2013

A Latent Dirichlet Allocation/N-Gram Composite Language Model

Raymond Daniel Kulhanek
Wright State University

A LATENT DIRICHLET ALLOCATION/N-GRAM COMPOSITE LANGUAGE MODEL

A thesis submitted in partial fulfilment
of the requirements for the degree of
Master of Science

By

RAYMOND DANIEL KULHANEK
B.S., Wright State University, 2010

2013
Wright State University

WRIGHT STATE UNIVERSITY
Graduate School

September 5, 2013

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY Raymond Daniel Kulhanek ENTITLED A Latent Dirichlet Allocation/N-gram Composite Language Model BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Master of Science.

Shaojun Wang, Ph.D.
Thesis Director

Mateen Rizki, Ph.D.
Chair, Department of Computer Science and Engineering

Committee on
Final Examination

Shaojun Wang, Ph.D.

John Gallagher, Ph.D.

Mateen Rizki, Ph.D.

R. William Ayres, Ph.D.
Interim Dean, Graduate School

ABSTRACT

Kulhanek, Raymond Daniel. M.S. Department of Computer Science and Engineering, Wright State University, 2013. A Latent Dirichlet Allocation/N-gram Composite Language Model.

I present a composite language model in which an n-gram language model is integrated with the Latent Dirichlet Allocation topic clustering model. I also describe a parallel architecture that allows this model to be trained over large corpora and present experimental results that show how the composite model compares to a standard n-gram model over corpora of varying size.

Contents

List of Figures

List of Tables

ACKNOWLEDGEMENTS

This work was supported in part by an allocation of computing time from the Ohio Supercomputer Center and by the AFRL/DAGSI Ohio Student-Faculty Research Fellowship Program (project number RH1-WSU-12-1).

Chapter 1

Introduction

1.1 Language Models

Language models are probability distributions that allow us to evaluate the likelihood that a sequence of words is a valid sample of a given natural language. Specifically, given a sequence of words $\mathbf{w}$, they give the probability that you will get $\mathbf{w}$ if you randomly draw a sequence from the set of all sequences of length $|\mathbf{w}|$ in the language.

This can prove useful for applications such as machine translation and automatic speech recognition though the use of the noisy channel model [Shannon 1948]. Under the noisy channel model, we have some information signal, which could be a sentence in a foreign language, an audio recording, or anything else that can be considered an encoding of the English sentence that we wish to recover. We assume that the output signal F was produced as the output of a system that takes an English sentence E as input and transforms it into F via a series of (possibly noisy) transformations. We wish to find the source sentence E that was most likely to have generated that signal, $\arg\max_{\mathbf{w}} \mathsf{P}(\mathbf{w} \mid F)$. To do so, we decompose that probability into a product of independent probabilities:

$$\arg\max_{\mathbf{w}} \mathsf{P}(\mathbf{w} \mid F) = \arg\max_{\mathbf{w}} \frac{\mathsf{P}(F \mid \mathbf{w})\,\mathsf{P}(\mathbf{w})}{\mathsf{P}(F)} = \arg\max_{\mathbf{w}} \mathsf{P}(F \mid \mathbf{w})\,\mathsf{P}(\mathbf{w}) \tag{1.1}$$

$\mathsf{P}(F \mid \mathbf{w})$ might then be further decomposed based on the nature of the specific

channel, but the important thing is that we have separated the question of "What constitutes a reasonable English sentence?" from the manner in which the sentence is encoded. Thus, a language model that allows us to estimate $\mathsf{P}(\mathbf{w})$ can be used regardless of the exact nature of the channel and encoding.

By far the most common class of language model is the n-gram. In this model, the sequence of words is assumed to be generated by an $(n-1)^{\text{th}}$ order Markov Chain, a generalization of normal Markov Chains [Markov 1906, 1910] in which each node is conditionally independent of all other nodes given the $(n-1)$ previous nodes in the sequence: $\mathsf{P}(w_i \mid w_1, ..., w_{i-1}) = \mathsf{P}(w_i \mid w_{i-n+1}, ..., w_{i-1})$. This is an effective model, but it is easy to see how the number of probabilities that need to be trained will increase exponentially with the order of the model.

A common measurement of the quality of a language model is perplexity. Not to be confused with the (related) term of the same name from information theory, perplexity is defined here to be $\exp(-\log \mathsf{P}(\mathbf{w}) / |\mathbf{w}|)$, the geometric mean over $\mathbf{w}$ of $1/\mathsf{P}(w_i \mid \cdot)$ [Stolcke 2002]. Thus, minimizing perplexity maximizes the average probability per word. It is important to note that the perplexity of a given set may vary wildly based on changes to the vocabulary or the manner in which out of vocabulary words are handled, among other factors. Thus, perplexity is only useful as a way of comparing the relative quality of two systems that handle these factors the same way, rather than as an absolute measure of quality.

1.2 Latent Dirichlet Allocation

An alternative to n-gram models is the topic model, in which the probability of a word is not conditioned on the preceding words but rather on some estimate as to the topics covered in the document. It should be noted that although the topics selected by the computer may correspond to the sort of things that a human would consider to be topics, they could instead correlate to aspects of the writing style or characteristics that have mathematical meaning but no intuitive description. All the computer will

care about is predictive power. There are a number of different approaches that can be used here, such as Latent Semantic Analysis [Deerwester et al. 1990; Dumais 2004], Probabilistic Latent Semantic Analysis [Hofmann 1999], and Latent Dirichlet Allocation [Blei et al. 2003], which is what I will be using here.

Latent Dirichlet Allocation is a multilevel topic clustering model in which for each document, a parameter vector θ for a multinomial distribution is drawn from a Dirichlet distribution, parameterized on the constant α. For each word in the document, an index into a finite set of k topic distributions z is selected from the θ multinomial, and then one of V possible words w is selected from the row of a matrix β, each row of which corresponds to a multinomial over words, and where the row is selected using the chosen topic index. w_n is a V-dimensional vector such that $w_{nj} = \begin{cases} 1 \text{ if word n is j} \\ 0 \text{ otherwise} \end{cases}$. Taken together, the probability of a document of N words is thus:

$$\begin{aligned} \mathsf{P}(\mathbf{w} \mid \boldsymbol{\alpha}, \boldsymbol{\beta}) &= \int \mathsf{P}(\boldsymbol{\theta} \mid \boldsymbol{\alpha}) \left(\prod_{n=1}^{N} \sum_{z_n} \mathsf{P}(z_n \mid \boldsymbol{\theta}) \mathsf{P}(w_n \mid z_n, \boldsymbol{\beta}) \right) \mathrm{d}\boldsymbol{\theta} \\ &= \frac{\Gamma\left(\sum_i \alpha_i\right)}{\prod_i \Gamma(\alpha_i)} \int \left(\prod_{i=1}^{k} \theta_i^{\alpha_i - 1} \right) \left(\prod_{n=1}^{N} \sum_{i=1}^{k} \prod_{j=1}^{V} (\theta_i \beta_{ij})^{w_{nj}} \right) \mathrm{d}\boldsymbol{\theta} \end{aligned} \tag{1.2}$$

The number of words in a document is assumed to be drawn from a Poisson distribution in the underlying generative model, but since in practice, LDA is used to evaluate the probability of existing documents, this is not a factor in practice.

