BAYESIAN JOINT-SEQUENCE MODELS FOR GRAPHEME-TO-PHONEME CONVERSION

Mirko Hannemann[1,4]*, Jan Trmal*[3]*, Lucas Ondel*[1]*, Santosh Kesiraju*[2]*, Lukáš Burget*[1]

[1]Speech@FIT, Brno University of Technology, Brno, Czech Republic [2]IIIT, Hyderabad, India
[3]CLSP, Johns Hopkins University, Baltimore, MD USA [4]HLT, RWTH Aachen, Germany
ihannema, iondel, burget@fit.vutbr.cz, yenda@jhu.edu, santosh.k@research.iiit.ac.in

ABSTRACT

We describe a fully Bayesian approach to grapheme-to-phoneme conversion based on the joint-sequence model (JSM). Usually, standard smoothed n-gram language models (LM, e.g. Kneser-Ney) are used with JSMs to model graphone sequences (joint grapheme-phoneme pairs). However, we take a Bayesian approach using a hierarchical Pitman-Yor-Process LM. This provides an elegant alternative to using smoothing techniques to avoid over-training. No held-out sets and complex parameter tuning is necessary, and several convergence problems encountered in the discounted Expectation-Maximization (as used in the smoothed JSMs) are avoided. Every step is modeled by weighted finite state transducers and implemented with standard operations from the OpenFST toolkit. We evaluate our model on a standard data set (CMUdict), where it gives comparable results to the previously reported smoothed JSMs in terms of phoneme-error rate while requiring a much smaller training/testing time. Most importantly, our model can be used in a Bayesian framework and for (partly) un-supervised training.

Index Terms— Bayesian approach, joint-sequence models, weighted finite state transducers, letter-to-sound, grapheme-to-phoneme conversion, hierarchical Pitman-Yor-Process

1. INTRODUCTION

Grapheme-to-phoneme conversion (G2P) refers to the task of converting a word from its orthographic form (sequence of letters / characters / graphemes) to its pronunciation (sequence of phonemes or other types of acoustic units). G2P has its application in speech synthesis and speech recognition. However, the techniques used for G2P can be applied to any monotonous translation problem.

To avoid the effort of manual rule crafting and to be able to generalize, most of the recent approaches to G2P are data-driven and probabilistic [1]. Recent discriminative approaches to G2P (e.g. [2]) seem to slightly outperform the generative ones. Both face the problem of over-fitting to the training data, which is alleviated by smoothing (e.g. [1]) and regularization techniques (e.g. [2]). The measurement of the training progress and the tuning of the smoothing/regularization parameters is done with the help of a held-out set. We take a Bayesian approach, which has a notion of uncertainty of the model parameters, the model cannot over-train and no held-out set is necessary, i.e. all data can be used to estimate the model parameters. While smoothing changes the objective function in an ad-hoc way, no such modification of the training is necessary for the Bayesian approach. As we will show, it also results in a faster training and evaluation and can be used with latent variables, i.e. in an un-supervised or partly supervised way. Our motivation was to design a model, that can be applied in a bigger Bayesian framework, to build an unsupervised speech recognizer, which was the objective of the 2016 Jelinek Memorial Summer Workshop[1].

Relation to prior work: Many techniques have been proposed for the G2P problem [1]. Popular are smoothed joint-sequence models (SJSM) [1] and the publicly available tool Sequitur, which serves as our baseline. Our approach (Bayesian joint-sequence model, BSJM) is very similar to [1], but we use a different LM. More recent work on G2P builds mainly on discriminative approaches: e.g. [3, 2, 4, 5, 6]. However, for the Bayesian approach, generative techniques are needed. Within a framework for unsupervised acoustic unit discovery, Lee et. al. [7] jointly learns a Bayesian model for G2P. Similar to our implementation, the training uses blocked Gibbs sampling of the letter-phoneme alignment to estimate the model parameters. However, [7] use a different parametrization (based on a context window around the current letter) and apply more restrictive constraints on the possible alignments (one letter can generate 0/1/2 phones). More importantly, the use of graphone units in our case makes the model easily reversible, i.e. the same model can be applied for the G2P and P2G task. Similar to this work, Phonetisaurus [8, 9, 10, 11] (referring to [12]) also realizes G2P with the help of WFSTs and the OpenFST toolkit. Wu et. al. [13] use Phonetisaurus and OpenFST and incorporate conditional random fields and system combination. Phonetisaurus performs the segmentation (graphone alignment) as a separate step. The set of graphones is estimated using a context-less model (as an approximation to speed-up), and then a standard n-gram LM is estimated on the segmented training set. However, in our case, similar to the SJSM, we jointly estimate the segmentation and the graphone LM. As opposed to the SJSM, we do not use bottom-up model construction (step-wise 'ramping-up' and training LMs of increasing order). This approximation is not necessary in the BJSM, we can immediately train the full order LM.

