Bayesian Unsupervised Word Segmentation with a Nested Pitman-Yor Language Modeling

Daichi Mochihashi Takeshi Yamada Naonori Ueda
NTT Communication Science Laboratories
Hikaridai 2-4, Keihanna Science City, Kyoto Japan
{daichi,yamada,ueda}@cslab.kecl.ntt.co.jp

Abstract

In this paper, we propose a new Bayesian model for fully unsupervised word segmentation and an efficient blocked Gibbs sampler combined with dynamic programming for inference. Our model is a nested hierarchical Pitman-Yor language model, where Pitman-Yor spelling model is embedded in the word model. We confirmed that our model significantly outperforms previous reported results in both phonetic transcripts and standard datasets for Chinese and Japanese word segmentation. Our model is also considered as a way to construct an accurate word n-gram language model directly from characters of arbitrary language without any "word" indications.

1 Introduction

"Word" is no trivial concept in many languages. Asian languages such as Japanese and Chinese have no explicit word boundaries, thus word segmentation is a crucial first step when processing them. Even in western languages, valid "words" are often not identical to space-separated tokens: proper nouns such as "United States" or idiomatic phrases such as "with respect to" actually function as a single word. In fact, we often condense them into the virtual words "US" and "w.r.t.".

In order to extract "words" from text streams, unsupervised word segmentation is an important research area because the criteria for creating supervised training data could be arbitrary, and will be suboptimal for applications that rely on segmentations. It is particularly difficult to create "correct" training data for speech transcripts, colloquial texts, and classics where segmentations are often ambiguous, let alone is impossible for unknown languages whose properties computational linguists would seek to uncover.

From a scientific point of view, it is also interesting because it can shed light on how children learn "words", without the explicitly given boundaries for every word assumed by supervised methods.

Recently, model-based methods have been proposed for unsupervised segmentation, in particular those based on Dirichlet processes on words (Goldwater et al., 2006; Xu et al., 2008). This maximizes the probability of segmentation $\mathbf{w}$ given a string s :

$$\hat{\mathbf{w}} = \operatorname*{argmax}_{\mathbf{w}} p(\mathbf{w}|s)\,. \tag{1}$$

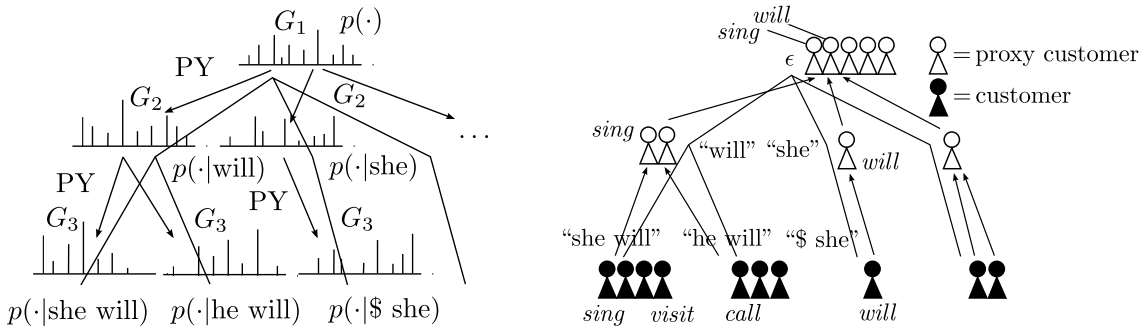
This approach often implicitly includes heuristic criteria proposed so far[1], while having a clear statistical semantics to find the most probable word segmentation that will maximize the probability of the data, here the strings.

However, they are still naïve with respect to word spellings, and the inference is very slow owing to inefficient Gibbs sampling. Crucially, since they rely on sampling a word boundary between two neighboring words, they can leverage only up to bigram word dependencies.

In this paper, we extend this work to propose a more efficient and accurate unsupervised word segmentation that will optimize the performance of the word n-gram Pitman-Yor (i.e. Bayesian Kneser-Ney) language model, with an accurate character ∞-gram Pitman-Yor spelling model embedded in word models. Furthermore, it can be viewed as a method for building a high-performance n-gram language model directly from character strings of arbitrary language. It is carefully smoothed and has no "unknown words" problem, resulting from its model structure.

This paper is organized as follows. In Section 2, we briefly describe a language model based on

[1] For example, the TANGO algorithm (Ando and Lee, 2003) essentially finds segments such that character n-gram probabilities are maximized blockwise, averaged over n.



(a) Generating n-gram distributions G hierarchically from the Pitman-Yor process. Here, $n = 3$.