Due to the intractability of evaluating Eq. 1.2, Blei et al. [2003] uses the variational approach from Jordan et al. [1999], in which an adjustable lower bound for Eq. 1.2 is defined in terms of a set of variational parameters:

$$q(\boldsymbol{\theta}, \mathbf{z} \mid \boldsymbol{\gamma}, \boldsymbol{\phi}) = q(\boldsymbol{\theta} \mid \boldsymbol{\gamma}) \prod_{n=1}^{N} q(z_n \mid \phi_n) \tag{1.3}$$

A variational EM algorithm is then run in which during the E-step, the variational parameters $\boldsymbol{\phi}$ and $\boldsymbol{\gamma}$ are selected to minimize the Kullback-Leibler divergence to

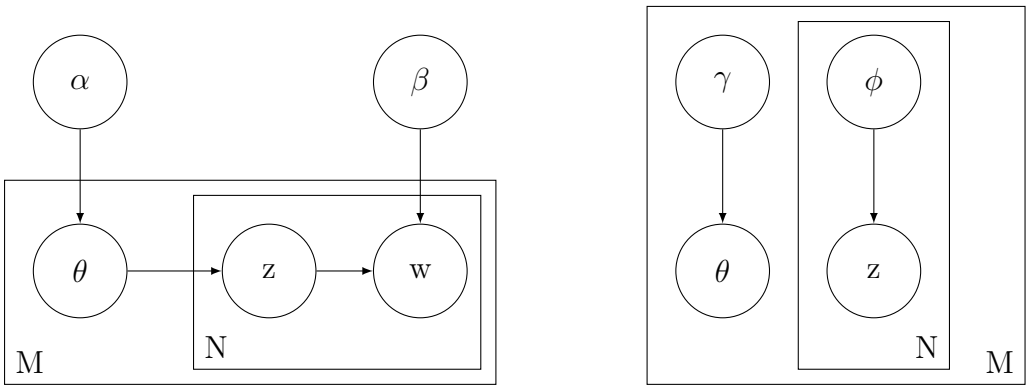


Figure 1.1: Unigram LDA (left) and Variational Distribution (right)

$\mathsf{P}(\boldsymbol{\theta}, \mathbf{z} \mid \mathbf{w}, \alpha, \boldsymbol{\beta})$ (where p is the true distribution and q is the variational distribution), and during the M-step, α and β are selected to maximize the probability of the corpus under the current variational model (The Kullback-Leibler divergence between distributions P and Q is defined as $\mathrm{KL}(P||Q) = \sum_x \mathsf{P}(x) \log(\mathsf{P}(x)/\mathsf{Q}(x))$ [Kullback and Leibler 1951]).

I use a similar approach for the composite model, modified to take word history into account.

1.3 Related Work

Wallach developed a bigram/LDA model, trained using Gibbs Sampling. The underlying graphical model is fairly similar to the one I use, except restricted to bigrams instead of arbitrary n-grams [Wallach 2006].

Wang, McCallum, and Wei introduce the Topical N-gram Model, which adds an additional Boolean latent variable between each pair of words in Wallach's model that indicates whether the latter word is to be conditioned on the former, as in M. Steyvers' and T. Griffiths' LDA Collocation model [Steyvers and Griffiths 2005, 2007].

The new variable is conditioned on both the previous word and the previous topic, where in LDA Collocation, it depends only on the word [Wang et al. 2007].

Tan, Zhou, Zheng, and Wang [Tan et al. 2011] developed a composite model

of n-grams, Structured Language Modeling (SLM) [Chelba and Jelinek 2000], and Probabilistic Latent Semantic Analysis (PLSA) [Hofmann 1999].

1.4 Chapter Organization

In Chapter 2, I describe the altered graphical model I used to integrate word history into the probability estimate of future words, and how the update equations for the variational EM procedure are affected. In Chapter 3, I discuss how computational and memory requirements scale as the order of the n-gram model is increased and describe the manner in which I parallelize the training of the model. In Chapter 4, I describe the experiments undertaken using the composite model and the results thereof. Chapter 5 summarizes my conclusions. Appendix A goes into more detail on the derivations of the equations from Chapter 2.

Chapter 2

LDA/N-gram Composite Model

2.1 Base Model

The graphical model for LDA/n-gram is similar to the original model shown in Figure 1.1. Each word node w_i becomes dependent on $w_{i-1} \dots w_{i-H+1}$ for an H^{th} order n-gram model, and the $\boldsymbol{\beta}$ matrix is greatly expanded in size, becoming a $k \times \underbrace{V \times \cdots \times V}_{H}$ element tensor. I make the simplifying assumption that each topic node z_i is independent of all other topic nodes given $\mathbf{w}$, in order to somewhat preserve the assumption made by LDA that the topics are infinitely exchangeable, as well as to keep the variational model tractable. Thus, we end up with the model shown in Figure 2.1.

For parameter estimation, I use the same variational model as in the unigram LDA model in Blei et al. [2003], as the modifications to the true model are all based around $\mathbf{w}$ and $\boldsymbol{\beta}$, both of which were dropped from the variational model in order render it tractable. As the remaining variables are document specific rather than corpus or vocabulary wide, the lack of increased dimensionality is not a problem; $\boldsymbol{\phi}$ and $\mathbf{z}$ now condition the probability of the i$^{\text{th}}$ n-gram in the document matching the observation rather than the i$^{\text{th}}$ word, but the number of observations remains the same (or slightly lower, as during training, the first few words are ignored since a word history has not been established yet).

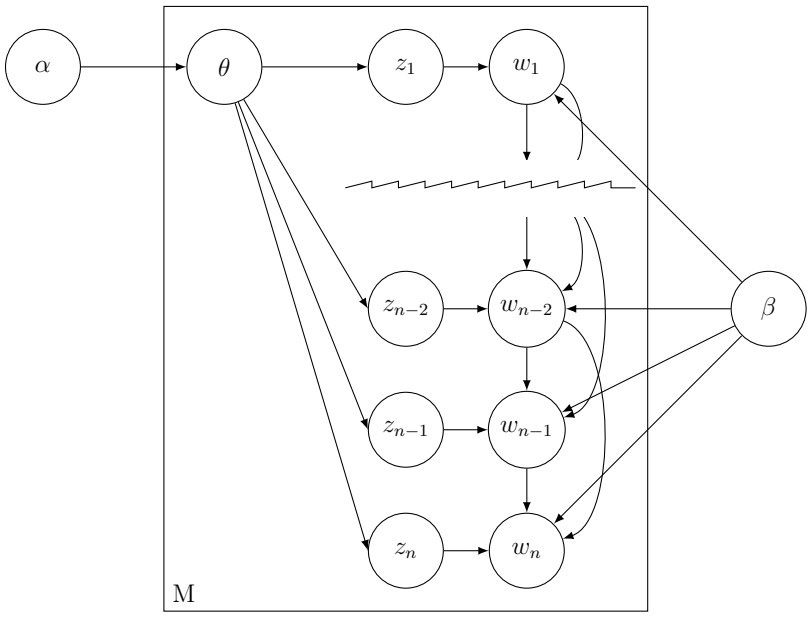


Figure 2.1: LDA/N-Gram Graphical Model (for N=3)

