches to computational linguistics, and many of their properties are unclear. For example, the samples generated from the particle filter should have a higher variance than the target distribution. If the variance is too high, the sampler will be slower to converge. While additional particles lower the relative variance, they also increase the run time linearly. It is possible that there is a threshold of particles necessary to ensure that some are high likelihood sequences, beyond which inference gains are minimal the additional computational expense is wasted. All of the experiments in this section were run on the Arabic corpus from the CoNLL-X shared language task, which is small enough to quickly experiment with these issues (Buchholz and Marsi, 2006).

The sentence based sampler, `sent`, samples from a distribution that can be exactly computed, facilitating comparisons between the exact sampler and the SMC approach. Figure 5.1 compares the posterior log-likelihoods of the `sent` sampler and the exact sentence sampler over 200 iterations. As expected, the likelihoods of the particle filters approach that of the exact sentence sampler as the number of particles increases from 25 to 100, which completely overlaps the performance of the exact sampler by the 50th iteration. This is impressive, because even with 99 additional sequences sampled (one for each particle) each iteration the SMC approach is still faster than the exact sampler. Furthermore, the Arabic tagset has only 20 distinct tags, while other data sets, e.g. the WSJ and Bulgarian, use tagsets more than twice as large. The particle filter, which is linear in the number of tags, should take twice as long per token

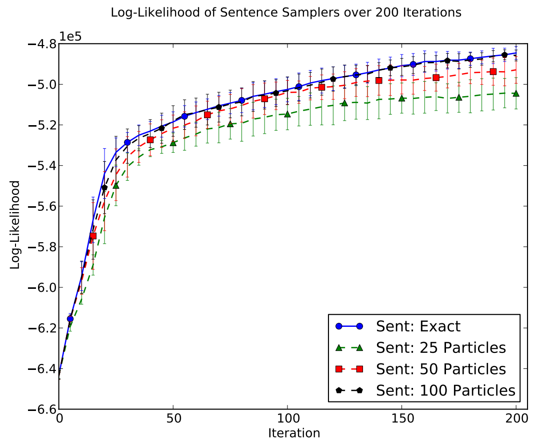

Figure 1: Posterior Log-Likelihood of PYP-HMM inference with exact as well as SMC sentence sampler with various numbers of particles. Error bars represent on standard deviation over three runs.

sampled on those data, relative to the arabic data. On the other hand, the forward-backward per token sample time, which is cubic in tagset size, should increase at least eightfold. So the time savings improve dramatically as the size of the tagset increases.

Figure 2 compares the table configuration log-likelihood of the 1HMM approximation implemented by Blunsom and Cohn (2011) with the `type` particle filter based sampler as well as the local, token-based sampler and the exact block sentence sampler. Unlike the sentence based block sampler, `type` sampler cannot be exactly calculated, even with the 1HMM approach of constraining inference to only consider sequences that assign the same tag to every token of the same word type. The 1HMM sampler approximates these probabilities using expected table counts. Theoretically, the `type` sampler should be a better approximation, being guaranteed to approach the true distribution as the number of particles increases. However, the `type` sampler does not constrain inference as the 1HMM does, slowing convergence by wasting particles on less likely tag sequences. As expected, the `type` sampler converges with the 1HMM sampler with sufficiently many particles in a few iterations. The exact block sentence sampler surpases approaches by iteration 75 and does not seem to have converged by the end of 200 iterations.

The local sampler samples a single site at a time with a standard, token-based Gibbs sampler. The

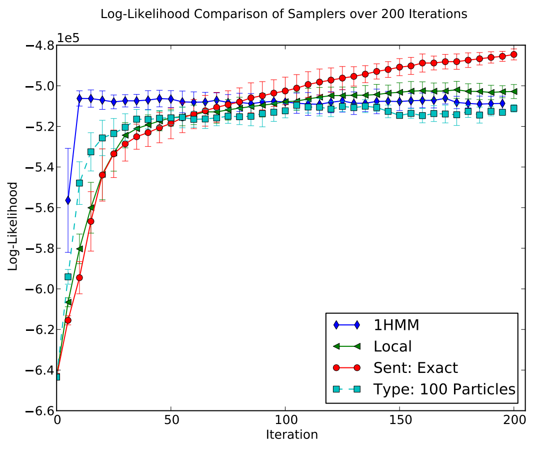

Figure 2: Posterior Log-Likelihood of PYP-HMM inference with particle filters, 1HMM approximation, and local samplers. Error bars represent one standard deviation over three runs.