2. JOINT-SEQUENCE MODELS

Joint-sequence models (JSM) [1] use a sequence of joint grapheme-phoneme units (graphones) to generate the orthographic form (letter sequence $\mathbf{g} \in G^*$) and pronunciation (phoneme sequence $\boldsymbol{\varphi} \in \Phi^*$) of a word. A graphone q is a pair of a letter sequence and a phoneme sequence of possibly different length and represents a mapping of $0..n$ letters to $0..m$ phonemes. In [1], 0..1-to-0..1 graphones where found sufficient, where each graphone corresponds to at most one letter and phoneme. The graphone inventory $\mathcal{Q}$ is usually derived automatically from data: $q = (\mathbf{g_q}, \boldsymbol{\varphi_q}) \in \mathcal{Q} \subseteq G^* \times \Phi^*$.

Fig. 1. Graphone alignment (considering only 0..1-to-0..1) for word 'mixing' [1] is a co-segmentation of spelling and pronunciation.

[1]www.clsp.jhu.edu/workshops/16-workshop/

The work reported here was carried out during the 2016 Jelinek Memorial Summer Workshop on Speech and Language Technologies, which was supported by Johns Hopkins University via DARPA LORELEI Contract No HR0011-15-2-0027, and gifts from Microsoft, Amazon, Google, Facebook.

The spelling and the pronunciation are segmented into graphones using a co-segmentation (Fig. 1): the letter sequence $\mathbf{g}$ and the phoneme sequence $\boldsymbol{\varphi}$ are grouped into an equal number of segments K. For a given pair of letter and phoneme sequence, the segmentation into graphones is usually not unique. The task of graphone segmentation is to find (all) possible graphone sequences and to calculate their probabilities. S is the set of all possible co-segmentations of $\mathbf{g}$ and $\boldsymbol{\varphi}$ (i.e. graphone sequences $\mathbf{q} \in \mathcal{Q}^*$):

$$S(\mathbf{g}, \boldsymbol{\varphi}) := \left\{ \mathbf{q} \in \mathcal{Q}^* \middle| \begin{matrix} \mathbf{g}_{q_1} \smile \ldots \smile \mathbf{g}_{q_K} \\ \boldsymbol{\varphi}_{q_1} \smile \ldots \smile \boldsymbol{\varphi}_{q_K} \end{matrix} \right\}. \quad (1)$$

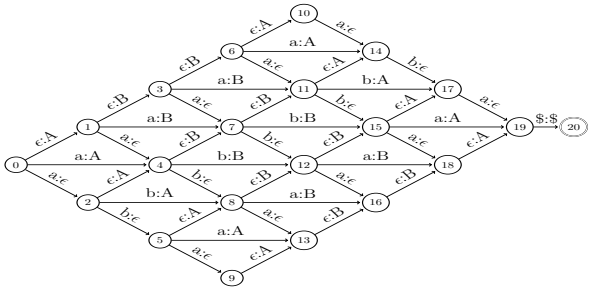

Fig. 2. Lattice of all possible co-segmentations of letters $\mathbf{g} = A, B, B, A$ and phonemes $\boldsymbol{\varphi} = a, b, a$. Each vertex corresponds to a pair of positions in $\mathbf{g}$ and $\boldsymbol{\varphi}$. Edges correspond to 0..1-to-0..1 graphones. $ indicates sentence end symbol.

