Nonparametric Bayesian Models for Spoken Language Understanding

Kei Wakabayashi
Tsukuba University, 1-2 Kasuga,
Tsukuba, Ibaraki 305-8550, Japan
kwakaba@slis.tsukuba.ac.jp

Johane Takeuchi, Kotaro Funakoshi
and **Mikio Nakano**
Honda Research Institute Japan Co., Ltd.
8-1 Honcho, Wako, Saitama 351-0188, Japan
{johane.takeuchi,funakoshi,nakano}
@jp.honda-ri.com

Abstract

In this paper, we propose a new generative approach for semantic slot filling task in spoken language understanding using a nonparametric Bayesian formalism. Slot filling is typically formulated as a sequential labeling problem, which does not directly deal with the posterior distribution of possible slot values. We present a nonparametric Bayesian model involving the generation of arbitrary natural language phrases, which allows an explicit calculation of the distribution over an infinite set of slot values. We demonstrate that this approach significantly improves slot estimation accuracy compared to the existing sequential labeling algorithm.

1 Introduction

Spoken language understanding (SLU) refers to the challenge of recognizing a speaker's intent from a natural language utterance, which is typically defined as a *slot filling task*. For example, in the utterance "Remind me to call John at 9am tomorrow", the specified information {**"time": *"9am tomorrow"***} and {**"subject": *"to call John"***} should be extracted. The term *slot* refers to a variable such as the **time** or **subject** that is expected to be filled with a value provided through the user's utterance.

The slot filling task is typically formulated as a sequential labeling problem as shown in Figure 1. This labeling scheme naturally represents the recognition of arbitrary phrases that appear in the transcription of an utterance. Formally speaking, when we assume a given set of slots $\{s_1, ..., s_M\}$ and denote the corresponding slot values by $\{v_{s_1}, ..., v_{s_M}\}$ where $v_{s_i} \in V_{s_i}$, the domain of each slot value V_{s_i} is an infinite set of word sequences. In this paper, we use the term *arbitrary slot filling task* to refer to this implicit problem statement, which inherently underlies the sequential labeling formulation.

In contrast, a different line of work has explored the case where V_{s_i} is provided as a finite set of possible values that can be handled by a backend system (Henderson, 2015). We refer to this type of task as a *categorical slot filling task*. In this case, the slot filling task is regarded as a classification problem that explicitly considers a value-based prediction, as shown in Figure 2. From this point of view, we can say that a distribution of slot values is actually concentrated in a small set of typical phrases, even in the arbitrary slot filling task, because users basically know what kind of function is offered by the system.

To reflect this observation, in this paper we explore the value-based formulation approach for arbitrary slot filling tasks. Unlike the sequential labeling formulation, which is basically position-based label prediction, our method directly estimates the posterior distribution over an infinite set of possible values for each slot V_{s_i}. The distribution is represented by using a Dirichlet process (Gershman and Blei, 2012), which is a nonparametric Bayesian formalism that generates a categorical distribution for any space. We demonstrate that this approach improves estimation accuracy in the arbitrary slot filling task compared with conventional sequential labeling approach.

The rest of this paper is organized as follows. In Section 2, we review the existing approaches for categorical and arbitrary slot filling tasks and intro-

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2144–2152,
Austin, Texas, November 1-5, 2016.

O	O	O	O	B-type	O	O	B-area	I-area	O
i'm	looking	for	a	restaurant	in	the	fen	ditton	area

Figure 1: sequential labeling formulation for slot filling tasks.

u = i'm looking for a restaurant in the fen ditton area

p(type)	p(area)	p(food)
$P(restaurant \mid u) = 0.96$	$P(fen\_ditton \mid u) = 0.98$	$P(italian \mid u) = 0.005$
$P(pub \mid u) = 0.03$	$P(girton \mid u) = 0.005$	$P(mexican \mid u) = 0.01$
$P(None \mid u) = 0.01$	$P(None \mid u) = 0.01$	$P(None \mid u) = 0.85$
...	...	...

Figure 2: Value-based formulation. The posterior probabilities of values for each slot are explicitly computed.

duce related work. In Section 3, we present our nonparametric Bayesian formulation, the hierarchical Dirichlet process slot model (HDPSM), which directly models an infinite set of slot values. On the basis of the HDPSM, we develop a generative utterance model that allows us to compute the posterior probability of slot values in Section 4. In Section 5, we introduce a two-stage slot filling algorithm that consists of a candidate generation step and a candidate ranking step using the proposed model. In Section 6, we show the experimental results for multiple datasets in different domains to demonstrate that the proposed algorithm performs better than the baseline sequential labeling method. We conclude in Section 7 with a brief summary.

2 Related Work

The difference between the categorical and arbitrary slot filling approaches has not been explicitly discussed in a comparative manner to date. In this section, we review existing work for both approaches.