(b) Equivalent representation using a hierarchical Chinese Restaurant process. Each word in a training text is a "customer" shown in italic, and added to the leaf of its two words context.

Figure 1: Hierarchical Pitman-Yor Language Model.

the Pitman-Yor process (Teh, 2006b), which is a generalization of Dirichlet process used in previous research. By embedding a character n-gram in word n-gram from a Bayesian perspective, Section 3 introduces a novel language model for word segmentation, which we call the Nested Pitman-Yor language model. Section 4 describes an efficient blocked Gibbs sampler that uses dynamic programming for inference. In Section 5 we describe experiments on the standard datasets in Chinese and Japanese, and semi-supervised experiments are also explored. Section 6 is a discussion and Section 7 concludes the paper.

2 Pitman-Yor process and n-gram models

To compute a probability $p(\mathbf{w}|s)$ in (1), we adopt the Bayesian language model lately proposed by (Teh, 2006b; Goldwater et al., 2005) based on a Pitman-Yor process, an extension of the Dirichlet process. As we shall see, this is a Bayesian theory of the best-performing Kneser-Ney smoothing of n-grams (Kneser and Ney, 1995), which allows an integrated modeling from a Bayesian perspective as persued in this paper.

The Pitman-Yor (PY) process is a stochastic process that generates discrete probability distribution G that is similar to another distribution G_0, called a *base measure*. It is written as

$$G \sim \mathrm{PY}(G_0, d, \theta)\,, \tag{2}$$

where d is a discount factor and θ controls how similar G is to G_0 on average.

Suppose we have a unigram word distribution $G_1 = \{\, p(\cdot)\,\}$ where $\cdot$ ranges over each word in the lexicon. The bigram distribution $G_2 = \{\, p(\cdot|v)\,\}$ given a word v is different from G_1, but will be similar to G_1 especially for high frequency words. Therefore, we can generate G_2 from a PY process of base measure G_1 as $G_2 \sim \mathrm{PY}(G_1, d, \theta)$. Similarly, trigram distribution $G_3 = \{\, p(\cdot|v'v)\,\}$ given an additional word v' is generated as $G_3 \sim \mathrm{PY}(G_2, d, \theta)$, and G_1, G_2, G_3 will form the tree structure shown in Figure 1(a).

Actually, we cannot observe G directly because it will be infinite dimensional distribution over possible vocabularies, as we shall see in this paper. However, if we integrate out G it is known that Figure 1(a) can be represented by an equivalent hierarchical Chinese Restaurant Process (CRP) (Aldous, 1985) as in Figure 1(b).

In the CRP representation, each word in the training data is called a "customer" who is added to the leaf in the learning phase. Imagine we have a sentence "she will sing" in the training data used to learn a trigram model. We add each word as a customer given two preceding words as the context: "she" given "$ $", "will" given "$ she", "sing" given "she will", $\cdots$, where "$" is a special token of a sentence boundary in language modeling (Brown et al., 1992).

Each customer w is added to the leaf node associated with its context h (see Figure 1(b)): this means an increment in the corresponding n-gram counts $c(w|h)$. However, in a back-off sense it might have actually been generated from bigrams. When this is the case, we must add a *proxy customer* to the parent node of its $(n-1)$-gram context h', and this process might recurse.

As a result, denoting the number of times $c(w|h)$ was generated from the parent and a proxy customer was sent as t_{hw}, the n-gram probability of this hierarchical Pitman-Yor language model

(HPYLM) can be hierarchically computed as

$$p(w|h) = \frac{c(w|h) - d \cdot t_{hw}}{\theta + c(h)} + \frac{\theta + d \cdot t_{h\cdot}}{\theta + c(h)} p(w|h'), \quad (3)$$

where $p(w|h')$ is the same probability using a $(n-1)$-gram context h'. When we set $t_{hw} \equiv 1$, (3) recovers a Kneser-Ney smoothing: thus a HPYLM is a Bayesian Kneser-Ney language model as well as an extension of the hierarchical Dirichlet Process (HDP) used in (Goldwater et al., 2006). θ, d are hyperparameters that can be learned as Gamma and Beta posteriors, respectively, given the data. For details, see (Teh, 2006a).

The inference of this model interleaves adding and removing a customer to optimize t_{hw}, d, and θ using MCMC. However, in our case "words" are not known a priori: the next section describes how to accomplish this by constructing a nested HPYLM of words and characters, with the associated inference algorithm.

3 Nested Pitman-Yor Language Model

Thus far we have assumed that the unigram G_1 is already given, but of course it should also be generated as $G_1 \sim \mathrm{PY}(G_0, d, \theta)$.