Complete derivations of the lower bound estimate and update equations can be found in Appendix A.2, but the end results are that $\boldsymbol{\phi}$'s update equation becomes:

$$\phi_{ni} \propto \sum_{\mathbf{h}} h_n \beta_{ij\boldsymbol{h}} \exp\left(\Psi\left(\gamma_i\right) + \Psi\left(\sum_{j=1}^{k} \gamma_j\right)\right) \tag{2.1}$$

where $\beta_{ij\boldsymbol{h}} = \mathsf{P}\left(w_n = j \mid k = i, w_{n-1}^{n-H} = \boldsymbol{h}\right)$ and $\boldsymbol{h}$ is the word history at that point. Note that $\boldsymbol{\phi}$ is document-specific, so n fully determines both the word and word history at that point for that document. $\boldsymbol{\beta}$'s update equation becomes:

$$\beta_{ij\boldsymbol{h}} = \left(\sum_{d=1}^{M}\sum_{n=1}^{N_d} \phi_{dni} h_n\right)^{-1} \sum_{d=1}^{M}\sum_{n=1}^{N_d} \phi_{dni} w_{dnj} h_n \tag{2.2}$$

where $h_n = \begin{cases} 1 & \text{if word history at } n = \mathbf{h} \\ 0 & \text{otherwise.} \end{cases}$

$\boldsymbol{\gamma}$'s update equation remains unchanged and is still:

$$\gamma_i = \alpha + \sum_{n=1}^{N} \phi_{ni} \tag{2.3}$$

α is still optimized using a linear time Newton-Raphson variant.

2.2 Smoothing

Once the initial training completes, it outputs a joint probability distribution $\mathsf{P}\left(w_{i-H+1}...w_i \mid \mathbf{z}\right)$. From this distribution, a set of discretized expected counts for each word sequence is generated separately under each topic, where

$$C_z(w_{i-H+1}...w_i) = \max\left(1, \left\lfloor 0.5 + \mathsf{P}\left(w_{i-H+1}...w_i \mid z\right) \sum_{d=1}^{M} N_d \right\rfloor\right) \tag{2.4}$$

for all observed sequences. The count files are then fed to MITLM's estimate-ngram tool [Hsu and Glass 2008] in order to generate a model for each topic that was smoothed using Modified Kneser-Ney [Chen and Goodman 1998].

Chapter 3

Program

3.1 Time and Space Complexity

The bounds given here are assuming the data representation used in the actual code, rather than the theoretical maximum. Let I_{em} be the maximum number of iterations of the EM loop and I_{var} be the maximum number of iterations of variational inference loop (per iteration of the EM loop). N_d is the length of document d. Then the time complexity of parameter estimation is:

$$T(\text{parameter estimation}) = O\left(I_{em} I_{var} \sum_{d=1}^{M} N_d \left(H \log V + k\right) + I_{em} k H V\right) \tag{3.1}$$

The calculation of expected probability of a test document is essentially just a single iteration of the e-step for that document and its complexity is thus:

$$T(\text{test likelihood}) = O\left(I_{var} N \left(H \log V + k\right)\right) \tag{3.2}$$

Note that the $\log V$ terms are likely to be smaller in practice due to the sparsity of observed words at the lower levels of the n-gram structure.

The space complexity for both parameter estimation and evaluation of test likeli-

hood is:

$$O\left(k(M + \max_d N_d) + kV^H + \sum_d^M N_d\right) \tag{3.3}$$

In the actual implementation, the kV^H term is vastly smaller, due to the same sparsity mentioned earlier. The exact sizes of the n-gram tables vary based on the characteristics of the language as well as the size of the training corpus and cannot be easily quantified, but in practice they tended towards something on the order of $k\prod_{h=1}^{H} Ve^{-\lambda h}$ for some constant λ.

3.2 Program Architecture

The program used Blei's original implementation of LDA [Blei 2006] as a starting point and was modified to integrate word history into all probabilities and to be efficient over training sets up to approximately one billion words. My experiments were run on a multicore machine with a large amount of RAM, so the program currently uses threads for parallelization; however, it could be converted to run on a cluster with minimal trouble. I only describe the training phase below. To calculate the probability of a test corpus, the program simply runs the E-step using the smoothed probabilities from Section 2.2 instead of the maximum likelihood estimates.

During the E-step, a number of threads are spawned, each of which is assigned a subset of the training corpus. For each document within the subset, the algorithm will compute the optimal $\boldsymbol{\phi}$ matrix and $\boldsymbol{\gamma}$ vector, per the equations in Chapter 2. As these parameters depend solely on the current document, there need be no coordination between threads to compute them. Each thread will need to make several queries to the $\boldsymbol{\beta}$ table to determine the current estimated log probability of the n-grams in its sub-corpus, but these are all read-only; so again, no coordination is necessary. The large number of evaluations of the digamma function during this step proved to be a bottleneck, so the code uses the Mortici [Mortici 2010] or Muqattash-Yahdi

[Muqattash and Yahdi 2006] approximations when possible and only relies on the (vastly slower) Taylor approximation over the range where the former approximations have poor accuracy.

Once the likelihood and variational parameters have been computed, the thread must update the expected counts for its observed n-grams. Technically, this is the beginning of the M-step, but in the actual code, it fits better with the E-step since it is parallelized over documents rather than n-grams. This part does have the potential for interference with other threads, so locking is required. Each entry in the n-gram table has its own associated mutex to keep the overhead to a minimum. The memory needed for this process is not insignificant, but is preferable to allowing significant contention over a smaller number of locks.

Next, the set of expected counts for all n-grams are marginalized over word histories and normalized in order to calculate the conditional log probabilities of each word given a word history and topic (i.e. the β tables). Again, these computations are split among a set of threads, with each thread being responsible for a subset of the topics. Although this potentially leaves some threads idle if the number of threads is greater than the number of topics, it allows the calculations to be carried out without any locking if the number of topics is large. As those are the runs that take the most time in practice, the trade-off is worth it. Once that is done, α is optimized numerically.

For an n-gram model with k topics over a V word vocabulary, the $\boldsymbol{\beta}$ matrix (corresponding to the log probabilities and expected counts) becomes a $k \times \underbrace{V \times \cdots \times V}_{N}$ tensor, which would quickly exhaust the memory of even a cluster of machines. Fortunately, the tensor is very sparse at higher n-gram orders, but it is still the dominant factor for memory use. Of the various representations I tried for the data, the one that yields the best combination of speed and memory use is storing the n-grams in a set of sorted arrays (with a separate array for each order) and computing an array of offsets into each n-gram array, indexed by the first word of the n-gram, such that the index of n-gram $a, b, ..., z$ is guaranteed to be in $[\,\text{offsets}_a, \text{offsets}_{a+1})$ if it exists

in the array at all. This range is generally small enough that a binary search over it takes negligible time. Compared to this, a normal hash function takes much more memory, and a minimal perfect hash requires significant time to generate at higher n-gram orders. Furthermore, there is no guarantee that the resulting hash will be simple enough to be faster than the offset array/binary search approach.