For the categorical slot filling approach, various algorithms that directly model the distribution of slot values have been proposed, including generative models (Williams, 2010), maximum entropy linear classifiers (Metallinou et al., 2013), and neural networks (Ren et al., 2014). However, none of these models are applicable for predicting a variable that ranges over an infinite set, and it is not straightforward to extend them suitably. In particular, a discriminative approach is not applicable for arbitrary slot filling tasks because it requires a fixed finite set of slot values to take statistics.

The arbitrary slot filling approach is a natural application of shallow semantic parsing (Gildea, 2002), which is naturally formulated as a sequential labeling problem. Various sequential labeling algorithms have been applied to this task, including support vector machines, conditional random fields (CRF) (Lafferty et al., 2001; Hahn et al., 2011), and deep neural networks (Mesnil et al., 2015; Xu and Sarikaya, 2013). Vukotic et al. (2015) reported that the CRF is still the most accurate, rapid, and stable method among them. Because the focus of this paper is arbitrary slot filling tasks, we use CRFs as our baseline method.

In this paper, we apply nonparametric Bayesian models (Gershman and Blei, 2012) to represent the distribution over arbitrary phrases for each slot. The effectiveness of this phrase modeling approach has been examined in various applications including morphological analysis (Goldwater et al., 2011) and infinite vocabulary topic models (Zhai and Boyd-graber, 2013). Our method can be regarded as an application of this idea, although it is not straightforward to integrate it with the utterance generation process, as we explain later.

Consequently, our proposed method is categorized as a generative approach. There are many advantages inherent in generative approaches that have been examined, including unsupervised SLU (Chen et al., 2015), automatic feature extraction (Tur et al., 2013), and integration with syntactic modeling (Lorenzo et al., 2013). Another convenient property of generative models is that prior knowledge can be integrated in an intuitive way (Raymond et al., 2006). This often leads to better performance with less training data compared with discriminative models trained completely from scratch (Komatani et al., 2010).

3 Hierarchical Dirichlet Process Slot Model

In this section, we present a nonparametric Bayesian formulation that directly models the distribution over an infinite set of possible values for each slot. Let $S = \{s_1, ..., s_{M_S}\}$ be a given set of slots and M_S be the number of slots. We define each slot s_i as a random variable ranging over an infinite set of

letter sequences V, which is represented as follows:

$$V = \{b_1, ..., b_L | b_\iota \in C, L \geq 0\}$$

where C is a set of characters including the blank character and any other character that potentially appears in the transcription of an utterance. Consequently, we regard the set of slots S as also being a random variable that ranges over V^{M_S}. The objective of this section is to develop the formulation of the probabilistic distribution $p(S)$.

3.1 Dirichlet Process

We apply the Dirichlet process (DP) to model both the distribution for an individual slot $p_i(s_i)$ and the joint distribution $p(S)$. In this subsection, we review the definition and key properties of DP with general notation for the target distribution G over the domain $\mathcal{X}$. In the DP for the prior of $p_i(s_i)$ that is described in Section 3.2, the domain $\mathcal{X}$ corresponds to a set of slot values V, e.g., "fen ditton", "new chesterton", and None. In the DP for $p(S)$ presented in Section 3.3, $\mathcal{X}$ indicates a set of tuples of slot values V^{M_S}, e.g., ("restaurant", "new chesterton", "fast food") and ("restaurant", "fen ditton", None).

The DP is a probabilistic distribution over the distribution G. DP is parameterized by α^0 and G^0, where $\alpha^0 > 0$ is a concentration parameter and G^0 is a base distribution over $\mathcal{X}$. If G is drawn from $DP(\alpha^0, G^0)$ (i.e., $G \sim DP(\alpha^0, G^0)$), then the following Dirichlet distributed property holds for any partition of $\mathcal{X}$ denoted by $\{A_1, ..., A_L\}$:

$$(G(A_1), ..., G(A_L)) \sim Dir(\alpha(A_1), ..., \alpha(A_L))$$

where $\alpha(A) = \alpha^0 G^0(A)$, which is known as the base measure of DP.