Here, a problem occurs: What can we use for G_0, namely the prior probabilities over words[2]? If a lexicon is finite, we can use a uniform prior $G_0(w) = 1/|V|$ for every word w in lexicon V. However, with word segmentation every substring could be a word, thus the lexicon is not limited but will be countably infinite.

Building an accurate G_0 is crucial for word segmentation, because it determines how the possible words will look like. Previous work using a Dirichlet process used a relatively simple prior for G_0, namely an uniform distribution over characters (Goldwater et al., 2006), or a prior solely dependent on word length with a Poisson distribution whose parameter is fixed by hand (Xu et al., 2008).

In contrast, in this paper we use a simple but more elaborate model, that is, a character n-gram language model that also employs HPYLM. This is important because in English, for example, words are likely to end in '–tion' and begin with 're–', but almost never end in '–tio' nor begin with 'sre–' [3].

[2]Note that this is different from unigrams, which are posterior distribution given data.

[3]Imagine we want to segment an English character string "itisrecognizedasthe$\cdots$."

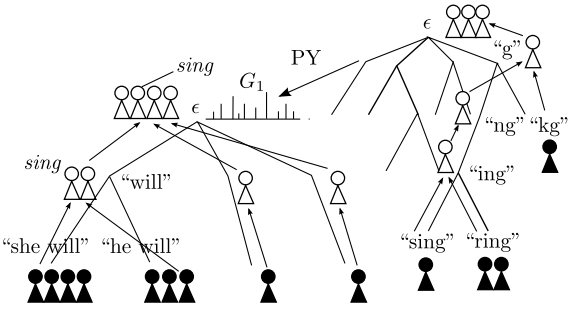


Figure 2: Chinese restaurant representation of our Nested Pitman-Yor Language Model (NPYLM).

Therefore, we use

$$G_0(w) = p(\underset{k}{c_1 \cdots c_k}) \quad (4)$$

$$= \prod_{i=1} p(c_i | c_1 \cdots c_{i-1}) \quad (5)$$

where string $c_1 \cdots c_k$ is a spelling of w, and $p(c_i|c_1 \cdots c_{i-1})$ is given by the character HPYLM according to (3).

This language model, which we call Nested Pitman-Yor Language Model (NPYLM) hereafter, is the hierarchical language model shown in Figure 2, where the character HPYLM is embedded as a base measure of the word HPYLM.[4] As the final base measure for the character HPYLM, we used a uniform prior over the possible characters of a given language. To avoid dependency on n-gram order n, we actually used the ∞-gram language model (Mochihashi and Sumita, 2007), a variable order HPYLM, for characters. However, for generality we hereafter state that we used the HPYLM. The theory remains the same for ∞-grams, except sampling or marginalizing over n as needed.

Furthermore, we corrected (4) so that word length will have a Poisson distribution whose parameter can now be estimated for a given language and word type. We describe this in detail in Section 4.3.

Chinese Restaurant Representation

In our NPYLM, the word model and the character model are not separate but connected through a nested CRP. When a word w is generated from its parent at the unigram node, it means that w is drawn from the base measure, namely a character HPYLM. Then we divide w into characters $c_1 \cdots c_k$ to yield a "sentence" of characters and feed this into the character HPYLM as data.

[4]Strictly speaking, this is not "nested" in the sense of a Nested Dirichlet process (Rodriguez et al., 2008). However, this name represents our architecture very well.

Conversely, when a table becomes empty, this means that the data associated with the table are no longer valid. Therefore we remove the corresponding customers from the character HPYLM using the inverse procedure of adding a customer in Section 2.

All these processes will be invoked when a string is segmented into "words" and customers are added to the leaves of the word HPYLM. To segment a string into "words", we used efficient dynamic programming combined with MCMC, as described in the next section.

4 Inference

To find the hidden word segmentation $\mathbf{w}$ of a string $s = c_1 \cdots c_N$, which is equivalent to the vector of binary hidden variables $\mathbf{z} = z_1 \cdots z_N$, the simplest approach is to build a Gibbs sampler that randomly selects a character c_i and draw a binary decision z_i as to whether there is a word boundary, and then update the language model according to the new segmentation (Goldwater et al., 2006; Xu et al., 2008). When we iterate this procedure sufficiently long, it becomes a sample from the true distribution (1) (Gilks et al., 1996).

However, this sampler is too inefficient since time series data such as word segmentation have a very high correlation between neighboring words. As a result, the sampler is extremely slow to converge. In fact, (Goldwater et al., 2006) reports that the sampler would not mix without annealing, and the experiments needed 20,000 times of sampling for every character in the training data.