A similar approach is used for mapping strings onto their word number, with the index into the offset array being the five least significant bits of the first two characters of the word, since all lower case letters are unique in those bits under ASCII. The vocabulary array is sorted first according to that hash function and then asciibetically within each offset range.

Should the program be converted to run over a cluster of machines, a set of machines should be dedicated to storing the $\boldsymbol{\beta}$ table, with each machine being given all values corresponding to a subset of hash indices. The subsets should be selected such that it is improbable that a single node would receive the bulk of the more frequent n-grams, so the load would be fairly distributed. The E-step would be largely unchanged; the lookups of log probabilities and calls that add to expected counts would be a bit slower due to the network overhead, but would otherwise be identical. As there is no other communication between threads, the addition of more threads running on separate nodes would not change anything significant.

The first part of the M-step would require only minor modification, as the needed locks are already present: the E-step node would make a blocking request, and the server would spawn a thread that tries to acquire the lock as usual and respond once it has the information.

The normalization part of the M-step could be left largely unchanged if only the storage is distributed, or the number of total threads (possibly spread across multiple nodes) does not exceed the number of topics. If responsibility for a given topic were split across multiple threads/nodes, further locking would be required.

The $\boldsymbol{\beta}$ tensor is initialized in one of two ways. It can be randomly populated, or

it can be initialized using a lower order LDA/n-gram model such that:

$$C_H\left(w_i \mid w_{i-1}...w_{i-1+H}, k\right) = P_{H-1}\left(w_i \mid w_{i-1}...w_{i-2+H}, k\right) \tag{3.4}$$

$$C_1\left(w_i \mid k\right) = \text{random}(0,1) + \frac{1}{V} \tag{3.5}$$

In practice, the latter approach leads to slightly better models, as well as faster convergence on the later steps and is the one used in the experiments. However, the difference is not sufficient to make the seeded start a necessity.

Chapter 4

Experiments

For the experiments, I used subsets of the LDC English Gigaword corpus [Graff and Cieri 2003]. The exact subsets are listed in Table 4.1. The baseline used was a modified Kneser-Ney smoothed n-gram model of the same order as the experiment. MITLM's estimate-ngram tool was used to train the baseline, and SRILM's ngram tool [Stolcke 2002] was used to evaluate the perplexity of the test set.

Experiments showed that the composite model yielded a lower perplexity over the test set at the unigram and bigram level for all these corpora, but at the trigram level, it had difficulty generalizing to unseen data. As Figure 4.1 shows, the test perplexity was worse than the baseline when trained on the 44 million word set, about the same at the 230 million level, and finally showed improvement when trained on the full 1.3 billion words. The number of topics that could be supported also increased with the training corpus size. The composite model did perform better over the training set, suggesting over-fitting; however, various experiments (not shown) showed that loosening the convergence criteria or lowering the maximum number of iterations made things worse, so I concluded that the problem is simply that a large training set is needed to train the large number of parameters in the higher order models.

The termination conditions for the training algorithm fall into two sets: the criteria for terminating the overall EM algorithm and those for terminating the variational parameter estimation within each EM iteration. For the majority of the experiments,

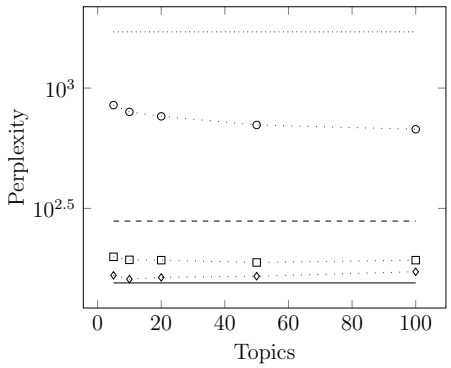
1.3 billion word training corpus	
AFP	1994-05 to 1996-10
APW	1994-11 to 1996-04
NYT	1994-07 to 1994-11
NYT	1995-01
NYT	1995-04 to 2003-01
NYT	2004-07 to 2004-09
XIN	1997-09 to 2001-11
XIN	2002-01 to 2004-11

230 million word training corpus	
AFP	1994-05 to 1996-10
APW	1994-11 to 1996-04
NYT	1994-07 to 1994-11

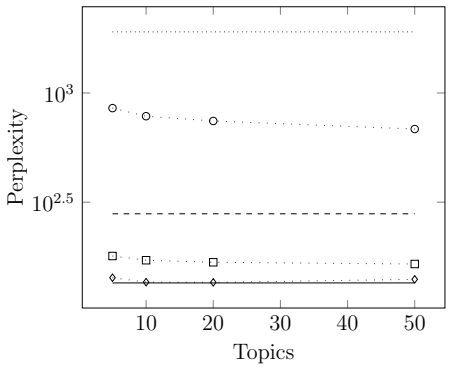
44 million word training corpus	
AFP	1994-06 to 1995-07