Ferguson (1973) proved an important property of a posterior distribution of repeated i.i.d. samples $\mathbf{x}_{1:N} = \{x_1, ..., x_N\}$ drawn from $G \sim DP(\alpha^0, G^0)$. Consider a countably infinite set of *atoms* $\boldsymbol{\phi} = \{\phi_1, \phi_2, ...\}$ that are independently drawn from G^0. Let $c_i \in \mathbb{N}$ be the *assignment* of an atom for sample x_i, which is generated by a sequential draw with the following conditional probability:

$$p(c_{N+1} = k | \mathbf{c}_{1:N}) = \begin{cases} \frac{n_k}{N+\alpha^0} & k \leq K \\ \frac{\alpha^0}{N+\alpha^0} & k = K+1 \end{cases}$$

where n_k is the number of times that the kth atom appears in $\mathbf{c}_{1:N}$ and K is the number of different atoms in $\mathbf{c}_{1:N}$. Given the assignment $\mathbf{c}_{1:N}$, the predictive distribution of $x_{N+1} \in \mathcal{X}$ is represented in the following form:

$$P(x_{N+1} = \theta | \mathbf{c}_{1:N}, \boldsymbol{\phi}_{1:K}, \alpha^0, G^0) = \sum_{k=1}^{K} \frac{n_k}{N+\alpha^0} \delta(\phi_k, \theta) + \frac{\alpha^0}{N+\alpha^0} G^0(\theta)$$

The base distribution possibly generates an identical value for different atoms, such as (ϕ_1 = "fen ditton", ϕ_2 = "new chesterton", ϕ_3 = "fen ditton"). The assignment c_i is an auxiliary variable to indicate which of these atoms is assigned to the ith data point x_i; when x_i = "fen ditton", c_i can be 1 or 3. The posterior distribution above depends on the frequency of atom n_k, not on the frequency of θ itself. The atoms $\boldsymbol{\phi}$ and the assignment $\mathbf{c}$ are latent variables that should be determined at runtime.

3.2 Individual Slot Model

First we formulate the distribution for an individual slot as $p_i(s_i) \sim DP(\alpha_i^0, G_i^0)$ where G_i^0 is a base distribution over the set of phrases V. [1] We define G_i^0 as a generative model that consists of two-step generation: generation of the phrase length $0 \leq L_i \leq L^{max}$ using a categorical distribution and generation of a letter sequence $s_i^{1:L}$ using an n-gram model, as follows:

$$L_i \sim Categorical(\boldsymbol{\lambda}_i)$$
$$s_i^\iota \sim p(s_i^\iota | s_i^{\iota-n+1:\iota-1}, \boldsymbol{\eta}_i)$$

where $\boldsymbol{\lambda}_i$ and $\boldsymbol{\eta}_i$ are parameters for the categorical distribution and the n-gram model for slot s_i, respectively. This explicit modeling of the length helps avoid the bias toward shorter phrases and leads to a better distribution, as reported by Zhai and Boydgraber (2013). We define G_i^0 as a joint distribution of these models:

$$G_i^0(s_i^{1:L_i}) = p(L_i | \boldsymbol{\lambda}_i) \prod_{\iota=1}^{L_i} p(s_i^\iota | s_i^{\iota-n+1:\iota-1}, \boldsymbol{\eta}_i) \quad (1)$$

G_i^0 potentially generates an empty phrase of $L_i = 0$ to express the case that the slot value v_{s_i} is not

[1] Note that the subscript i for s, p, α^0 and G^0 indicates the slot type such as "type", "area" and "food" in Figure 2.

provided by an utterance. Therefore, the distribution $p_i(s_i)$ can naturally represent the probability of $None$, which is shown in Figure 2.

We consider prior distributions of the parameters $\boldsymbol{\lambda}_i$ and $\boldsymbol{\eta}_i$ to treat the n-gram characteristics of each slot in a fully Bayesian manner. $p(\boldsymbol{\lambda})$ is given as a L^{max}-dimensional symmetric Dirichlet distribution with parameter a. We also define the $|C|$-dimensional symmetric Dirichlet distributions with parameter b for each n-gram context, since given the context $p(s_i^\iota | s_i^{\iota-n+1:\iota-1}, \boldsymbol{\eta}_i)$ is just a categorical distribution that ranges over C. Consider we observe N phrases $\mathbf{s}_i$ for slot i. Let $n_{i\iota}^L$ be the number of phrases that have length ι and n_{ih}^γ be the number of times that letter $s^\iota = h$ appears after context $s^{\iota-n+1:\iota-1} = \gamma$. The predictive probability of a phrase is represented as follows:

$$G_i^0(s_i^{1:L_i}|\mathbf{s}_i) = \frac{n_{i\iota}^L + b}{N + bC} \prod_{\iota=1}^{L_i} \frac{n_{is_i^\iota}^\gamma + a}{\sum_c n_{ic}^\gamma + a \sum_{l=1}^{L^{max}} n_{il}^L}$$

3.3 Generative Model for a Set of Slot Values

A naive definition of the joint distribution $p(S)$ is a product of all slot probabilities $\prod_{i=1}^{M_S} p_i(s_i)$ for making an independence assumption. However, the slot values are generally correlated with each other (Chen et al., 2015). To obtain more accurate distribution, we formulate $p(S)$ using another DP that recognizes a frequent combination of slot values, as $p(S) \sim DP(\alpha^1, G^2)$ where G^2 is a base distribution over V^{M_S}. We apply the naive independence assumption to G^2 as follows:

$$G^2(S) = \prod_{i=1}^{M_S} p_i(s_i)$$

The whole generation process of S involves two-layered DPs that share atoms among them. In this sense, this generative model is regarded as a hierarchical Dirichlet process (Teh et al., 2005).