Furthermore, it has an inherent limitation that it cannot deal with larger than bigrams, because it uses only *local* statistics between directly contiguous words for word segmentation.

4.1 Blocked Gibbs sampler

Instead, we propose a sentence-wise Gibbs sampler of word segmentation using efficient dynamic programming, as shown in Figure 3.

In this algorithm, first we randomly select a string, and then remove the "sentence" data of its word segmentation from the NPYLM. Sampling a new segmentation, we update NPYLM by adding a new "sentence" according to the new segmentation. If we repeat this process, it is expected to mix rapidly because it implicitly considers all possible segmentations of the given string at the same time.

This is called a *blocked* Gibbs sampler that samples $\mathbf{z}$ block-wise for each sentence. It has an additional advantage in that we can accommodate higher-order relationships than bigrams, particularly trigrams, for word segmentation. [5]

1: **for** $j = 1 \cdots J$ **do**
2: **for** s in randperm $(s_1, \cdots, s_D)$ **do**
3: **if** $j > 1$ **then**
4: Remove customers of $\mathbf{w}(s)$ from Θ
5: **end if**
6: Draw $\mathbf{w}(s)$ according to $p(\mathbf{w}|s, \Theta)$
7: Add customers of $\mathbf{w}(s)$ to Θ
8: **end for**
9: Sample hyperparameters of Θ
10: **end for**

Figure 3: Blocked Gibbs Sampler of NPYLM Θ.

4.2 Forward-Backward inference

Then, how can we sample a segmentation $\mathbf{w}$ for each string s? In accordance with the Forward filtering Backward sampling of HMM (Scott, 2002), this is achieved by essentially the same algorithm employed to sample a PCFG parse tree within MCMC (Johnson et al., 2007).

Forward Filtering. For this purpose, we maintain a forward variable $\alpha[t][k]$ in the bigram case. $\alpha[t][k]$ is the probability of string $c_1 \cdots c_t$ with the final k characters being a word (see Figure 4). Segmentations before the final k characters are marginalized using the following recursive relationship:

$$\alpha[t][k] = \sum_{j=1}^{t-k} p(c_{t-k+1}^{t} | c_{t-k-j+1}^{t-k}) \cdot \alpha[t-k][j] \quad (6)$$

where $\alpha[0][0] = 1$ and we wrote $c_n \cdots c_m$ as c_n^m. [6] The rationale for (6) is as follows. Since maintaining binary variables $z_1, \cdots, z_N$ is equivalent to maintaining a distance to the nearest backward word boundary for each t as q_t, we can write

$$\alpha[t][k] = p(c_1^t, q_t = k) \quad (7)$$

$$= \sum_j p(c_1^t, q_t = k, q_{t-k} = j) \quad (8)$$

$$= \sum_j p(c_1^{t-k}, c_{t-k+1}^t, q_t = k, q_{t-k} = j) \quad (9)$$

[5] In principle fourgrams or beyond are also possible, but will be too complex while the gain will be small. For this purpose, Particle MCMC (Doucet et al., 2009) is promising but less efficient in a preliminary experiment.

[6] As Murphy (2002) noted, in semi-HMM we cannot use a standard trick to avoid underflow by normalizing $\alpha[t][k]$ into $p(k|t)$, since the model is asynchronous. Instead we always compute (6) using `logsumexp()`.

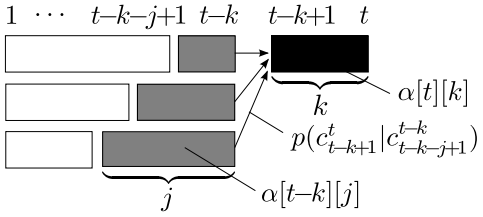


Figure 4: Forward filtering of $\alpha[t][k]$ to marginalize out possible segmentations j before $t-k$.

for $t = 1$ to N **do**
  for $k = \max(1, t-L)$ to t **do**
    Compute $\alpha[t][k]$ according to (6).
  end for
end for
Initialize $t \leftarrow N$, $i \leftarrow 0$, $w_0 \leftarrow \$$
while $t > 0$ **do**
  Draw $k \propto p(w_i | c_{t-k+1}^{t}, \Theta) \cdot \alpha[t][k]$
  Set $w_i \leftarrow c_{t-k+1}^{t}$
  Set $t \leftarrow t-k$, $i \leftarrow i+1$
end while
Return $\mathbf{w} = w_i, w_{i-1}, \cdots, w_1$.