local sampler performs surprisingly well, averaging a slightly higher likelihood than both the 1HMM and the `type` samplers. This may be an indication that the PYP-HMM model is not too tightly coupled for the local sampler to eventually migrate toward more likely modes. Note that both the `type` and 1HMM samplers initially take much larger steps, before eventually hitting a plateau. This suggests that some sort of mixed sampler may outperform its component samplers by only occasionally taking large steps.

5.2 Unsupervised Part-of-Speech Tagging

The samplers evaluated in section 5.1 induce syntactic categories analogous to PoS tags. However, to induce PoS tags, each syntactic category must be assigned to a PoS tag in the tagset. Each site in a corpus is assigned the most commonly visited syntactic category at that site over all iterations. The many-to-one (M-1) assignment for a category is the most common gold standard PoS tag of the tokens assigned to that category. While this assignment theoretically allows a perfect accuracy if each token is assigned to its own category, these experiments limit the number of induced categories to the size of the tagset. Table 1 compares the M-1 accuracy of the `sent` and `type` particle filter samplers, from sections 4.1 and 4.2, with 100 particles each. The particle filter based samplers rarely score a higher accuracy than even the local sampler, which completes 500 iterations before the particle filters complete 200.

While figure 2 shows that the sentence based block sampler eventually surpasses the 1HMM sampler in likelihood, the accuracies of the 1HMM and 1HMM-LM approximations remain well above the other approaches. The 1HMM sampler and the 100 particle `type` sampler have approximately the same likelihood, yet the M-1 accuracy of the 1HMM sampler is much higher. This suggests that there are high-likelihood assignments that produce lower accuracy results, presumably related to the fact that the `type` sampler is not restricted to assignments with exactly one tag for each word type. If the model assigns equal likelihood to these assignments, inference will not be able to distinguish between them. Perhaps a model that assigned higher likelihoods to tag sequences with fewer tags per word type would have a stronger correlation between likelihood and accuracy.

6 Future Work

While the results leave much room for improvement, the approach presented here is the most basic of particle methods. There has been considerable research in improvements to particle methods since their introduction in 1993 (Gordon et al., 1993). Two common approaches to improving particle filters are resampling and particle smoothing (Doucet and Johansen, 2009; Godsill and Clapp, 2001; Pitt, 2002). Resampling ensures that particles aren't wasted on unlikely sequences. Particle smoothing reduces the variability of the marginal distributions by combining the final particles.

7 Conclusion

This paper presented a preliminary approach to incorporating particle methods into computational linguistic inference applications. Such approaches show great potential for inference even in highly dependent distributions, but at a serious computational cost. However, the type particle filter itself can be largely run in parallel, only bottlenecking when the particle weights need to be normalized. Further expansion of the basic ideas presented will enable scalable inference in otherwise intractable models.

Language	Sent-100	Type-100	Local	1HMM	1HMM-LM	Tokens	Tag types
WSJ	69.8%	70.1%	70.2%	75.6%	77.5%	1,173,766	45
Arabic	53.5%	57.6%	56.2%	61.9%	62.0%	54,379	20
Bulgarian	64.8%	67.8%	67.6%	71.4%	76.2%	190,217	54
Czech	59.8%	61.6%	64.5%	65.4%	67.9%	1,249,408	12^c
Danish	65.0%	70.3%	69.1%	70.6%	74.6%	94,386	25
Dutch	61.6%	71.6%	64.1%	73.2%	72.9%	195,069	13^c
Hungarian	61.8%	61.8%	64.8%	69.6%	73.2%	131,799	43
Portuguese	59.4%	71.1%	68.1%	72.0%	77.1%	206,678	22

Table 1: Many-to-1 accuracies on CoNLL and Penn-Treebank Wall Street Journal corpora for sentence- (Sent) and type- (Type) based filtering. The table lists the average M-1 accuracy measured according to the maximum marginal tag assignments over 3 seperate runs after 200 iterations for the `sent`, `type`, 1HMM and 1HMM-LM samplers, and 500 iterations for the HMM local sampler. The 1HMM-LM model has been shown to achieve state-of-the-art unsupervised M-1 accuracies on these datasets. and thus represents the limit of unsupervised M-1 accuracy.