Let $G_i^1(s_i) = p_i(s_i)$ and $G^3(S) = p(S)$ for consistent notations. In summary, we define the hierarchical Dirichlet process slot model (HDPSM) as a generative model that has the following generation process.

$$\begin{aligned} G_i^1 &\sim DP(\alpha_i^0, G_i^0) \\ G^3 &\sim DP(\alpha^1, G^2) \\ S &\sim G^3 \end{aligned}$$

3.4 Inference of HDPSM

In a slot filling task, observations of $\mathbf{S}_{1:T} = \{S_1, ..., S_T\}$ are available as training data. The inference of HDPSM refers to the estimation of $\boldsymbol{\lambda}$, $\boldsymbol{\eta}$ and the atom assignments for each DP.

We formulate the HDPSM in a form of the Chinese restaurant franchise process, which is one of the explicit representations of hierarchical DPs obtained by marginalizing out the base distributions. Teh et al. (2005) presents a Gibbs sampler for this representation, which involves a repetitive resampling of atoms and assignment. In our method, we prefer to adopt a single pass inference, which samples the assignment for each observation only once. Our preliminary experiments showed that the quality of inference is not affected because S is observed unlike the settings in Teh et al. (2005).

We denote the atoms and the atom assignment in the first level DP $DP(\alpha^1, G^2)$ by $\boldsymbol{\phi}^1$ and $\mathbf{c}_{1:N}^1$, respectively. The posterior probability of atom assignment for a new observation S_{N+1} is represented as follows:

$$\begin{aligned} &p(c_{N+1}^1 = k | \mathbf{c}_{1:N}^1, \boldsymbol{\phi}^1, S_{N+1}) \\ &\quad \propto \begin{cases} n_k^1 \delta(\phi_k^1, S_{N+1}) & k \leq K \\ \alpha^1 G^2(S_{N+1}) & k = K+1 \end{cases} \end{aligned}$$

where n_k^1 is the number of times that the kth atom appears in $\mathbf{c}_{1:N}^1$ and K is the number of different atoms in $\mathbf{c}_{1:N}^1$.

$\boldsymbol{\phi}_i^0$ and $\mathbf{c}_{i1:K}^0$ denote the atoms and the assignment in the second level DPs $DP(\alpha_i^0, G_i^0)$. The second level DPs assign atoms to each first level atom ϕ_k^1, i.e. the second level atom ϕ_{it}^0 is generated only when a new atom is assigned for S_{N+1} at the first level. The posterior probability of atom assignment at the second level is:

$$\begin{aligned} &p(c_{iK+1}^0 = t | \mathbf{c}_{i1:K}^0, \boldsymbol{\phi}_i^0, s_{N+1i}) \\ &\quad \propto \begin{cases} n_{it}^0 \delta(\phi_{it}^0, S_{N+1}) & t \leq T_i \\ \alpha_i^0 G^0(S_{N+1}) & t = T_i + 1 \end{cases} \end{aligned}$$

where n_{it}^0 is the number of times that the tth atom appears in $\mathbf{c}_{i1:K}^0$ and T_i is the number of different atoms in $\mathbf{c}_{i1:K}^0$.

The single pass inference procedure is presented in Algorithm 1. Given the atoms $\boldsymbol{\phi}$ and the assignments $\mathbf{c}$, the predictive distribution of $S_{N+1} =$

Algorithm 1 Single pass inference of HDPSM

Input: A set of observations $\mathbf{S}_{1:N}$

Set empty list to $\mathbf{c}^1$ and $\mathbf{c}^0_i$
for $d = 1$ to N **do**
  $k \sim p(c^1_d = k|\mathbf{c}^1_{1:d-1}, \boldsymbol{\phi}^1, S_d)$
  if $k = K + 1$ **then**
    for $i = 1$ to M_S **do**
      $t_i \sim p(c^0_{iK+1} = t_i|\mathbf{c}^0_{i1:K}, \boldsymbol{\phi}^0_i, s_{di})$
      if $t_i = T_i + 1$ **then**
        Update n^L_i and n^γ_i with s_{di}
      end if
      $c^0_{K+1} \leftarrow t_i$ and $\phi^0_{it_i} \leftarrow s_{di}$
    end for
  end if
  $c^1_d \leftarrow k$ and $\phi^1_k \leftarrow S$
end for