The set of all possible alignments S can be represented as a lattice (Fig. 2). The joint probability $p(\mathbf{g}, \boldsymbol{\varphi})$ is determined by summing over all possible matching graphone sequences:

$$p(\mathbf{g}, \boldsymbol{\varphi}) = \sum_{\mathbf{q} \in S(\mathbf{g}, \boldsymbol{\varphi})} p(\mathbf{q}), \quad (2)$$

where the probability $p(\mathbf{q})$ of the graphone sequence $\mathbf{q}_1^K = q_1, \ldots, q_K$ (positions $j < 1$ and $j > K$ are appended with the boundary symbol) can be modeled using a graphone LM, using the standard M-gram approximation:

$$p(\mathbf{q}_1^K) \cong \prod_{j=1}^{K+1} p(q_j | q_{j-1}, \ldots, q_{j-M+1}). \quad (3)$$

To obtain a Bayesian JSM, we have to replace interpolated Kneser-Ney (KN) used in [1] with a Bayesian LM. In section 4, we introduce the hierarchical Pitman-Yor Process LM for that purpose. The task of G2P is to search for the most likely pronunciation given the orthographic form using Bayes' decision rule:

$$\boldsymbol{\varphi}(\mathbf{g}) = \arg \max_{\boldsymbol{\varphi}' \in \Phi^*} p(\mathbf{g}, \boldsymbol{\varphi}'). \quad (4)$$

3. MODEL ESTIMATION: DISCOUNTED EM

Many G2P algorithms require the grapheme-phoneme alignment (segmentation) as external input. JSMs have the advantage, that the alignment and the model parameters are optimized jointly on the training data $\mathcal{O} = (\mathbf{g}_1, \boldsymbol{\varphi}_1) \ldots (\mathbf{g}_N, \boldsymbol{\varphi}_N)$. The parameters to be estimated are the graphone M-grams $p(q_j | h_j; \boldsymbol{\vartheta})$ in (3), where $h_j = q_{j-1}, \ldots, q_{j-M+1}$ and $\boldsymbol{\vartheta}$ indicates a particular setting of the parameters. For smoothed JSMs, the training is performed with an Expectation-Maximization algorithm (EM) [1]:

$$e(q, h; \boldsymbol{\vartheta}) = \sum_{i=1}^{N} \sum_{\mathbf{q} \in S(\mathbf{g_i}, \boldsymbol{\varphi}_i)} \frac{p(\mathbf{q}; \boldsymbol{\vartheta})}{\sum_{\mathbf{q}' \in S(\mathbf{g_i}, \boldsymbol{\varphi}_i)} p(\mathbf{q}'; \boldsymbol{\vartheta})} \, n_{q,h}(\mathbf{q}). \quad (5)$$

In this expectation step, $e(q, h; \boldsymbol{\vartheta})$ is the expected number of occurrences (fractional count) of the graphone q in context h given the current parameters $\boldsymbol{\vartheta}$, and $n_{q,h}(\mathbf{q})$ is the number of times the particular graphone q occurs in the sequence $\mathbf{q}$. The set of all possible alignments can be represented as a lattice (Fig. 2), where the arc weights correspond to the likelihoods $p(q_j | h_j; \boldsymbol{\vartheta})$. The posterior probability of each arc in such a lattice can be calculated with the standard forward-backward algorithm. (5) can then be efficiently evaluated as the sum of the posteriors of all arcs corresponding to graphone q with history h. In maximum likelihood training, we start with a flat initialization of all possible graphones and we alternate the expectation and the maximization steps (5), (6):

$$p(q|h; \boldsymbol{\vartheta'}) = \frac{e(q, h; \boldsymbol{\vartheta})}{\sum_{q'} e(q', h; \boldsymbol{\vartheta})}, \quad (6)$$

where $\boldsymbol{\vartheta'}$ denotes the parameter set to be used in the next iteration. The use of this original EM guarantees that the likelihood on the training set reaches a (local) optimum, but it has several problems: it over-fits the training data, results in a huge graphone inventory, and when, in any iteration $e(q, h; \boldsymbol{\vartheta}) = 0$, the graphone $q|h$ can never emerge again in future iterations. To avoid over-fitting and to keep the set of graphones manageable, the evidence counts are smoothed and pruned. As explained in [1], smoothing in this case needs to deal with fractional counts and an interpolated KN LM is used:

$$p_M(q|h) = \frac{\max\left(e(q, h) - d_M, 0\right)}{\sum_{q'} e(q', h)} + \lambda(h) \cdot p_{M-1}(q|\bar{h}). \quad (7)$$

Here, d_M is the discount used for model order M, $\lambda(h)$ is the interpolation weight, and $p_{M-1}(q|\bar{h})$ is the lower-order distribution (using a shortened history $\bar{h}$), which recursively has exactly the same shape as p_M, but uses a different kind of evidence counts $\hat{e}(q, \bar{h})$ according to a marginal constraint (details in [1]).

