Using Rejuvenation to Improve Particle Filtering for Bayesian Word Segmentation

Benjamin Börschinger[*†]
benjamin.borschinger@mq.edu.au

Mark Johnson[*]
mark.johnson@mq.edu.au

[*]Department of Computing
Macquarie University
Sydney, Australia

[†]Department of Computational Linguistics
Heidelberg University
Heidelberg, Germany

Abstract

We present a novel extension to a recently proposed incremental learning algorithm for the word segmentation problem originally introduced in Goldwater (2006). By adding *rejuvenation* to a particle filter, we are able to considerably improve its performance, both in terms of finding higher probability and higher accuracy solutions.

1 Introduction

The goal of word segmentation is to segment a stream of segments, e.g. characters or phonemes, into words. For example, given the sequence "youwanttoseethebook", the goal is to recover the segmented string "you want to see the book". The models introduced in Goldwater (2006) solve this problem in a fully unsupervised way by defining a generative process for word sequences, making use of the Dirichlet Process (DP) prior.

Until recently, the only inference algorithm applied to these models were batch Markov Chain Monte Carlo (MCMC) sampling algorithms. Börschinger and Johnson (2011) proposed a strictly incremental particle filter algorithm that, however, performed considerably worse than the standard batch algorithms, in particular for the Bigram model. We extend that algorithm by adding *rejuvenation* steps and show that this leads to considerable improvements, thus strengthening the case for particle filters as another tool for Bayesian inference in computational linguistics.

The rest of the paper is structured as follows. Sections 2 and 3 provide the relevant background about word segmentation and previous work. Section 4 describes our algorithm. Section 5 reports on an experimental evaluation of our algorithm, and section 6 concludes and suggests possible directions for future research.

2 Model description

The *Unigram* model assumes that words in a sequence are generated independently whereas the *Bigram* model models dependencies between adjacent words. This has been shown by Goldwater (2006) to markedly improve segmentation performance. We perform experiments on both models but, for reasons of space, only give an overview of the Unigram model, referring the reader to the original papers for more detailed descriptions. (Goldwater, 2006; Goldwater et al., 2009)

A sequence of words or utterance is generated by making independent draws from a discrete distribution over words, G. As neither the actual "true" words nor their number is known in advance, G is modelled as a draw from a DP. A DP is parametrized by a base distribution P_0 and a concentration parameter α. Here, P_0 assigns a probability to every possible word, i.e. sequence of segments, and α controls the sparsity of G; the smaller α, the sparser G tends to be.

To computationally cope with the unbounded nature of draws from a DP, they can be "integrated out", yielding the Chinese Restaurant Process (CRP), an infinitely exchangeable conditional predictive distribution. The CRP also provides an intuitive generative story for the observed data. Each generated word token corresponds to a customer sit-

Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, pages 85–89,
Jeju, Republic of Korea, 8-14 July 2012.

ting at one of the unboundedly many tables in an imaginary Chinese restaurant. Customers choose their seats sequentially, and they sit either at an already occupied or a new table. The former happens with probability proportional to the number of customers already sitting at a table and corresponds to generating one more token of the word type all customers at a table instantiate. The latter happens with probability proportional to α and corresponds to generating a token by sampling from the base distribution, thus also determining the type for all potential future customers at the new table.

Given this generative process, word segmentation can be cast as a probabilistic inference problem. For a fixed input, in our case a sequence of phonemes, our goal is to determine the posterior distribution over segmentations. This is usually infeasible to do exactly, leading to the use of approximate inference methods.

3 Previous Work

The "standard" inference algorithms for the Unigram and Bigram model are MCMC samplers that are batch algorithms making multiple iterations over the data to non-deterministically explore the state space of possible segmentations. If an MCMC algorithm runs long enough, the probability of it visiting any specific segmentation is the probability of that segmentation under the target posterior distribution, here, the distribution over segmentations given the observed data.