$\{s_{N+11}, ..., s_{N+1M_S}\}$ is calculated as follows:

$$P(S_{N+1}|\mathbf{c}, \boldsymbol{\phi}) = \sum_{k=1}^{K} \frac{n^1_k}{N + \alpha^1} \delta(\phi^1_k, S_{N+1}) + \frac{\alpha^1}{N + \alpha^1} \prod_{i=1}^{M_S} P(s_{N+1i}|\mathbf{c}^0_i, \boldsymbol{\phi}^0_i) \quad (2)$$

$$P(s_{N+1i}|\mathbf{c}^0_i, \boldsymbol{\phi}^0_i) = \sum_{t=1}^{T_i} \frac{n^0_{it}}{K + \alpha^0_i} \delta(\phi^0_{it}, s_{N+1i}) + \frac{\alpha^0_i}{K + \alpha^0_i} G^0_i(s_{N+1i}|\boldsymbol{\phi}^0_i)$$

4 Generative Model for an Utterance

We present a generative utterance model to derive a slot estimation algorithm given utterance u. Figure 3 presents the basic concept of our generative model. In the proposed model, we formulate the distribution of slot values as well as the distribution of non-slot parts. In Figure 3, the phrases "hi we're in um" and "and we need a" should be removed to identify the slot information. We call these non-slot phrases as *functional fillers* because they more or less have a function to convey information. Identifying the set of non-slot phrases is equivalent to identifying the set of slot phrases. Therefore, we define a generative model of functional fillers in the same way as the slot values.

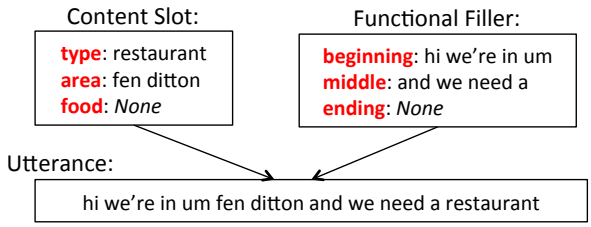

Figure 3: The proposed generative utterance model. We attempt to find the best combination of the slot parts and the non-slot parts (i.e., functional filler parts) by using this model.

4.1 Functional Filler

We assume an utterance u is a concatenation of slot values S and functional fillers F. A functional filler is represented as a phrase that ranges over V. To derive the utterance model, we first formulate a generative model for functional fillers.

In our observation, the distribution of the functional filler depends on its position in an utterance. For example, utterances often begin with typical phrases such as "Hello I'm looking for ..." or "Hi please find ...", which can hardly ever appear at other positions. To reflect this observation, we introduce a *filler slot* to separately model the functional fillers based on a position feature. Specifically, we define three filler slots: *beginning filler* f_1, which precedes any slot value, *ending filler* f_3, which appears at the end of an utterance, and *middle filler* f_2, which is inserted between slot values. We use the term *content slot* to refer to S when we intend to explicitly distinguish it from a filler slot.

Let $F = \{f_1, f_2, f_3\}$ be a set of filler slots and $M_F = 3$ be the number of filler slots. Each slot f_i is a random variable ranging over V and F is a random variable over V^{M_F}. These notations for filler slots indicate compatibility to a content slot, which suggests that we can formulate F using HDPSMs, as follows:

$$H^1_i \sim DP(\beta^0_i, H^0_i)$$
$$H^3 \sim DP(\beta^1, H^2)$$
$$F \sim H^3$$

where H^0_i is an n-gram-based distribution over V that is defined in an identical way to (1) and $H^2(F) = \prod_{i=1}^{M_F} H^1_i(F)$.

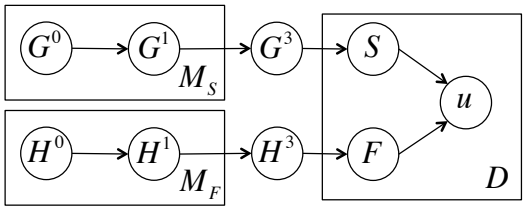

Figure 4: Graphical model of the utterance model.

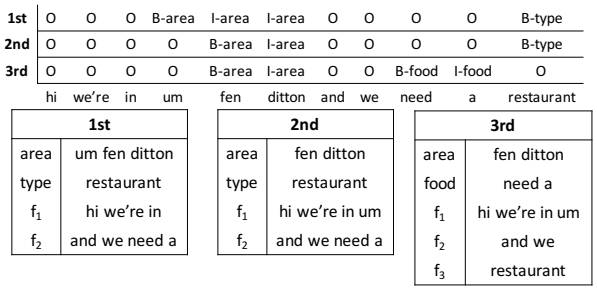

Figure 5: Candidate generation using sequential labeling algorithm. The figure shows the case of $N = 3$.