Figure 5: Forward-Backward sampling of word segmentation $\mathbf{w}$. (in bigram case)

$$= \sum_j p(c_{t-k+1}^{t} | c_{t-k-j+1}^{t-k}) p(c_1^{t-k}, q_{t-k} = j) \quad (10)$$

$$= \sum_j p(c_{t-k+1}^{t} | c_{t-k-j+1}^{t-k}) \alpha[t-k][j] \,, \quad (11)$$

where we used prior independency between q_t and q_{t-k} and uniform prior over q_t in (10) above.

Backward Sampling. Once the probability table $\alpha[t][k]$ is obtained, we can sample a word segmentation backwards. Since $\alpha[N][k]$ is a marginal probability of string c_1^N with the last k characters being a word, and there is always a sentence boundary token \$ at the end of the string, with probability proportional to $p(\$ | c_{N-k}^{N}) \cdot \alpha[N][k]$ we can sample k to choose the boundary of the final word. The second final word is similarly sampled using the probability of preceding the last word just sampled: we continue this process until we arrive at the beginning of the string (Figure 5).

Trigram case. For simplicity, we showed the algorithm for bigrams above. For trigrams, we maintain a forward variable $\alpha[t][k][j]$, which represents a marginal probability of string $c_1 \cdots c_t$ with both the final k characters and further j characters preceding it being words. Forward-Backward algorithm becomes complicated thus omitted, but can be derived following the extended algorithm for second order HMM (He, 1988).

Complexity This algorithm has a complexity of $O(NL^2)$ for bigrams and $O(NL^3)$ for trigrams for each sentence, where N is the length of the sentence and L is the maximum allowed length of a word ($\leq N$).

4.3 Poisson correction

As Nagata (1996) noted, when only (4) is used inadequately low probabilities are assigned to long words, because it has a largely exponential distribution over length. To correct this, we assume that word length k has a Poisson distribution with a mean λ:

$$\mathrm{Po}(k|\lambda) = e^{-\lambda} \frac{\lambda^k}{k!} \,. \quad (12)$$

Since the appearance of $c_1 \cdots c_k$ is equivalent to that of length k and the content, by making the character model explicit as Θ we can write

$$p(c_1 \cdots c_k | \Theta) = p(c_1 \cdots c_k, k | \Theta) \quad (13)$$

$$= \frac{p(c_1 \cdots c_k, k | \Theta)}{p(k|\Theta)} \mathrm{Po}(k|\lambda) \quad (14)$$

where $p(c_1 \cdots c_k, k | \Theta)$ is a character n-gram probability given by (5), and $p(k|\Theta)$ is a prior probability that a word of length k will be generated from Θ. While previous work used $p(k|\Theta) = (1 - p(\$))^{k-1} p(\$)$, this is only true for unigrams. Instead, we employed a Monte Carlo method that generates words randomly from Θ to obtain the empirical estimates of $p(k|\Theta)$.

Estimating λ. Of course, we do not leave λ as a constant. Instead, we put a Gamma distribution

$$p(\lambda) = \mathrm{Ga}(a, b) = \frac{b^a}{\Gamma(a)} \lambda^{a-1} e^{-b\lambda} \quad (15)$$

to estimate λ from the data for given language and word type.[7] Here, $\Gamma(x)$ is a Gamma function and a, b are the hyperparameters chosen to give a nearly uniform prior distribution.

Denoting W as a set of "words" obtained from word segmentation, the posterior distribution $p(\lambda|W)$ given W is

$$p(\lambda|W) \propto p(W|\lambda) p(\lambda)$$

$$= \mathrm{Ga}\left(a + \sum_{w \in W} t(w)|w|,\ b + \sum_{w \in W} t(w)\right), \quad (16)$$

where $t(w)$ is the number of times word w is generated from the character HPYLM, i.e. the number of tables $t_{\epsilon w}$ for w in word unigrams. We sampled λ from this posterior for each Gibbs iteration.

[7] We used different λ for different word types, such as digits, alphabets, hiragana, CJK characters, and their mixtures. W is a set of words of each such type in this case.

Model	P	R	F	LP	LR	LF
NPY(3)	74.8	75.2	75.0	47.8	**59.7**	53.1
NPY(2)	74.8	**76.7**	**75.7**	57.3	56.6	57.0
HDP(2)	**75.2**	69.6	72.3	**63.5**	55.2	**59.1**

Table 1: Segmentation accuracies on English phonetic transcripts. Results for HDP(2) are taken from Goldwater et al. (2009), which corrects the errors in Goldwater et al. (2006).