The MCMC algorithm of Goldwater et al. (2009) is a *Gibbs* sampler that makes very small moves through the state space by changing individual word boundaries one at a time. An alternative MCMC algorithm that samples segmentations for entire utterances was proposed by Mochihashi et al. (2009). Below, we correct a minor error in the algorithm, recasting it as a Metropolis-within-Gibbs sampler.

Moving beyond MCMC algorithms, Pearl et al. (2010) describe an algorithm that can be seen as a degenerate limiting case of a particle filter with only one particle. Their *Dynamic Programming Sampling* algorithm makes a single pass through the data, processing one utterance at a time by sampling a segmentation given the choices made for all previous utterances. While their algorithm comes with no guarantee that it converges on the intended posterior distribution, Börschinger and Johnson (2011) showed how to construct a particle filter that is asymptotically correct, although experiments suggested that the number of particles required for good performance is impractically large.

This paper shows how their algorithm can be improved by adding *rejuvenation steps*, which we will describe in the next section.

4 A Particle Filter with Rejuvenation

The core idea of a *particle filter* is to sequentially approximate a target posterior distribution P by N weighted point samples or "particles". Each particle is updated one observation at a time, exploiting the insight that Bayes' Theorem can be applied recursively, as illustratively shown for the case of calculating the posterior probability of a hypothesis H given two observations O_1 and O_2:

$$P(H|O_1) \propto P(O_1|H)P(H) \quad (1)$$

$$P(H|O_1, O_2) \propto P(O_2|H)P(H|O_1) \quad (2)$$

If the observations are conditionally independent given the hypothesis, one can simply take the posterior at time step t as the prior for the posterior update at time step $t + 1$.

Here, each particle corresponds to a specific segmentation of the data observed so far, or more precisely, the specific CRP seating of word tokens in this segmentation; we refer to this as its *history*. Its weight indicates how well a particle is supported by the data, and each observation corresponds to an unsegmented utterance. With this, the basic particle filter algorithm can be described as follows: Begin with N "empty" particles. To get the particles at time $t+1$ from the particles at time t, update each particle using the observation at time $t+1$ as follows: sample a segmentation for this observation, given the particle's history, then add the words in this segmentation to that history. After each particle has been updated, their weights are adjusted to reflect how well they are now supported by the observations. The set of updated and reweighted particles constitutes the approximation of the posterior at time $t + 1$.

To overcome the problem of degeneracy (the situation where only very few particles have non-negligible weights), Börschinger and Johnson use

resampling; basically, high-probability particles are permitted to have multiple descendants that can replace low-probability particles. For reasons of space, we refer the reader to Börschinger and Johnson (2011) for the details of these steps.

While necessary to address the degeneracy problem, resampling leads to a *loss of sample diversity*; very quickly, almost all particles have an identical history, descending from only a small number of (previously) high probability particles. With a strict online learning constraint, this can only be counteracted by using an extremely large number of particles. An alternative strategy which we explore here is to use *rejuvenation*; the core idea is to restore sample diversity after each resampling step by performing MCMC resampling steps on each particle's history, thus leading to particles with different histories in each generation, even if they all have the same parent. (e.g., Canini et al. (2009)) This makes it necessary to store previously processed observations and thus no longer qualifies as online learning in a strict sense, but it still yields an incremental algorithm that learns as the observations arrive sequentially, instead of delaying learning until all observations are available.