4.2 Utterance Model

Figure 4 presents the graphical model of our utterance model. We assume that an utterance u is built with phrases provided by S and F. Therefore, the conditional distribution $p(u|S, F)$ basically involves a distribution over the permutation of these slot values with two constraints: f_1 is placed first and f_3 has to be placed last. In our formulation, we simply adopt a uniform distribution over all possible permutations.

For training the utterance model, we assume that a set of annotated utterances is available. Each training instance consists of utterance u and annotated slot values S. Given u and S, we assume that the functional fillers F can be uniquely identified. For the example in Figure 3, we can identify the subsequence in u that corresponds to each content slot value of "restaurant" and "fen ditton". This matching result leads to the identification of filler slot values. Consequently, a triple (u, S, F) is regarded as an observation. Because the HDPSMs of the content slot and of the filler slot are conditionally independent given S and F, we can separately apply Algorithm 1 to train each HDPSM.

For slot filling, we examine the posterior probability of content slot values S given u, which can be reformed as follows:

$$P(S|u) \propto \sum_F P(u|S, F)P(S)P(F)$$

In this equation, we can remove the summation of F because filler slot values F are uniquely identified regarding u and S in our assumption. Additionally, we approximately regard $P(u|S, F)$ as a constant if u can be built with S and F. By using these assumptions, the posterior probability is reduced to the following formula:

$$P(S|u) \propto P(S)P(F) \quad (3)$$

where F in this formula is fillers identified given u and S. Consequently, the proposed method attempts to find the most likely combination of the slot values and the non-slot phrases, since all words in an utterance have to belong to either of them. By using trained HDPSM (i.e., the posterior given all training data), $P(S)$ and $P(F)$ can be computed by (2).

5 Candidate Generation

For estimating slot values given u, we adopt a candidate generation approach (Williams, 2014) that leverages another slot filling algorithm to enumerate likely candidates. [2] Specifically, we assume a candidate generation function $g(u)$ that generates N candidates $\{S_1, ..., S_N\}$ regarding u. Our slot filling algorithm computes the posterior probability by (3) for each candidate slot S_j and takes the candidate that has the highest posterior probability. In this estimation process, our utterance model works as a secondary filter that covers the error of the primary analysis.

Figure 5 provides an example of candidate generation by using a sequential labeling algorithm with IOB tags. The subsequences to which the O tag is assigned can be regarded as functional fillers. The values for each filler slot are identified depending on the position of the subsequence, as the figure shows.

6 Experiments

We evaluate the performance of the proposed generative model with an experiment using the algorithm

[2] The direct inference of the generative utterance model is a topic for near future work. The MCMC method will circumvent the difficulty of searching the entire candidate space.

name	#utterances	#slots	max. diversity
DSTC	1,441	6	55
Weather	1,442	3	191

Table 1: Datasets in the experiment. Max. diversity refers to the maximum number of value types that are taken by a slot.

described in Section 5. We adopt a conditional random field (CRF) as a candidate generation algorithm that generates N-best estimation as candidates. For the CRF, we apply commonly used features including unigram and bigram of the surface form and part of speech of the word. We used CRF++[3] as the CRF implementation.

6.1 Dataset

The performance of our method is evaluated using two datasets from different languages, as summarized in Table 1. The first dataset is provided by the third Dialog State Tracking Challenge (Henderson, 2015), hereafter referred to as the DSTC corpus. The DSTC corpus consists of dialogs in the tourist information domain. In our experiment, we use the user's first utterance in each dialog, which typically describes the user's query to the system. Utterances without any slot information are excluded. We manually modified the annotated slot values into "as-is form" to allow a sequential labeling method to extract the ground-truth values. This identification process can be done in a semi-automatic manner that involves no expert knowledge. We apply the part of speech tagger in NLTK[4] for the CRF application.

The second dataset is a weather corpus consisting of user utterances in an in-house corpus of human-machine dialogues in the weather domain. It contains 1,442 questions spoken in Japanese. In this corpus, the number of value types for each slot is higher than DSTC, which indicates a more challenging task. We applied the Japanese morphological analyzer MeCab (Kudo et al., 2004) to segment the Japanese text into words before applying CRF.

For both datasets, we examine the effect of the amount of available annotated utterances by varying the number of training data in $25, 50, 75, 100, 200, 400, 800, all$.

[3]https://taku910.github.io/crfpp/

[4]http://www.nltk.org/

#train	CRF best	HDP $N = 5$	HDP $N = 300$
25	0.560	0.706*	0.684*
50	0.709	0.791*	0.765*
75	0.748	0.824*	0.817*
100	0.791	0.845*	0.837*
200	0.839	0.901*	0.876*
400	0.904	0.938*	0.936*
800	0.926	0.953*	0.947*
1296	0.938	0.960*	0.951

Table 2: Slot estimation accuracy for the DSTC corpus. The asterisk (*) indicates that the accuracy is statistically significant compared against CRF best ($p < 0.005$).