References

Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. 2010. Particle markov chain monte carlo methods. *Journal Of The Royal Statistical Society Series B*, 72(3):269–342.

Mark A. Beaumont. 2003. Estimation of Population Growth or Decline in Genetically Monitored Populations. *Genetics*, 164(3):1139–1160, July.

Phil Blunsom and Trevor Cohn. 2011. A hierarchical pitman-yor process hmm for unsupervised part of speech induction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 865–874, Portland, Oregon, USA, June. Association for Computational Linguistics.

Benjamin Borschinger and Mark Johnson. 2011. A particle filter algorithm for bayesian wordsegmentation. In *Proceedings of the Australasian Language Technology Association Workshop 2011*, pages 10–18, Canberra, Australia, December.

Sabine Buchholz and Erwin Marsi. 2006. Conll-x shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, CoNLL-X '06, pages 149–164, Morristown, NJ, USA. Association for Computational Linguistics.

Kevin R. Canini, Lei Shi, and Thomas L. Griffiths. 2009. Online inference of topics with latent Dirichlet allocation. In David van Dyk and Max Welling, editors, *Proceeings of the 12th International Conference on Artificial Intelligence and Statistics (AISTATS*09)*, pages 65–72.

Pascal Denis and Jason Baldridge. 2007. Joint determination of anaphoricity and coreference resolution using integer programming. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 236–243, Rochester, New York, April. Association for Computational Linguistics.

Arnaud Doucet and Adam M. Johansen, 2009. *A Tutorial on Particle Filtering and Smoothing: Fifteen Years Later*. Oxford University Press.

Paul Fearnhead, David Wyncoll, and Jonathan Tawn. 2008. A sequential smoothing algorithm with linear computational cost. Technical report, Department of Mathematics and Statistics, Lancaster University.

Jesus Fernandez-Villaverde and Juan F. Rubio-Ramirez. 2007. Estimating macroeconomic models: A likelihood approach. *Review of Economic Studies*, 74(4):1059–1087, October.

Jianfeng Gao and Mark Johnson. 2008. A comparison of bayesian estimators for unsupervised hidden markov model pos taggers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 344–352, Morristown, NJ, USA. Association for Computational Linguistics.

Simon Godsill and Tim Clapp. 2001. Improvement strategies for monte carlo particle filters. In A. Doucet, J. F. G. de Freitas, and N.J. Gordon, editors, *SEQUENTIAL MONTE CARLO METHODS IN PRACTICE*, pages 139–158. Springer-Verlag.

Sharon Goldwater, Tom Griffiths, and Mark Johnson. 2006. Interpolating between types and tokens by estimating power-law generators. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 459–466. MIT Press, Cambridge, MA.

N. J. Gordon, D. J. Salmond, and A. F. M. Smith. 1993. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *Radar and Signal Processing, IEE Proceedings F*, 140(2):107–113, April.

A. Jasra, A. Doucet, D. Stephens, and C. Holmes. 2008. Interacting sequential Monte Carlo samplers for transdimensional simulation. *Computational Statistics & Data Analysis*, 52(4):1765–1791, January.

G Kitagawa. 1996. Monte carlo filter and smoother for non-gaussian nonlinear state space models. *Journal Of Computational And Graphical Statistics*, 5(1):1–25.

P. Liang, M. I. Jordan, and D. Klein. 2010. Type-based MCMC. In *North American Association for Computational Linguistics (NAACL)*.

Michael K Pitt. 2002. Smooth particle filters for likelihood evaluation and maximisation. The Warwick Economics Research Paper Series (TWERPS) 651, University of Warwick, Department of Economics, July.

David A. Smith and Jason Eisner. 2008. Dependency parsing by belief propagation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 145–156, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yee Whye Teh. 2006. A hierarchical bayesian language model based on pitman-yor processes. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 985–992, Morristown, NJ, USA. Association for Computational Linguistics.