In our setting, rejuvenation works as follows. After each resampling step, for each particle the algorithm performs a fixed number of the following rejuvenation steps:

1. randomly choose a previously observed utterance
2. resample the segmentation for this utterance and update the particle accordingly

For the resampling step, we use Mochihashi et al. (2009)'s algorithm to efficiently sample segmentations for an unsegmented utterance o, given a sequence of n previously observed words $W_{1:n}$. As the CRP is exchangeable, during resampling we can treat every utterance as if it were the last, making it possible to use this algorithm for any utterance, irrespective of its actual position in the data. Crucially, however, the distribution over segmentations that this algorithm samples from is not the true posterior distribution $P(\cdot|o, \alpha, W_{1:n})$ as defined by the CRP, but a slightly different *proposal* distribution $Q(\cdot|o, \alpha, W_{1:n})$ that does not take into account the intra-sentential word dependencies for a segmentation of o. It is precisely because we ignore these dependencies that an efficient dynamic programming algorithm is possible, but because Q is different from the target conditional distribution P, our algorithm that uses Q instead of P needs to correct for this. In a particle filter, this is done when the particle weights are calculated (Börschinger and Johnson, 2011). For an MCMC algorithm or our rejuvenation step, a *Metropolis-Hastings accept/reject* step is required, as described in detail by Johnson et al. (2007) in the context of grammatical inference.[1]

In our case, during rejuvenation an utterance u with current segmentation s is reanalyzed as follows:

- remove all the words contained in s from the particle's current state L, yielding state $L*$
- sample a proposal segmentation s' for u from $Q(\cdot|u, L*, \alpha)$, using Mochihashi et al. (2009)'s dynamic programming algorithm
- calculate $m = \min\{1, \frac{P(s'|L*,\alpha)Q(s|L*,\alpha)}{P(s|L*,\alpha)Q(s'|L*,\alpha)}\}$
- with probability m, accept the new sample and update $L*$ accordingly, else keep the original segmentation and set the particle's state back to L

This completes the description of our extension to the algorithm. The remainder of the paper empirically evaluates the particle filter with rejuvenation.

5 Experiments

We compare the performance of a batch Metropolis-Hastings sampler for the Unigram and Bigram model with that of particle filter learners both with and without rejuvenation, as described in the previous section. For the batch samplers, we use simulated annealing to facilitate the finding of high probability solutions, and for the particle filters, we compare the performance of a 'degenerate' 1-particle learner with a 16-particle learner in the rejuvenation setting.

To get an impression of the contribution of particle number and rejuvenation steps, we compare

[1]Because Mochihashi et al. (2009)'s algorithm samples directly from the proposal distribution without the accept-reject step, it is not actually sampling from the intended posterior distribution. Because Q approaches the true conditional distribution as the size of the training data increases, however, there may be almost no noticeable difference between using and not using the accept/reject step, though strictly speaking, it is required to guarantee convergence to the the target posterior.

	Unigram		Bigram	
	TF	logProb	TF	logProb
MHS	50.39	**-196.74**	70.93	**-237.24**
PF_1	55.82	-248.21	49.43	-265.40
PF_{16}	62.34	-239.22	50.14	-262.34
PF_{1000}	64.11	-234.87	57.88	-254.17
$PF_{1,100}$	63.17	-245.32	66.88	-257.65
$PF_{16,100}$	68.05	-235.71	70.05	-251.66
$PF_{1,1600}$	**77.06**	-228.79	**74.47**	-249.78

Table 1: Results for both the Unigram and the Bigram model. MHS is a Metropolis-Hastings batch sampler. PF_x is a particle filter with x particles and no rejuvenation. $PF_{x,s}$ is a particle filter with x particles and s rejuvenation steps. TF is token f-score, logProb is the log-probability ($\times 10^3$) of the training-data at the end of learning. Less negative logProb indicates a better solution according to the model, higher TF indicates a better quality segmentation. All results are averaged across 4 runs. Results for the 1000 particle setting are taken from Börschinger and Johnson (2011).

the 16-particle learner with rejuvenation with a 1-particle learner that performs 16 times as many rejuvenation samples. For comparison, we also cite previous results for the 1000-particle learners without rejuvenation reported in Börschinger and Johnson (2011), using their choice of parameters to allow for a direct comparison: $\alpha = 20$ for the Unigram model, $\alpha_0 = 3000$, $\alpha_1 = 100$ for the Bigram model, and we use their base-distribution which differs from the one described in Goldwater et al. (2009) in that it doesn't assume a uniform distribution over segments in the base-distribution but puts a Dirichlet Prior on it.