The discounted EM algorithm as implemented in Sequitur [1] is:

1. Initialize all graphones (flat).
2. Compute expected counts (5).
3. Estimate new parameters (7).
4. If likelihood on held-out set improved, continue with 2.
5. Tune discounting parameters $d_1, \ldots, d_M$ by optimizing the held-out likelihood.
6. If held-out likelihood improves, continue with 2.
7. Prune model and terminate.

While, in the original EM (5), (6), the training likelihood is guaranteed to converge to a (local) optimum, for discounted EM, there are effectively two possibly conflicting objective functions: the setting of the optimal discount parameters (estimated on the held-out set) can in some cases deteriorate the training likelihood and prevent the discounted EM from converging, causing a sub-optimal termination of the overall algorithm. In order to reach the (local) optimum, it is, in some cases, necessary to manually keep the discounts small in the first few iterations until the training data 'guides' the model towards the optimum, and to apply the discounts only in the fine-tuning phase. As already pointed out in [1], starting from a larger initial graphone set (e.g. 0..2-to-0..2 graphones) always gave worse performance than when allowing only 0..1-to-0..1 graphones. Since those are a subset of the larger set, the training algorithm should be able to pick at least the same optimum.

Sequitur uses a bottom-up model construction: Starting with unigrams, the lower-order $M - 1$ model is trained until convergence, and then the higher-order model $p_M(q|h)$ is initialized with $p_{M-1}(q|\bar{h})$ (called 'ramping-up'). Here, histories h can only be constructed from $\bar{h}$ that were not pruned in the lower-order model. This greedy approximation is necessary to keep the model tractable for higher orders M and when using graphones with more than one

letters and phonemes. Surprisingly, when starting directly with a higher-order model (e.g. bigram), the training finishes in a worse local optimum than when training bottom-up (fixing the optimal set of graphones in the unigram, and training a bigram on top of that). That indicates, that the training procedure is not able to find good sets of unigram graphones, even if the bigger context should help to make an even better selection.

4. HIERARCHICAL PITMAN-YOR PROCESS LM

As seen in the last section, the implementation of the discounted EM for the SJSM needs a good deal of engineering, and sometimes it is necessary to force the model into the right direction. We therefore propose to replace the smoothed graphone LM (7) with a non-parametric Bayesian LM, and to train the model in a more principled, fully Bayesian way. To model graphone sequences, we use the Hierarchical Pitman-Yor LM (HPYLM) proposed in [16, 17]. It is necessary to be familiar with [16, 17] in order to fully understand and re-implement the following presentation, which only focuses on problems specific to applying HPYLM to the G2P task.

To start with a simpler model, assume that (3) can be modeled using a unigram LM (i.e. a categorical distribution G over a limited vocabulary of graphones $\mathcal{Q}$). In a Bayesian setting, we treat G as latent variable with a suitable prior distribution. For the unigram, we choose a Pitman-Yor process (PY) [14, 15] as the prior $P(G) = PY(d, \theta, G_0)$ with discount factor d, concentration parameter θ and base measure G_0. G_0, which is the mean of $P(G)$, can be set to the "un-informed" uniform distribution over $\mathcal{Q}$. PY can be seen as a generalization of the Dirichlet distribution[1]. Unlike the Dirichlet distribution, however, the proper setting of d can shape the tails of the prior $P(G)$ to express the preference for G to follow Zipf's law as observed in natural languages. This makes PY a suitable and effective prior in Bayesian language modeling [16, 17]. As it is standard with Bayesian inference, the final graphone LM for the G2P inference could be obtained as the predictive distribution:

$$P(q|\mathbf{q}_{train}) = \int P(q|G)P(G|\mathbf{q}_{train})\mathrm{d}G, \tag{8}$$

given a training graphone sequence $\mathbf{q}_{train}$[2]. Although there is no known analytic form for the posterior distribution over the unigram LMs $P(G|\mathbf{q}_{train}) \propto P(\mathbf{q}_{train}|G)P(G)$, the predictive distribution (8) has a tractable form in the representation as a Chinese restaurant process (CRP) [15]. In the metaphor, the Chinese restaurant has a number of tables t (unlimited), where each table k serves a single dish - a graphone (type) $q_k \in \mathcal{Q}$. Each table k holds a number of customers c_k (unlimited). The CRP assumes that the graphone sequences (e.g. $\mathbf{q}_{train}$) are generated as follows: Each new customer corresponding to the next element in the graphone sequences is either seated to one of the already existing tables ($k = 1 \dots t; c_k := c_k + 1$) or to a new table ($k = new = t + 1; c_k := 1; t := t + 1$) with probabilities:

$$P(k_j = k) = \begin{cases} \dfrac{c_k - d}{\theta + \sum c_i} & (k = 1 \dots t) \\ \dfrac{\theta + d \cdot t}{\theta + \sum c_i} & (k = new). \end{cases} \tag{9}$$

The new graphone in the sequence is then given by the dish being served at that table. For a new table, a dish $q_k \in \mathcal{Q}$ is sampled from G_0. (9) is the predictive distribution of assigning the next customer, given the assignment of all previous customers (seating arrangement). If the seating arrangement for the training data was known, (9) would directly define the predictive distribution (8). However, if $\mathcal{Q}$ is limited (G_0 discrete), there can be more than one table serving a particular graphone q. In this case, given a graphone sequence, the seating arrangement is not unique, but is a hidden variable, which needs to be inferred.

There is a Gibbs sampling scheme that can be used for the inference: In a first run over the training data, all customers are seated with probabilities proportional to (9), but with the constraint, that the customer can only be seated to tables already serving the same graphone. Then, iteratively, each customer is removed and re-seated again to a (possibly) different table, while keeping the seating arrangement of all other customers fixed. (9) is used to re-seat the customer by treating it as the last customer being added. Remembering the assigned table k_j for each customer, the removal of the customer is equivalent to decreasing c_{k_j} and, if the table becomes empty, decreasing t.

Additionally to the seating arrangement, in the Bayesian JSM, there is a second hidden variable: the graphone sequence is not directly observed but given by the co-segmentation $S(\mathbf{g}, \boldsymbol{\varphi})$ of the grapheme/phoneme sequence. We can also apply Gibbs sampling to obtain the corresponding graphone sequence. For an approximate but efficient inference, we use a blocked Gibbs sampler, where we sample the co-segmentation $S(\mathbf{g}, \boldsymbol{\varphi})$ of a whole training example (i.e. pronunciation dictionary entry) and re-seat all the corresponding customers at once. Sampling means to select one path (graphone sequence) in the alignment lattice (Fig. 2) according to the posterior probability $p(\mathbf{q}|\mathbf{g}, \boldsymbol{\varphi}; \boldsymbol{\vartheta})$. Thus, the inference procedure in the BJSM is to iterate over all training examples (dictionary entries):

1. Remove customers from old graphone sequence.
2. Sample co-segmentation $S(\mathbf{g}, \boldsymbol{\varphi})$ according to posterior.
3. Sample seating arrangements in the Chinese restaurant.

So far, we have not considered graphone context h, i.e. were using an unigram PY with uniform distribution over graphones as base measure. Now, we use a hierarchical PY LM (HPYLM) [16, 17, 18], which has achieved good results as word LM. It introduces a hierarchical structure of PY. Each PY models the distributions G_h of graphones given a particular context h, where the prior (base measure) $G_{\bar{h}}$ is given by the PY of the shortened context $\bar{h}$ (leaving out the earliest graphone). This structure corresponds to interpolating (backing-off) between higher and lower order n-grams. At the lowest hierarchy level, we use $G_0 = 1/|\mathcal{Q}|$:

$$\begin{aligned} G_1 &\sim PY(d_1, \theta_1, G_0 = 1/|\mathcal{Q}|) \\ G_{\bar{h}} &\sim PY(d_{|\bar{h}|}, \theta_{|\bar{h}|}, G_{\bar{\bar{h}}}) \\ G_h &\sim PY(d_{|h|}, \theta_{|h|}, G_{\bar{h}}) \end{aligned}$$