1.8 million word test corpus	
XIN	2004-11

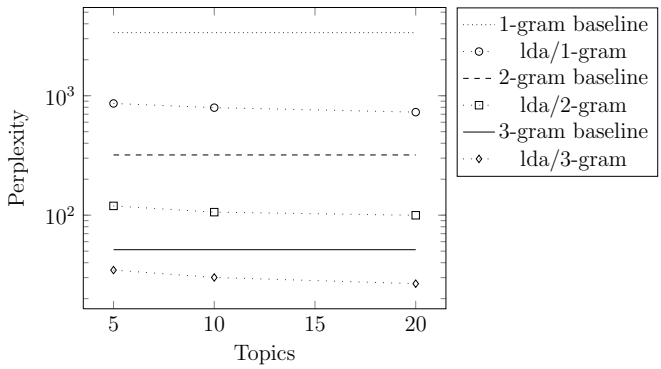
Table 4.1: Corpora Used



(a) 44M Corpus

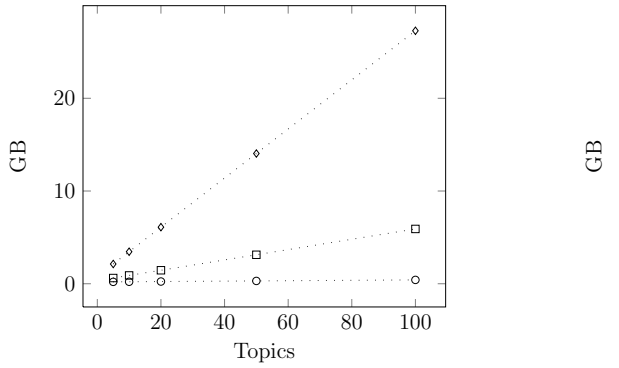


(b) 230M Corpus

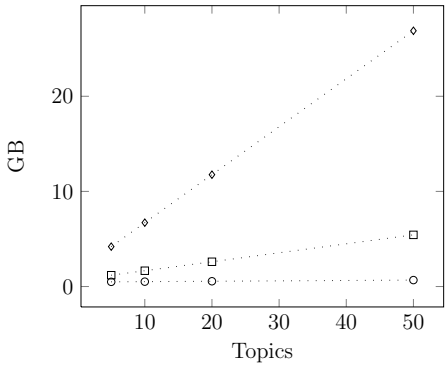


(c) 1.3B Corpus

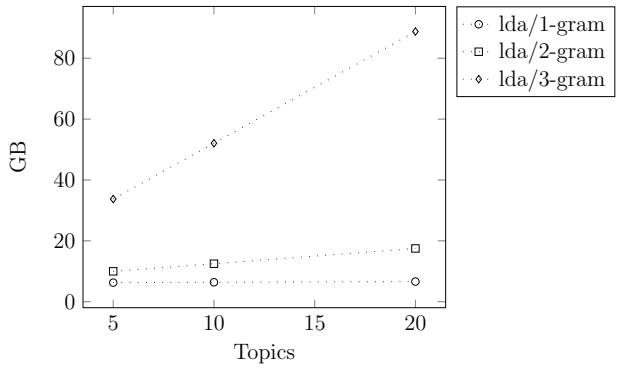
Figure 4.1: Results



(a) 44M Corpus

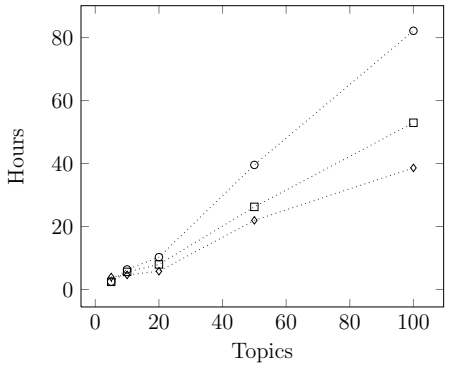


(b) 230M Corpus

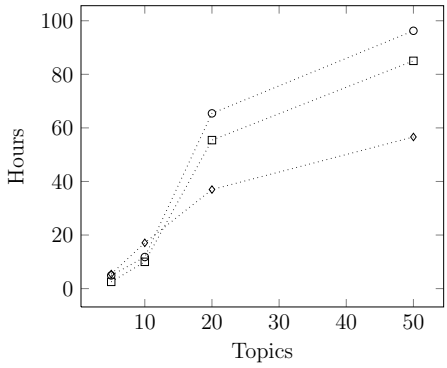


(c) 1.3B Corpus

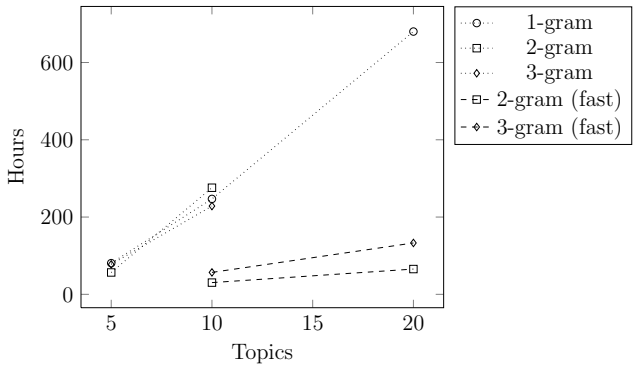
Figure 4.2: Memory Use



(a) 44M Corpus



(b) 230M Corpus



(c) 1.3B Corpus

Figure 4.3: Training times for joint distributions over word sequences

the EM algorithm was terminated when the difference in log probability of the training set between two iterations dropped below 10^{-5} or after 60 iterations, whichever came first. The estimation of the variational parameters on each EM iteration terminated when the difference in log probability dropped below 10^{-5} or after 50 iterations. For the 1.3 billion word, 20-topic experiments, I relaxed the convergence criteria to 10^{-4}, which yields slightly worse results, but is much faster. The unigram results for the 1.3 billion word, 20 topic experiments (and thus the initial values used for the bigram experiment) used the slower convergence criteria, since the run was already mostly done when I decided that the 10^{-5} convergence criteria was too slow on that set. For the 1.3 billion word, 10-topic experiments, I used both the fast and slow criteria to estimate how much quality I was sacrificing: the bigram perplexity was only 3% higher and the trigram perplexity was 2% higher. The 44 million word, 10 and 20 topic experiments and the 230 million word, 20 topic experiment were run with a convergence threshold of 10^{-6} and a maximum of 80 iterations (as they were run before I selected the criteria I used for most of the experiments and I forgot to rerun them). I ran the 44 million word, 10 topic experiment with both the 10^{-5} and 10^{-6} convergence criteria (although with an 80 iteration maximum instead of 60 for the former), and they showed only a 1% difference in perplexity, which is why I loosened the criteria in the first place.

Figure 4.3 shows the amount of time needed to train the model. It only considers the training of the parameters α and $\boldsymbol{\beta}$, i.e. the shape parameter of the Dirichlet distribution and the joint word sequence distributions given the topic. This accounts for the majority of the total training time and is the part that is least dependent on system load at the time the job is scheduled. The "fast" entries are the ones that used a 10^{-4} convergence criteria instead of 10^{-5}.

Note that the higher order models tend to take less time to train than the lower order ones, despite the fact that the algorithm's theoretical runtime does scale directly with n-gram order. This can be at least partly explained by the time saved by having

the lower order model provide a better starting position; the unigram runs generally take more EM iterations to converge than the higher order models. The trigram runs do not generally take fewer EM iterations than the bigram runs, but in the runs where the time difference is most pronounced, they do take fewer iterations to fit the variational model on average. There are a few cases where the higher order is faster but does not take noticeably fewer iterations, either EM or variational; however, these tend to be the ones where the difference in time is minor, so I believe the starting position to be the dominant factor.

Chapter 5

Conclusion

5.1 Conclusion

Although the model did eventually show better trigram results than the standard Kneser-Ney baseline, the amount of training data that it needed before it could generalize properly to unseen data was far more than is needed to train a trigram model normally, and the amount of training time needed also greatly exceeds that of the baseline. That said, the LDA/n-gram model did show a significant improvement over the baseline once it had enough data that it was able to generalize well and also appeared to be able to support a larger number of topics than I was able to use in the experiments due to time requirements. Given these facts, it may be worth pursuing composite LDA/n-gram models in general, but the variational inference approach simply has too many parameters that need to be optimized. Unless a way of reducing the dimensionality can be found, the variational approach should be abandoned.

5.2 Future Work

There are two directions of research that might improve the generalizability of the model. The first is to use a different training method in hopes that it was specifically the variational EM algorithm that caused the problem. However, I have doubts as to how well this will work. The majority of the parameters to be fit were in the $\boldsymbol{\beta}$

table, which would be present in some form no matter the training method, while the variational parameters had the same dimensionality as in unigram LDA.

The second approach would be to actively reduce the dimensionality of the model by mapping the words onto a lower dimensional underlying semantic space, as was done in Bengio et al. [2003]. However, this would require that LDA's generative model be modified to act over a continuous semantic space rather than a discrete set of words, which may not be possible. If it is, however, this approach would be preferred, as it attacks the curse of dimensionality directly rather than just working around it.