Model	time	iterations
NPYLM	17min	200
HDP	10h 55min	20000

Table 2: Computations needed for Table 1. Iterations for "HDP" is the same as described in (Goldwater et al., 2009).

5 Experiments

To validate our model, we conducted experiments on standard datasets for Japanese and Chinese word segmentation that are publicly available, as well as the same dataset used in (Goldwater et al., 2006). Note that NPYLM maximizes the probability of strings, equivalently, minimizes the perplexity per character. Therefore, the recovery of the "ground truth" that is not available for inference is a kind of side-effect in unsupervised learning.

Since our implementation is based on Unicode and learns all hyperparameters from the data, we also confirmed that NPYLM segments the Arabic Gigawords equally well.

5.1 English phonetic transcripts

In order to directly compare with the previously reported result, we first used the same dataset as (Goldwater et al., 2006). This dataset consists of 9,790 English phonetic transcripts from CHILDES data (MacWhinney and Snow, 1985).

Since our algorithm converges rather fast, we ran the Gibbs sampler of trigram NPYLM for 200 iterations to obtain the results in Table 1. All of the precision (P), recall (R), F-measure (F) significantly outperformed the previous result based on HDP: while the same measures over the obtained lexicon (LP, LR, LF) are not always improved. Moreover, the average length of words inferred was surprisingly similar to ground truth: 2.88, while the ground truth is 2.87.

Table 2 shows the empirical computational time needed to obtain these results. Although the convergence in MCMC is not uniquely identified, improvement in efficiency is also outstanding.

Model	MSR	CITYU	Kyoto
NPY(2)	0.804 (52.0)	**0.820** (**126.7**)	0.614 (23.5)
NPY(3)	**0.806** (**49.2**)	0.818 (128.0)	**0.653** (**20.8**)
ZK08	0.667 (—)	0.692 (—)	—

Table 3: Accuracy and perplexity per character by different models. NPY(2) and NPY(3) means bigram and trigram NPYLM, respectively. ZK08 is the best results reported in (Zhao and Kit, 2008). We used ∞-gram for characters.

	MSR	CITYU	Kyoto
Semi	0.895 (**48.8**)	0.898 (**124.7**)	0.913 (**20.3**)
Sup	**0.945** (81.4)	**0.941** (194.8)	**0.971** (21.3)

Table 4: Semi-supervised and supervised results. Semi-supervised results used only 10K sentences (1/5) of supervised segmentations.

5.2 Chinese and Japanese word segmentation

To show applicability beyond small phonetic transcripts, we used standard datasets for Chinese and Japanese word segmentation, with all supervised segmentations removed in advance.

Chinese For Chinese, we used a publicly available SIGHAN Bakeoff 2005 dataset (Emerson, 2005). To compare with the latest unsupervised results (using a closed dataset of Bakeoff 2006), we chose the common sets prepared by Microsoft Research Asia (MSR) for simplified Chinese, and by City University of Hong Kong (CITYU) for traditional Chinese. We used a random subset of 50,000 sentences from each dataset for training, and the evaluation was conducted on the enclosed test data. [8]

Japanese For Japanese, we used the Kyoto Corpus (Kyoto) (Kurohashi and Nagao, 1998): we used random subset of 1,000 sentences for evaluation and the remaining 37,400 sentences for training. In all cases we removed all whitespaces to yield raw character strings for inference, and set $L = 4$ for Chinese and $L = 8$ for Japanese to run the Gibbs sampler for 200 iterations.

The results are shown in Table 3. Our NPYLM significantly ourperforms the best results using a heuristic approach reported in (Zhao and Kit, 2008). While bigram and trigram performances are similar for Chinese, trigram performs better for

[8] Notice that analyzing a test data is not easy for character-wise Gibbs sampler of previous work. Meanwhile, NPYLM easily finds the best segmentation using the Viterbi algorithm once the model is learned.

いづれ の 御 時 に か 、 女御 更衣 あまた さぶらひ たまひ ける 中に 、 いと やむごとなき 際 に は あ らぬ が 、 すぐれて 時めき たまふ あり けり 。 はじ めより 我 は と 思ひ あがり たまへ る 御 方々 、 め ざまし き もの に おとしめ そねみ たまふ 。 同じ ほ ど 、 それ より 下臈 の 更衣 たち は 、 まして やす から ず 。 朝夕 の 宮仕 につけても 、 ...

Figure 6: Unsupervised segmentation result for "*The Tale of Genji*". (16,443 sentences, 899,668 characters in total)

Japanese. In fact, although the difference in perplexity per character is not so large, the perplexity per word is radically reduced: 439.8 (bigram) to 190.1 (trigram). This is because trigram models can leverage complex dependencies over words to yield shorter words, resulting in better predictions and increased tokens.