Again, there is an equal representation of the HPYLM with the G_h integrated out, in the form of a hierarchy of CRP, which we use to evaluate the predictive distribution $p(q|h; \boldsymbol{\vartheta})$. There is one CRP for each graphone q in context h (including the empty context $\emptyset$ for unigrams). The training of the model is done by seating customers (graphone n-gram counts $c(q|h)$) over tables $1 \dots t_{hq}$ (the number of tables for a particular graphone/context). The customers $c(q|h)$ are only directly seated at the highest order M. Every time a new table k is created, a proxy-customer is hierarchically sent down to the lower-order CRP with shortened history $\bar{h}$. Since the lower-order CRP are only updated for new tables in higher-order CRPs (graphones in unseen contexts), their distributions are not proportional to counts $c(q|\bar{h})$. The resulting equation for the graphone HYPLM resembles the interpolated KN (7) (KN is a constraint variant [17]):

$$p(q|h) = \frac{c(q|h) - d \cdot t_{hq}}{\theta + c(h)} + \frac{\theta + d \cdot t_{h\cdot}}{\theta + c(h)} \cdot p(q|\bar{h}) \tag{10}$$

[1] With $d = 0$, PY is equivalent to Dirichlet process, and degrades to Dirichlet distribution with concentration parameter θ, if G_0 is categorical.

[2] In the Bayesian JSM, graphone sequences are not directly observed but given by the co-segmentation $S(\mathbf{g}, \boldsymbol{\varphi})$. We will address this problem later.

5. IMPLEMENTATION WITH WFSTS

We implemented the whole Bayesian G2P framework with the help of weighted finite state transducers (WFST) [19] mostly using standard library functions of OpenFST <www.openfst.org/>. The generation of all possible graphone segmentations in an alignment lattice can be implemented using WFST composition.

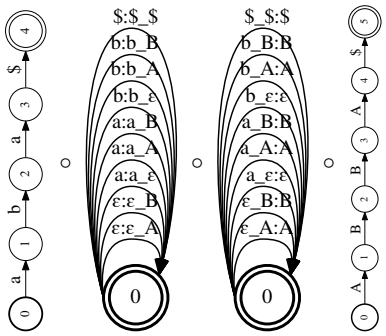

Fig. 3. Transducer chain $P \circ P2G \circ G2L \circ L$ for toy example with grapheme inventory A, B and phoneme inventory a, b. Outer left: phoneme acceptor P for $\boldsymbol{\varphi} = a, b, a$; Outer right: letter acceptor L for $\mathbf{g} = A, B, B, A$ corresponding to a pronunciation dictionary entry '$ABBA\ a\,b\,a$'. Middle part: Left: transducer $P2G$ mapping from phonemes to the set of all possible graphones. Right: $G2L$ mapping from graphones to letters.

As shown in Fig. 3, the letter sequence $\mathbf{g}$ and the phoneme sequence $\boldsymbol{\varphi}$ can be represented as linear acceptors L and P, respectively. To construct a lattice containing all possible alignments, we use two mapping transducers. In Fig. 3, transducer $P2G$ (middle left) maps from graphones to phonemes and transducer $G2L$ (middle right) maps from graphones to letters. For simplicity, we use only 0..1-to-0..1 graphones (Fig. 1), so the set of all possible graphones stays reasonable. Given these transducers, we can form a chain of compositions to produce the alignment lattice transducer (example in Fig. 3 results in Fig. 2.):

$$A = P \circ P2G \circ G2L \circ L.$$

We use a blocked Gibbs sampling approach, where we always sample a new alignment for a whole pronunciation entry (word) at once. A sample alignment is a particular path through the lattice A (Fig. 2). Also the graphone LM (HPYLM) can be represented as a WFST G,where nodes correspond to n-gram histories and each arc corresponds to a graphone (used for both input and output symbols) and its n-gram probability (arc weight). To represent an n-gram LM as WFST, we use the compact representation using back-off arcs ([19], page 19). We can apply the probabilities of the graphone HPYLM with the help of WFST composition (which corresponds to lattice re-scoring):

$$B = P \circ P2G \circ G \circ G2L \circ L. \tag{11}$$

As already pointed out by [10], to correctly evaluate the interpolated LM in the WFST framework, we need to encode the back-offs as failure arcs [20] and to use the correct matchers in the composition (phi-composition, indicated by $\circ_\varphi$). For higher-order graphone LMs, and already for small graphone inventories, the G transducer gets huge. Moreover, for a particular training example (dictionary entry), only a small portion of G is accessed. Therefore, we use OpenFST's interface for lazy composition. We implemented the HPYLM with the source code developed by Walter/Heymann [21] <https://github.com/fgnt/nhpylm> and wrote our own wrapper, that creates a lazy (on-the-fly) OpenFST WFST object.