Additionally, there are a few experiments that I would still like to run, and for which time constraints did not permit me to do so. I integrated this model into the Moses machine translation system [Hoang et al. 2007] as a feature function alongside the standard Moses features (including a normal trigram model). When the LDA/n-gram feature was used at the bigram level (the highest level at which it showed improvement in perplexity at the time), it did not lead to any improvement in BLEU score over the test corpus. I would like to run the same experiment using the 1.3 billion word trigram models. If this experiment shows an improvement in BLEU score, it would give more reason to pursue the alternate approaches from the paragraphs above, as the model would be known to have value in real-world systems as well as in more theoretical experiments.

Appendix A

Derivations of Update Equations

Let n be the number of words in the document, k be the number of topics, H be the n-gram order, V be the vocabulary size, N be the number of words in the document (and N_d be the number of words in document d), and M be the number of documents in the corpus.

A.1 True model

$$\begin{aligned}\mathsf{P}(\mathbf{w}, \boldsymbol{\theta}, \mathbf{z}, \boldsymbol{\beta}, \alpha) &= \mathsf{P}(\mathbf{w}, \mathbf{z}, \boldsymbol{\beta} \mid \boldsymbol{\theta})\, \mathsf{P}(\boldsymbol{\theta} \mid \alpha)\, \mathsf{P}(\alpha) \\ &= \mathsf{P}(\mathbf{w} \mid \mathbf{z}, \boldsymbol{\beta})\, \mathsf{P}(\boldsymbol{\beta})\, \mathsf{P}(\mathbf{z} \mid \boldsymbol{\theta})\, \mathsf{P}(\boldsymbol{\theta} \mid \alpha)\, \mathsf{P}(\alpha) \end{aligned} \tag{A.1}$$

$$\mathsf{P}(\mathbf{z} \mid \boldsymbol{\theta}) = \prod_{i=1}^{n} \mathsf{P}(z_i \mid \theta_i) \tag{A.2}$$

$$\mathsf{P}(\mathbf{w} \mid \mathbf{z}, \boldsymbol{\beta}) = \prod_{i=1}^{n} \mathsf{P}\left(w_i \mid w_{i-1}^{i-N}, z_i, \boldsymbol{\beta}\right) \tag{A.3}$$

$$\mathsf{P}(\mathbf{w}, \boldsymbol{\theta}, \mathbf{z} \mid \alpha, \boldsymbol{\beta}) = \mathsf{P}(\boldsymbol{\theta} \mid \alpha) \prod_{i=1}^{n} \left[\mathsf{P}(z_i \mid \boldsymbol{\theta})\, \mathsf{P}\left(w_i \mid w_{i-1}^{i-H}, z_i, \boldsymbol{\beta}\right)\right] \tag{A.4}$$

$$\begin{aligned}\mathsf{P}(\mathbf{w} \mid \alpha, \boldsymbol{\beta}) &= \int \mathsf{P}(\mathbf{w}, \boldsymbol{\theta} \mid \alpha, \boldsymbol{\beta}) \, \mathrm{d}\boldsymbol{\theta} \\ &= \int \mathsf{P}(\boldsymbol{\theta} \mid \alpha) \prod_{i=1}^{n} \left[\sum_{j=1}^{k} \left[\mathsf{P}(z_i = j \mid \theta_j) \, \mathsf{P}\left(w_i \mid w_{i-1}^{i-H}, \boldsymbol{\beta}, z_i = j\right) \right] \right] \mathrm{d}\boldsymbol{\theta}\end{aligned} \tag{A.5}$$

A.2 Variational model

As in the variational inference of unigram LDA, we seek to maximize $\mathsf{P}(\mathbf{w} \mid \alpha, \boldsymbol{\beta})$ by minimizing the Kullback-Leibler (KL) divergence between the variational distribution and the true distribution, and then maximizing α and $\boldsymbol{\beta}$ under the variational model. From Blei et al. [2003], we know that:

$$\log \mathsf{P}(\mathbf{w} \mid \alpha, \boldsymbol{\beta}) = \mathcal{L}(\boldsymbol{\gamma}, \boldsymbol{\phi}; \alpha, \boldsymbol{\beta}) + KL\left(q(\boldsymbol{\theta}, \mathbf{z} \mid \boldsymbol{\gamma}, \boldsymbol{\phi}) \,||\, \mathsf{P}(\boldsymbol{\theta}, \mathbf{z} \mid \mathbf{w}, \alpha, \boldsymbol{\beta})\right)$$

$$\begin{aligned}\text{where } \mathcal{L}(\boldsymbol{\gamma}, \boldsymbol{\phi}; \alpha, \boldsymbol{\beta}) = {} & E_q[\log \mathsf{P}(\boldsymbol{\theta} \mid \alpha)] + E_q[\log \mathsf{P}(\mathbf{z} \mid \boldsymbol{\theta})] + E_q[\log \mathsf{P}(\mathbf{w} \mid \mathbf{z}, \boldsymbol{\beta})] \\ & - E_q[\log q(\boldsymbol{\theta})] - E_q[\log q(\mathbf{z})]\end{aligned} \tag{A.6}$$

(where q is the probability under the variational model), and therefore maximizing $\mathcal{L}(\boldsymbol{\gamma}, \boldsymbol{\phi}; \alpha, \boldsymbol{\beta})$ will minimize the Kullback-Leibler divergence between the distributions. Due to the conditional independence implied by the altered true model, and the variational model being unchanged, most of the terms of $\mathcal{L}(\boldsymbol{\gamma}, \boldsymbol{\phi}; \alpha, \boldsymbol{\beta})$ remain the same as under the original model. $E_q[\log \mathsf{P}(\mathbf{w} \mid \mathbf{z}, \boldsymbol{\beta})]$, however, requires minor alterations in order to take into account the dependence on word history, encoded in the expanded $\boldsymbol{\beta}$ tensor.

Let $\Psi(x)$ be the digamma function, $\frac{\mathrm{d}}{\mathrm{dx}} \log \Gamma(x)$. Let w_n be a V-dimensional vector, z_n be a k-dimensional vector, and h_n be a V^H-dimensional vector such that $w_{nj} = \begin{cases} 1 \text{ if word } n = j \\ 0 \quad \text{otherwise} \end{cases}$, $z_{ni} = \begin{cases} 1 \text{ if topic } n = i \\ 0 \quad \text{otherwise} \end{cases}$, and $h_n = \begin{cases} 1 \text{ if word history at } n = \mathbf{h} \\ 0 \qquad \text{otherwise} \end{cases}$
Let $\beta_{ij\boldsymbol{h}}$ be the element of the $\boldsymbol{\beta}$ tensor corresponding to word j, topic i, and word history $\mathbf{h}$. $\beta_{ij\boldsymbol{h}} = \mathsf{P}\left(w_{nj} = 1 \mid z_{ni} = 1, w_{n-1}^{n-H} = \mathbf{h}\right)$. And let $\sum_{\mathbf{h}}(\cdot) = \sum_{h_1=1}^{V} \cdots \sum_{h_{H-1}=1}^{V}(\cdot)$. Then:

$$
\begin{aligned}
Eq\left[\log \mathsf{P}\left(\mathbf{w} \mid \mathbf{z}, \boldsymbol{\beta}\right)\right] &= Eq\left[\sum_{n=1}^{N} \log \mathsf{P}\left(w_n \mid w_{n-1}^{n-H}, z_n, \boldsymbol{\beta}\right)\right] \\
&= Eq\left[\sum_{n=1}^{N}\sum_{j=1}^{V} w_{nj} \log \mathsf{P}\left(w_{nj} \mid w_{n-1}^{n-H}, z_n, \boldsymbol{\beta}\right)\right] \\
&= Eq\left[\sum_{n=1}^{N}\sum_{j=1}^{V}\sum_{i=1}^{k} w_{nj} z_{ni} \log \mathsf{P}\left(w_{nj} \mid w_{n-1}^{n-H}, z_n, \boldsymbol{\beta}\right)\right] \\
&= Eq\left[\sum_{n=1}^{N}\sum_{j=1}^{V}\sum_{i=1}^{k}\sum_{\mathbf{h}} w_{nj} z_{ni} h_n \log \beta_{ij\boldsymbol{h}}\right] \\
&= \sum_{n=1}^{N}\sum_{j=1}^{V}\sum_{i=1}^{k}\sum_{\mathbf{h}} w_{nj} h_n \phi_{ni} \log \beta_{ij\boldsymbol{h_n}} \qquad \text{(A.7)}
\end{aligned}
$$

Thus the full expansion of $\mathcal{L}\left(\boldsymbol{\gamma}, \boldsymbol{\phi}; \alpha, \boldsymbol{\beta}\right)$ becomes:

$$
\begin{aligned}
&\mathcal{L}\left(\boldsymbol{\gamma}, \boldsymbol{\phi}; \alpha, \boldsymbol{\beta}\right) \qquad \text{(A.8)} \\
&= \log \Gamma\left(\sum_{j=1}^{k} \alpha_j\right) - \sum_{i=1}^{k} \log \Gamma\left(\alpha_i\right) + \sum_{i=1}^{k} \left(\alpha_i - 1\right)\left(\Psi\left(\gamma_i\right) - \Psi\left(\sum_{j=1}^{k} \gamma_j\right)\right) \\
&+ \sum_{n=1}^{N}\sum_{i=1}^{k} \phi_{ni}\left(\Psi\left(\gamma_i\right) - \Psi\left(\sum_{j=1}^{k} \gamma_j\right)\right) \\
&+ \sum_{n=1}^{N}\sum_{j=1}^{V}\sum_{i=1}^{k}\sum_{\mathbf{h}} w_{nj} h_n \phi_{ni} \log \beta_{ij\boldsymbol{h}} \\
&- \log \Gamma\left(\sum_{j=1}^{k} \gamma_j\right) + \sum_{i=1}^{k} \log \Gamma\left(\gamma_i\right) - \sum_{i=1}^{k} \left(\gamma_i - 1\right)\left(\Psi\left(\gamma_i\right) - \Psi\left(\sum_{j=1}^{k} \gamma_j\right)\right) \\
&- \sum_{n=1}^{N}\sum_{i=1}^{k} \phi_{ni} \log \phi_{ni} \qquad \text{(A.9)}
\end{aligned}
$$

where all lines except the third were derived in Blei et al. [2003].

As the differences in the model are all encoded in $\boldsymbol{\beta}$, the optimization of the parameters can be kept fairly similar to that of the original paper, as long as all lookups of the $\boldsymbol{\beta}$ matrix are redirected to the appropriate part of the tensor. As in the original paper, I form a Lagrangian that constrains $\boldsymbol{\phi}$, $\boldsymbol{\gamma}$, and $\boldsymbol{\beta}_{i.\mathbf{h_n}}$ to each sum

to 1. None of the terms containing α or $\boldsymbol{\gamma}$ have been changed, so only the terms relating to $\boldsymbol{\phi}$ and $\boldsymbol{\beta}$ must be updated at all.

$$\mathcal{L}_{[\phi_{ni}]} = \phi_{ni}\left(\Psi\left(\gamma_i\right) - \Psi\left(\sum_{j=1}^{k}\gamma_j\right)\right) + \phi_{ni}\sum_{j=1}^{V}\sum_{\mathbf{h}} w_{nj}h_n \log\beta_{ij\boldsymbol{h}} - \phi_{ni}\log\phi_{ni} + \lambda_n\left(-1 + \sum_{i=1}^{k}\phi_{ni}\right) \tag{A.10}$$

$$0 = \frac{\partial\mathcal{L}}{\partial\phi_{ni}} = \Psi\left(\gamma_i\right) - \Psi\left(\sum_{j=1}^{k}\gamma_j\right) + \sum_{\mathbf{h}} h_n\log\beta_{ij\boldsymbol{h}} - \log\phi_{ni} - 1 + \lambda \tag{A.11}$$

Therefore the optimal value for ϕ_{ni} is:

$$\phi_{ni} \propto \sum_{\mathbf{h}} h_n\beta_{ij\boldsymbol{h}}\exp\left(\Psi\left(\gamma_i\right) + \Psi\left(\sum_{j=1}^{k}\gamma_j\right)\right) \tag{A.12}$$

Next, we derive the optimal β, under the constraint that $\beta_{ij\boldsymbol{h}}$ sums to 1 over j.

$$\mathcal{L}_{[\boldsymbol{\beta}]} = \sum_{\mathbf{h}}\sum_{i=1}^{k}\left[\left(\sum_{d=1}^{M}\sum_{n=1}^{N_d}\sum_{j=1}^{V}\phi_{dni}w_{dnj}h_n\log\beta_{ij\boldsymbol{h}}\right) + \lambda_{i\mathbf{h}}\left(-1 + \sum_{j=1}^{V}\beta_{ij\boldsymbol{h}}\right)\right] \tag{A.13}$$

$$\begin{aligned} 0 &= \frac{\partial\mathcal{L}[\beta]}{\partial\beta_{ij\boldsymbol{h}}} \\ &= \frac{\partial}{\partial\beta_{ij\boldsymbol{h}}}\sum_{d=1}^{M}\sum_{n=1}^{N_d}\phi_{dni}w_{dnj}h_n\log\beta_{ij\boldsymbol{h}} + \lambda_{i\mathbf{h}}\beta_{ij\boldsymbol{h}} - \lambda_{i\mathbf{h}} \\ &= \sum_{d=1}^{M}\sum_{n=1}^{N_d}\phi_{dni}w_{dnj}h_n{\beta_{ij\boldsymbol{h}}}^{-1} + \lambda_{i\mathbf{h}} \end{aligned} \tag{A.14}$$

$$\text{So } \lambda_{i\mathbf{h}} = -\sum_{d=1}^{M}\sum_{n=1}^{N_d}\phi_{dni}w_{dnj}h_n{\beta_{ij\boldsymbol{h}}}^{-1} \text{ and } \beta_{ij\boldsymbol{h}} = -{\lambda_{i\mathbf{h}}}^{-1}\sum_{d=1}^{M}\sum_{n=1}^{N_d}\phi_{dni}w_{dni}h_n \tag{A.15}$$