Furthermore, NPYLM is easily amenable to semi-supervised or even supervised learning. In that case, we have only to replace the word segmentation $\mathbf{w}(s)$ in Figure 3 to the supervised one, for all or part of the training data. Table 4 shows the results using 10,000 sentences (1/5) or complete supervision. Our completely generative model achieves the performance of 94% (Chinese) or even 97% (Japanese) in supervised case. The result also shows that the supervised segmentations are suboptimal with respect to the perplexity per character, and even worse than unsupervised results. In semi-supervised case, using only 10K reference segmentations gives a performance of around 90% accuracy and the lowest perplexity, thanks to a combination with unsupervised data in a principled fashion.

5.3 Classics and English text

Our model is particularly effective for spoken transcripts, colloquial texts, classics, or unknown languages where supervised segmentation data is difficult or even impossible to create. For example, we are pleased to say that we can now analyze (and build a language model on) "*The Tale of Genji*", the core of Japanese classics written 1,000 years ago (Figure 6). The inferred segmentations are mostly correct, with some inflectional suffixes being recognized as words, which is also the case with English.

Finally, we note that our model is also effective for Western languages: Figure 7 shows a training text of "*Alice in Wonderland*" with all whitespaces removed, and the segmentation result.

While the data is extremely small (only 1,431 lines, 115,961 characters), our trigram NPYLM

lastly,shepicturedtoherselfhowthissamelittlesisterofhersw ould,intheafter-time,beherselfagrownwoman;andhowshe wouldkeep,throughallherriperyears,thesimpleandlovingh eartofherchildhood:andhowshewouldgatheraboutherothe rlittlechildren,andmaketheireyesbrightandeagerwithmany astrangetale,perhapsevenwiththedreamofwonderlandoflo ngago:andhowshewouldfeelwithalltheirsimplesorrows,an dfindapleasureinalltheirsimplejoys,rememberingherownc hild-life,andthehappysummerdays.

(a) Training data (in part).

last ly , she pictured to herself how this same little sister of her s would , inthe after - time , be herself agrown woman ; and how she would keep , through allher ripery ears , the simple and loving heart of her child hood : and how she would gather about her other little children ,and make theireyes bright and eager with many a strange tale , perhaps even with the dream of wonderland of longago : and how she would feel with all their simple sorrow s , and find a pleasure in all their simple joys , remember ing her own child - life , and thehappy summerday s .

(b) Segmentation result. Note we used no dictionary.

Figure 7: Word segmentation of "*Alice in Wonderland*".

can infer the words surprisingly well. This is because our model contains both word and character models that are combined and carefully smoothed, from a Bayesian perspective.

6 Discussion

In retrospect, our NPYLM is essentially a hierarchical Markov model where the units (=words) evolve as Markov processes, and each unit has subunits (=characters) that also evolve as Markov processes. Therefore, for such languages as English that have already space-separated tokens, we can also begin with tokens besides the character-based approach in Section 5.3. In this case, each token is a "character" whose code is the integer token type, and a sentence is a sequence of "characters." Figure 8 shows a part of the result computed over 100K sentences from Penn Treebank. We can see that some frequent phrases are identified as "words", using a fully unsupervised approach. Notice that this is only attainable with NPYLM where each phrase is described as a n-gram model on its own, here a word ∞-gram language model.

While we developed an efficient forward-backward algorithm for unsupervised segmentation, it is reminiscent of CRF in the discriminative approach. Therefore, it is also interesting to combine them in a discriminative way as persued in POS tagging using CRF+HMM (Suzuki et al., 2007), let alone a simple semi-supervised approach in Section 5.2. This paper provides a foundation of such possibilities.

nevertheless ,
he was admired
by many of his immediate subordinates
for his long work hours
and dedication to building northwest
into what he called a " mega carrier
. "

---

although
preliminary findings
were reported
more than a year ago ,
the latest results
appear
in today 's
new england journal of medicine ,
a forum
likely to bring new attention to the problem
.

---

south korea
registered a trade deficit of $ 101 million
in october
, reflecting the country 's economic sluggishness
, according to government figures released wednesday
.

Figure 8: Generative phrase segmentation of Penn Treebank text computed by NPYLM. Each line is a "word" consisting of actual words.

7 Conclusion

In this paper, we proposed a much more efficient and accurate model for fully unsupervised word segmentation. With a combination of dynamic programming and an accurate spelling model from a Bayesian perspective, our model significantly outperforms the previous reported results, and the inference is very efficient.