While WFST composition is an associative operation, the grouping of compositions in (11) has an important impact on memory use and speed, especially when using lazy composition (and possibly pruning). Since the composition with G is the most costly operation and the linear acceptors P and L are the knowledge sources that constrain the possible graphone sequences, we want to apply them as early as possible, before applying G. The final composition is:

$$B = \Pi_2(\Pi_1(P \circ P2G) \circ G2L \circ L) \circ_\varphi G. \tag{12}$$

The projection operations $\Pi_1(T)$ and $\Pi_2(T)$ obtain an acceptor from WFST T by omitting the input or output labels, respectively. We project onto the output graphone symbols after composing $P \circ P2G$ and project the resulting alignment lattice A onto the input graphone symbols to obtain an acceptor lattice with graphone labels. (12) results in a 2-3x speed-up over (11). In the WFST framework, sampling the graphone sequence from the lattice B can be implemented by applying weight pushing towards the initial state in the (log) probability semi-ring, and then forward-sampling a path (graphone sequence)[3], which is used to sample a new seating arrangement in the Chinese restaurant processes.

During training, we go through all training examples in random order. Typically, 3-4 iterations through the data are sufficient to converge to a likely segmentation and seating arrangement. After each iteration, we re-sample the hyper-parameters for d and ϑ as described in [17], appendix C. Since we use Gibbs sampling to approximate $p(q|h)$, correct estimates can be obtained by averaging several HPYLM with different seating arrangements as obtained from different Gibbs sampling iterations. As a first approximation, we used just a single sample of the HPYLM in the experiments.

6. EXPERIMENTAL RESULTS AND CONCLUSIONS

We trained the HPYLM G2P on the CMUdict v0.7 kindly provided by [10]. It contains 106,837 unique training words with 113,438 pronunciations. The test set contains 12,000 unique words with 12,753 pronunciations. Our baseline is a 7-gram joint-sequence model trained with Sequitur [1] using the default settings and selected 1% of the training data as held-out set to tune the discounts. We reached 5.92% phoneme error rate (PER) and 24.65% word error rate (WER) after 11h of training, which is very close to what is reported in [1]. Using a 9-gram LM as in [1] took an additional 9h training and gave the same performance. Our Bayesian HPYLM G2P does not need a held-out set. After three iterations of sampling the training set in 2h, we reached 5.92% PER and 24.73% WER, which is basically the same as our baseline. We can expect further improvement from averaging several sampled HPYLM. With Phonetisaurus [10], we reached 5.80% PER and 24.36% WER in the order of minutes.

We presented a fully Bayesian approach to G2P, which is fully implemented with WFSTs. The Bayesian G2P based on a hierarchical Pitman-Yor-Process does not need a held-out set and complicated parameter tuning and avoids the pitfalls of the discounted EM algorithm. The Bayesian model has the same performance as the smoothed joint-sequence models. Despite the fact, that Gibbs sampling was used and the resulting models (7-grams) are already significantly large, the training is much faster than using Sequitur, but still slower than Phonetisaurus. No greedy assumptions are necessary, as e.g. the bottom-up model initialization [1] and the segmentation is done jointly in training, using full context. However, the most important advantage is that the resulting model can be used in a bigger Bayesian framework and can deal with (partly) un-annotated data.

[3]This corresponds to the forward filtering and backward sampling procedure as used in [7] and [22]. Forward filtering is the forward part of the lattice forward-backward algorithm (as in Section 3) and backward sampling picks a path according to the forward probabilities, starting from the final state.

7. REFERENCES

[1] Maximilian Bisani and Hermann Ney, "Joint-Sequence Models for Grapheme-to-Phoneme Conversion," *Elsevier Speech Communication*, vol. 50, no. 5, pp. 434–451, 2008.

[2] Keigo Kubo, Sakriani Sakti, Graham Neubig, Tomoki Toda, and Satoshi Nakamura, "Structured soft margin confidence weighted learning for grapheme-to-phoneme conversion.," in *Proceedings Interspeech*, 2014, pp. 1263–1267.

[3] Sittichai Jiampojamarn, Colin Cherry, and Grzegorz Kondrak, "Integrating joint n-gram features into a discriminative training framework," in *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the ACL*. 2010, pp. 697–700, Association for Computational Linguistics.