$$
\begin{aligned}
0 &= \frac{\partial \mathcal{L}[\beta]}{\partial \lambda_{i\mathbf{h}}} \\
&= \frac{\partial}{\partial \lambda_{i\mathbf{h}}} \sum_{d=1}^{M}\sum_{n=1}^{N_d}\sum_{j=1}^{V} \phi_{dni} w_{dnj} h_n \log\left[-\lambda_{i\mathbf{h}}^{-1}\sum_{d=1}^{M}\sum_{n=1}^{N_d}\phi_{dni} w_{dnj} h_n\right] \\
&\quad + \lambda_{i\mathbf{h}}\left(\sum_{j=1}^{V}\left[-{\lambda_{i\mathbf{h}}}^{-1}\sum_{d=1}^{M}\sum_{n=1}^{N_d}\phi_{dni} w_{dnj} h_n\right]\right) - \lambda_{i\mathbf{h}} \\
&= \frac{\partial}{\partial \lambda_{i\mathbf{h}}} \sum_{d=1}^{M}\sum_{n=1}^{N_d}\sum_{j=1}^{V} \phi_{dni} w_{dnj} h_n \log\left(-\lambda_{i\mathbf{h}}^{-1}\right) - \lambda_{i\mathbf{h}} \\
&= -\sum_{d=1}^{M}\sum_{n=1}^{N_d}\sum_{j=1}^{V} \phi_{dni} w_{dnj} h_n \lambda_{i\mathbf{h}}^{-1} - 1
\end{aligned} \tag{A.16}
$$

$$
\text{So } {\lambda_{i\mathbf{h}}}^{-1} = -\left(\sum_{d=1}^{M}\sum_{n=1}^{N_d}\sum_{j=1}^{V} \phi_{dni} w_{dnj} h_n\right)^{-1} \tag{A.17}
$$

Therefore the optimal value for $\beta_{ij\boldsymbol{h}}$ is:

$$
\begin{aligned}
\beta_{ij\boldsymbol{h}} &= \left(\sum_{d=1}^{M}\sum_{n=1}^{N_d}\sum_{j=1}^{V} \phi_{dni} w_{dnj} h_n\right)^{-1} \sum_{d=1}^{M}\sum_{n=1}^{N_d} \phi_{dni} w_{dnj} h_n \\
&= \left(\sum_{d=1}^{M}\sum_{n=1}^{N_d} \phi_{dni} h_n \sum_{j=1}^{V} w_{dnj}\right)^{-1} \sum_{d=1}^{M}\sum_{n=1}^{N_d} \phi_{dni} w_{dnj} h_n \\
&= \left(\sum_{d=1}^{M}\sum_{n=1}^{N_d} \phi_{dni} h_n\right)^{-1} \sum_{d=1}^{M}\sum_{n=1}^{N_d} \phi_{dni} w_{dnj} h_n
\end{aligned} \tag{A.18}
$$

The update equation for γ_i is not affected by the use of n-grams, so it remains $\gamma_i = \alpha + \sum_{n=1}^{N} \phi_{ni}$. Similarly, α is still calculated using a linear-time Newton-Raphson method [Blei et al. 2003].

References

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. 2003. A neural probabilistic language model. *Journal of Machine Learning Research 3*, 1137–1155.

Blei, D. M. 2006. Latent dirichlet allocation in C. `http://www.cs.princeton.edu/~blei/lda-c/`.

Blei, D. M., Ng, A. Y., and Jordan, M. I. 2003. Latent dirichlet allocation. *The Journal of Machine Learning Research 3*, 993–1022.

Chelba, C. and Jelinek, F. 2000. Structured language modeling. *Computer Speech and Language 14*, 283–332.

Chen, S. and Goodman, J. 1998. An empirical study of smoothing techniques for language modeling. Tech. Rep. TR-10-98, Computer Science Group, Harvard University.

Deerwester, S. C., Dumais, S. T., Landauer, T. K., Furnas, G. W., and Harshman, R. A. 1990. Indexing by latent semantic analysis. *JASIS 41,* 6, 391–407.

Dumais, S. T. 2004. Latent semantic analysis. *Annual review of information science and technology 38,* 1, 188–230.

Graff, D. and Cieri, C. 2003. English gigaword corpus. Linguistic Data Consortium, Philidelphia.

Hoang, H., Birch, A., Callison-burch, C., Zens, R., Aachen, R., Constantin, A., Federico, M., Bertoldi, N., Dyer, C., Cowan, B., Shen, W., Moran, C., and Bojar, O. 2007. Moses: Open source toolkit for statistical machine translation. Association for Computational Linguistics, 177–180.

Hofmann, T. 1999. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on research and development in information retrieval*. SIGIR '99. ACM, 50–57.

Hsu, B. and Glass, J. 2008. Iterative language model estimation: Efficient data structure & algorithms. In *Proc. Interspeech*.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. 1999. An introduction to variational methods for graphical models. *Machine Learning 37*, 183–233.

Kullback, S. and Leibler, R. A. 1951. On information and sufficiency. *The Annals of Mathematical Statistics 22,* 1, 79–86.

Markov, A. 1906. The extension of the law of large numbers onto quantities depending on each other. In *Probability and Statistics. Russian Papers*, O. Sheynin, Ed. NG Verlag. Translated by O. Sheynin.

Markov, A. 1910. An investigation of the general case of trials connected into a chain. In *Probability and Statistics. Russian Papers*, O. Sheynin, Ed. NG Verlag. Translated by O. Sheynin.

Mortici, C. 2010. The proof of the Muqattash-Yahdi conjecture. *Mathematical and Computer Modelling 51,* 9-10, 1154 – 1159.

Muqattash, I. and Yahdi, M. 2006. Infinite family of approximations of the digamma function. *Mathematical and Computer Modelling 43,* 11-12, 1329 – 1336.

Shannon, C. E. 1948. A mathematical theory of communication. *The Bell Technical Journal 27,* 4, 379–423.

Steyvers, M. and Griffiths, T. 2005. Topic modeling toolbox. `http://psiexp.ss.uci.edu/research/programs_data/toolbox.htm`.

Steyvers, M. and Griffiths, T. 2007. Topics in semantic representation. *Psychological Review 114(2)*, 211–244.

Stolcke, A. 2002. SRILM - an extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Language Processing*. Vol. 2. 901–904.

Tan, M., Zhou, W., Zheng, L., and Wang, S. 2011. A large scale distributed syntactic, semantic, and lexical language model for machine translation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*. HLT '11. Association for Computational Linguistics, 201 – 210.

Wallach, H. M. 2006. Topic modeling: beyond bag-of-words. In *Proceedings of the 23rd international conference on Machine learning*. ICML '06. ACM, New York, NY, USA, 977–984.

Wang, X., McCallum, A., and Wei, X. 2007. Topical n-grams: Phrase and topic discovery, with an application to information retrieval. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*. 697 –702.