#train	CRF best	HDP $N = 5$	HDP $N = 300$
25	0.327	0.452*	0.480*
50	0.379	0.488*	0.499*
75	0.397	0.504*	0.522*
100	0.418	0.501*	0.512*
200	0.493	0.526*	0.531*
400	0.512	0.551*	0.549*
800	0.533	0.555*	0.554*
1297	0.546	0.560*	0.554

Table 3: Slot estimation accuracy for the Japanese weather corpus. An asterisk (*) indicates statistical significance against CRF best ($p < 0.01$).

6.2 Evaluation Metrics

The methods are compared in terms of slot estimation accuracy. Let n_c be the number of utterances for which the estimated slot S and the ground-truth slot $\hat{S}$ are perfectly matched, and let n_e be the number of the utterances including an estimation error. The slot estimation accuracy is simply calculated as $\frac{n_c}{n_c+n_e}$. All evaluation scores are calculated as the average of 10-fold cross validation. We also conduct a binomial test to examine the statistical significance of the improvement in the proposed algorithm compared to the CRF baseline.

6.3 Results

Tables 2 and 3 present the slot estimation accuracy for the DSTC corpus and the Japanese weather corpus, respectively. The baseline (CRF best) is a method that takes only one best output of CRF for slot estimation. HDP with $N = 5$ and $N = 300$ is the proposed method, where N is the number of candidates generated by the CRF candidate genera-

utterance	estimation by CRF best	estimation by HDP $N = 5$
im looking for a restaurant that serves fast food	type:restaurant (*)	type:restaurant,food:fast food
i want a moderate restaurant in the new chesterton area	area:new chesterton, type:restaurant, food:moderate (*)	area:new chesterton, type:restaurant, pricerange:moderate
im looking for a cheap chine chinese takeaway restaurant	pricerange:cheap,type:restaurant, food:chinese takeaway	pricerange:cheap,type:restaurant, food:chine chinese takeaway (*)

Table 4: Examples of estimated slot values for the condition of #train is 800. An asterisk (*) indicates misrecognition.

tor. The asterisks (*) beside the HDP accuracy indicate the statistical significance against CRF best, which is tested using the binomial test.

Results show that our proposed method performs significantly better than CRF. Especially when the amount of training data is limited, the proposed method outperforms the baseline. This property is attractive for practical speech recognition systems that offer many different functions. Accurate recognition at an early stage of development allows a practitioner to launch a service that results in quickly collecting hundreds of speech examples.

Since we use the CRF as a candidate generator, we expect that the CRF N-best can rank the correct answer higher in the candidate list. In fact, the top five candidates cover almost all of the correct answers. Therefore, the result in the comparison of $N = 5$ and $N = 300$ suggests the stability of the proposed method against the mostly noisy 295 candidates. Because the proposed algorithm makes no use of the original ranking order, $N = 300$ is a harder condition in which to identify the correct answer. Nevertheless, the result shows that the drop in the performance is limited; the accuracy is still significantly better than the baseline. This result suggests that the proposed method is less dependent on the performance of the candidate generator.

Table 4 presents some examples of the slot values estimated by CRF best and HDP with $N = 5$ for the condition where the number of training utterances is 800. The first two are samples where CRF best failed to predict the correct values. These errors are attributed to infrequent sequential patterns caused by the less trained expressions "that serves fast food" and "moderate restaurant" because CRF is a position-based classifier. The value-based formulation allows the model to learn that the phrase "fast food" is more likely to be a food name than to be a functional filler and to reject the candidate.

The third example in Table 4 shows an error using HDP, which extracted "chine chinese takeaway" which includes a reparandum of disfluency (Georgila et al., 2010). This error can be attributed to the fact that this kind of disfluency resembles the true slot value, which leads to a higher probability of "chine" in the food slot model compared to in the functional filler model. Regarding this type of error, preliminary application of a disfluency detection method (Zayats et al., 2016) is promising for improving accuracy.

The execution time for training the proposed HDP utterance model with 1297 training data in the Japanese weather corpus was about 0.3 seconds. This is a good performance since the CRF training takes about 5.5 seconds. Moreover, the training of the proposed HDP model is scalable and works in an online manner because it is a single pass algorithm. When we have a very large number of training examples, the bottleneck is the CRF training, which requires scanning the whole dataset repeatedly.

7 Conclusion

In this paper, we proposed an arbitrary slot filling method that directly deals with the posterior probability of slot values by using nonparametric Bayesian models. We presented a two-stage method that involves an N-best candidate generation step, which is typically done using a CRF. Experimental results show that our method significantly improves recognition accuracy. This empirical evidence suggests that the value-based formulation is a promising approach for arbitrary slot filling tasks, which is worth exploring further in future work.