This model is also considered as a way to build a Bayesian Kneser-Ney smoothed word n-gram language model directly from characters with no "word" indications. In fact, it achieves lower perplexity per character than that based on supervised segmentations. We believe this will be particularly beneficial to build a language model on such texts as speech transcripts, colloquial texts or unknown languages, where word boundaries are hard or even impossible to identify a priori.

Acknowledgments

We thank Vikash Mansinghka (MIT) for a motivating discussion leading to this research, and Satoru Takabayashi (Google) for valuable technical advice.

References

David Aldous, 1985. *Exchangeability and Related Topics*, pages 1–198. Springer Lecture Notes in Math. 1117.

Rie Kubota Ando and Lillian Lee. 2003. Mostly-Unsupervised Statistical Segmentation of Japanese Kanji Sequences. *Natural Language Engineering*, 9(2):127–149.

Peter F. Brown, Vincent J. Della Pietra, Robert L. Mercer, Stephen A. Della Pietra, and Jennifer C. Lai. 1992. An Estimate of an Upper Bound for the Entropy of English. *Computational Linguistics*, 18:31–40.

Arnaud Doucet, Christophe Andrieu, and Roman Holenstein. 2009. Particle Markov Chain Monte Carlo. *in submission*.

Tom Emerson. 2005. SIGHAN Bakeoff 2005. http://www.sighan.org/bakeoff2005/.

W. R. Gilks, S. Richardson, and D. J. Spiegelhalter. 1996. *Markov Chain Monte Carlo in Practice*. Chapman & Hall / CRC.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2005. Interpolating Between Types and Tokens by Estimating Power-Law Generators. In *NIPS 2005*.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2006. Contextual Dependencies in Unsupervised Word Segmentation. In *Proceedings of ACL/COLING 2006*, pages 673–680.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2009. A Bayesian framework for word segmentation: Exploring the effects of context. *Cognition*, *in press*.

Yang He. 1988. Extended Viterbi algorithm for second order hidden Markov process. In *Proceedings of ICPR 1988*, pages 718–720.

Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. 2007. Bayesian Inference for PCFGs via Markov Chain Monte Carlo. In *Proceedings of HLT/NAACL 2007*, pages 139–146.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *Proceedings of ICASSP*, volume 1, pages 181–184.

Sadao Kurohashi and Makoto Nagao. 1998. Building a Japanese Parsed Corpus while Improving the Parsing System. In *Proceedings of LREC 1998*, pages 719–724. http://nlp.kuee.kyoto-u.ac.jp/nl-resource/corpus.html.

Brian MacWhinney and Catherine Snow. 1985. The Child Language Data Exchange System. *Journal of Child Language*, 12:271–296.

Daichi Mochihashi and Eiichiro Sumita. 2007. The Infinite Markov Model. In *NIPS 2007*.

Kevin Murphy. 2002. Hidden semi-Markov models (segment models). http://www.cs.ubc.ca/˜murphyk/Papers/segment.pdf.

Masaaki Nagata. 1996. Automatic Extraction of New Words from Japanese Texts using Generalized Forward-Backward Search. In *Proceedings of EMNLP 1996*, pages 48–59.

Abel Rodriguez, David Dunson, and Alan Gelfand. 2008. The Nested Dirichlet Process. *Journal of the American Statistical Association*, 103:1131–1154.

Steven L. Scott. 2002. Bayesian Methods for Hidden Markov Models. *Journal of the American Statistical Association*, 97:337–351.

Jun Suzuki, Akinori Fujino, and Hideki Isozaki. 2007. Semi-Supervised Structured Output Learning Based on a Hybrid Generative and Discriminative Approach. In *Proceedings of EMNLP-CoNLL 2007*, pages 791–800.

Yee Whye Teh. 2006a. A Bayesian Interpretation of Interpolated Kneser-Ney. Technical Report TRA2/06, School of Computing, NUS.

Yee Whye Teh. 2006b. A Hierarchical Bayesian Language Model based on Pitman-Yor Processes. In *Proceedings of ACL/COLING 2006*, pages 985–992.

Jia Xu, Jianfeng Gao, Kristina Toutanova, and Hermann Ney. 2008. Bayesian Semi-Supervised Chinese Word Segmentation for Statistical Machine Translation. In *Proceedings of COLING 2008*, pages 1017–1024.

Hai Zhao and Chunyu Kit. 2008. An Empirical Comparison of Goodness Measures for Unsupervised Chinese Word Segmentation with a Unified Framework. In *Proceedings of IJCNLP 2008*.