[4] Patrick Lehnen, Alexandre Allauzen, Thomas Lavergne, Francois Yvon, Stefan Hahn, and Hermann Ney, "Structure learning in hidden conditional random fields for grapheme-to-phoneme conversion.," in *Proceedings Interspeech*, 2013, pp. 2326–2330.

[5] Kanishka Rao, Fuchun Peng, Haşim Sak, and Françoise Beaufays, "Grapheme-to-phoneme conversion using long short-term memory recurrent neural networks," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 4225–4229.

[6] Kaisheng Yao and Geoffrey Zweig, "Sequence-to-sequence neural net models for grapheme-to-phoneme conversion," *arXiv preprint arXiv:1506.00196*, 2015.

[7] Chia-ying Lee, Yu Zhang, and James R Glass, "Joint learning of phonetic units and word pronunciations for asr.," in *Proceedings EMNLP*, 2013, pp. 182–192.

[8] Josef R Novak, Nobuaki Minematsu, and Keikichi Hirose, "Wfst-based grapheme-to-phoneme conversion: open source tools for alignment, model-building and decoding," in *10th International Workshop on Finite State Methods and Natural Language Processing*, 2012, p. 45.

[9] Josef R Novak, Nobuaki Minematsu, Keikichi Hirose, Chiori Hori, Hideki Kashioka, and Paul R Dixon, "Improving wfst-based g2p conversion with alignment constraints and rnnlm n-best rescoring.," in *Proceedings Interspeech*, 2012, pp. 2526–2529.

[10] Josef R Novak, Nobuaki Minematsu, and Keikichi Hirose, "Failure transitions for joint n-gram models and g2p conversion.," in *Proceedings Interspeech*, 2013, pp. 1821–1825.

[11] Josef Robert Novak, Nobuaki Minematsu, and Keikichi Hirose, "Phonetisaurus: Exploring grapheme-to-phoneme conversion with joint n-gram models in the wfst framework," *Natural Language Engineering*, pp. 1–32, 2015.

[12] Diamantino Caseiro, Isabel Trancoso, Luis Oliveira, and Ceu Viana, "Grapheme-to-phone using finite state transducers," in *Proc. 2002 IEEE Workshop on Speech Synthesis*, 2002, vol. 2, pp. 1349–1360.

[13] Ke Wu, Cyril Allauzen, Keith B Hall, Michael Riley, and Brian Roark, "Encoding linear models as weighted finite-state transducers.," in *Proceedings Interspeech*, 2014, pp. 1258–1262.

[14] Jim Pitman and Marc Yor, "The Two-Parameter Poisson-Dirichlet Distribution Derived from a Stable Subordinator," *The Annals of Probability*, vol. 25, no. 2, pp. 855–900, 1997.

[15] Jim Pitman, "Combinatorial stochastic processes," Tech. Rep. 621, Department of Statistics, University of California at Berkeley, 2002.

[16] Yee Whye Teh, "A hierarchical Bayesian language model based on Pitman-Yor processes," in *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2006, pp. 985–992.

[17] Yee Whye Teh, "A Bayesian interpretation of interpolated Kneser-Ney," Tech. Rep. TRA2/06, School of Computing, 2006.

[18] Sharon Goldwater, Tom Griffiths, and Mark Johnson, "Interpolating between types and tokens by estimating power-law generators," *Advances in neural information processing systems NIPS*, vol. 18, pp. 459, 2006.

[19] Mehryar Mohri, Fernando C. N. Pereira, and Michael Riley, "Speech recognition with weighted finite-state transducers," in *Handbook on Speech Processing and Speech Communication, Part E: Speech recognition*, Larry Rabiner and Fred Juang, Eds., Heidelberg, Germany, 2008, p. 31, Springer-Verlag.

[20] Alfred V Aho and Margaret J Corasick, "Efficient string matching: an aid to bibliographic search," *Communications of the ACM*, vol. 18, no. 6, pp. 333–340, 1975.

[21] Jahn Heymann, Oliver Walter, Reinhold Haeb-Umbach, and Bhiksha Raj, "Iterative bayesian word segmentation for unsupervised vocabulary discovery from phoneme lattices," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 4057–4061.

[22] Daichi Mochihashi, Takeshi Yamada, and Naonori Ueda, "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: Volume 1*. Association for Computational Linguistics, 2009, pp. 100–108.