References

Yun-Nung Chen, William Yang Wang, Anatole Gershman, and Alexander Rudnicky. 2015. Matrix Factorization with Knowledge Graph Propagation for Unsupervised Spoken Language Understanding. In *Proc. Annual Meeting of the Association for Computational Linguistics*.

Thomas S. Ferguson. 1973. A Bayesian Analysis of Some Nonparametric Problems. *The Annual of Statistics*, 1(2):209–230.

Kallirroi Georgila, Ning Wang, and Jonathan Gratch. 2010. Cross-Domain Speech Disfluency Detection. In *Proc. Annual SIGDIAL Meeting on Discourse and Dialogue*.

Samuel J. Gershman and David M. Blei. 2012. A tutorial on Bayesian nonparametric models. *Journal of Mathematical Psychology*, 56(1):1–12.

Daniel Gildea. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2011. Producing Power-Law Distributions and Damping Word Frequencies with Two-Stage Language Models. *Journal of Machine Learning Research*, 12:2335–2382.

Stefan Hahn, Marco Dinarelli, Christian Raymond, Fabrice Lefevre, Patrick Lehnen, Renato De Mori, Alessandro Moschitti, Hermann Ney, and Giuseppe Riccardi. 2011. Comparing stochastic approaches to spoken language understanding in multiple languages. *IEEE Transactions on Audio, Speech and Language Processing*, 19(6):1569–1583.

Matthew Henderson. 2015. Machine Learning for Dialog State Tracking: A Review. In *Proc. Workshop on Machine Learning in Spoken Language Processing*.

Kazunori Komatani, Masaki Katsumaru, Mikio Nakano, Kotaro Funakoshi, Tetsuya Ogata, and Hiroshi G. Okuno. 2010. Automatic Allocation of Training Data for Rapid Prototyping. In *Proc. International Conference on Computational Linguistics*.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying Conditional Random Fields to Japanese Morphological Analysis. In *Proc. Empirical Methods in Natural Language Processing*.

John Lafferty, Andrew McCallum, and Fernando C N Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. International Conference on Machine Learning*.

Alejandra Lorenzo, Lina M Rojas-barahona, and Christophe Cerisara. 2013. Unsupervised structured semantic inference for spoken dialog reservation tasks. In *Proc. Annual SIGDIAL Meeting on Discourse and Dialogue*.

Gregoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, and Geoffrey Zweig. 2015. Using Recurrent Neural Networks for Slot Filling in Spoken Language Understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):530–539.

Angeliki Metallinou, Dan Bohus, and Jason Williams. 2013. Discriminative state tracking for spoken dialog systems. *Proc. Annual Meeting of the Association for Computational Linguistics*.

Christian Raymond, Frédéric Béchet, Renato De Mori, and Géraldine Damnati. 2006. On the use of finite state transducers for semantic interpretation. *Speech Communication*, 48(3-4):288–304.

Hang Ren, Weiqun Xu, and Yonghong Yan. 2014. Markovian discriminative modeling for dialog state tracking. In *Proc. Annual SIGDIAL Meeting on Discourse and Dialogue*.

Yee W. Teh, Michael I. Jordan, Matthew J. Beal, and David M. Blei. 2005. Hierarchical Dirichlet Processes. *Journal of the American Statistical Association*, 101:1566–1581.

Gokhan Tur, Asli Celikyilmaz, and Dilek Hakkani-Tur. 2013. Latent Semantic Modeling for Slot Filling in Conversational Understanding. In *Proc. International Conference on Acoustics, Speech and Signal Processing*.

Vedran Vukotic, Christian Raymond, and Guillaume Gravier. 2015. Is it Time to Switch to Word Embedding and Recurrent Neural Networks for Spoken Language Understanding? In *Proc. Interspeech*.

Jason D. Williams. 2010. Incremental partition recombination for efficient tracking of multiple dialog states. In *Proc. International Conference on Acoustics, Speech and Signal Processing*.

Jason D Williams. 2014. Web-style ranking and SLU combination for dialog state tracking. In *Proc. Annual SIGDIAL Meeting on Discourse and Dialogue*.

Puyang Xu and Ruhi Sarikaya. 2013. Convolutional neural network based triangular CRF for joint intent detection and slot filling. In *Proc. IEEE Workshop on Automatic Speech Recognition and Understanding*.

Vicky Zayats, Mari Ostendorf, and Hannaneh Hajishirzi. 2016. Disfluency Detection using a Bidirectional LSTM. arXiv preprint arXiv:1604.03209.

Ke Zhai and Jordan Boyd-graber. 2013. Online Latent Dirichlet Allocation with Infinite Vocabulary. In *Proc. International Conference on Machine Learning*.