We apply each learner to the Bernstein-Ratner corpus (Brent, 1999) that is standardly used in the word segmentation literature, which consists of 9790 unsegmented and phonemically transcribed child-directed speech utterances. We evaluate each algorithm in two ways: inference performance, for which the final log-probability of the training data is the criterion, and segmentation performance, for which we consider token f-score to be the best measure, since it indicates how well the actual word tokens in the data are recovered.Note that these two measures can diverge, as previously documented for the Unigram model (Goldwater, 2006) and, less so, for the Bigram model (Pearl et al., 2010). Table 1 gives the results for our experiments.

For both models, adding rejuvenation always improves performance markedly as compared to the corresponding run without rejuvenation both in terms of log-probability and segmentation f-score. Note in particular that for the Bigram model, using 16 particles with 100 rejuvenation steps leads to an improvement in token f-score of more than 10% points over 1000 particles without rejuvenation.

Comparing the 1-particle learner with 1600 rejuvenation steps to the 16-particle learner with 100 rejuvenation steps, for both models the former outperforms the latter in both log-probability and token f-score. This suggests that if one has to trade-off particle number against rejuvenation steps, one may be better off favouring the latter.

Despite the dramatic improvement over not using rejuvenation, there is still a considerable gap between all the incremental learners and the batch sampling algorithm in terms of log-probability. A similar observation was made by Johnson and Goldwater (2009) for incremental initialisation in word segmentation using adaptor grammars. Their *batch* sampler converged on higher token f-score but lower probability solutions in some settings when initialized in an incremental fashion as opposed to randomly. We agree with their suggestion that this may be due to the "greedy" character of an incremental learner.

6 Conclusion and outlook

We have shown that adding rejuvenation to a particle filter improves segmentation scores and log-probabilities. Yet, our incremental algorithm still finds lower probability but high quality token f-scores compared to its batch counterpart. While in principle, increasing the number of rejuvenation steps and particles will make this gap smaller and smaller, we believe the existence of the gap to be interesting in its own right, suggesting a general difference in learning behaviour between batch and incremental learners, especially given the similar results in Johnson and Goldwater (2009). Further research into incremental learning algorithms may help us better understand how processing limitations can affect learning and why this may be beneficial for language acquisition, as suggested, for example, in Newport (1988).

References

Benjamin Börschinger 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.

Michael R. Brent. 1999. An efficient, probabilistically sound algorithm for segmentation and word discovery. *Machine Learning*, 34(1-3):71–105.

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)*, pages 65–72.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2009. A bayesian framework for word segmentation: Exploring the effects of context. *Cognition*, 112(1):21–54.

Sharon Goldwater. 2006. *Nonparametric Bayesian Models of Lexical Acquisition*. Ph.D. thesis, Brown University.

Mark Johnson and Sharon Goldwater. 2009. Improving nonparametric bayesian inference: Experiments on unsupervised word segmentation with adaptor grammars. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, Boulder, Colorado.

Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. 2007. Bayesian inference for pcfgs via markov chain monte carlo. In *Proceedings of Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics*.

Daichi Mochihashi, Takeshi Yamada, and Naonori Ueda. 2009. Bayesian unsupervised word segmentation with nested pitman-yor language modeling. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 100–108, Suntec, Singapore, August. Association for Computational Linguistics.

Elissa L Newport. 1988. Constraints on learning and their role in language acquisition: Studies of the acquisition of american sign language. *Language Sciences*, 10:147–172.

Lisa Pearl, Sharon Goldwater, and Mark Steyvers. 2010. Online learning mechanisms for bayesian models of word segmentation. *Research on Language and Computation*, 8(2):107–132.