Low Rank Language Models for Small Training Sets

Brian Hutchinson, *Student Member, IEEE*, Mari Ostendorf, *Fellow, IEEE*, and Maryam Fazel, *Member, IEEE*

***Abstract*—Several language model smoothing techniques are available that are effective for a variety of tasks; however, training with small data sets is still difficult. This letter introduces the low rank language model, which uses a low rank tensor representation of joint probability distributions for parameter-tying and optimizes likelihood under a rank constraint. It obtains lower perplexity than standard smoothing techniques when the training set is small and also leads to perplexity reduction when used in domain adaptation via interpolation with a general, out-of-domain model.**

***Index Terms*—Language model, low rank tensor.**

I. Introduction

LANGUAGE model smoothing has been well studied, and it is widely known that performance improves substantially when training on large data sets. Empirical studies show that the modified Kneser–Ney method works well over a range of training set sizes on a variety of sources [1], although other methods are more effective when pruning is used in training large language models [2]. While large training sets are valuable, there are situations where they are not available, including system prototyping for a new domain or training language models that specialize for communicative goals or roles. This letter addresses the problem of language model training from sparse data sources by casting the smoothing problem as low rank tensor estimation. By permitting precise control over model complexity, our low rank language models are able to fit the small in-domain data with better generalization performance.

II. Rank in Language Modeling

Every n-gram language model implicitly defines an nth-order joint probability tensor $\mathcal{T}$:

$$P(w_1 w_2 \dots w_n = i_1 i_2 \dots i_n) = \mathcal{T}_{i_1 i_2 \dots i_n}. \tag{1}$$

An unsmoothed maximum likelihood-estimated language model can be viewed as an entrywise sparse tensor. The obvious problem with parameterizing a language model directly by the entries of $\mathcal{T}$ is that, under nearly all conditions, there are too many degrees of freedom for reliable estimation. Hence, a substantial amount of research has gone into the smoothing of n-gram language models.

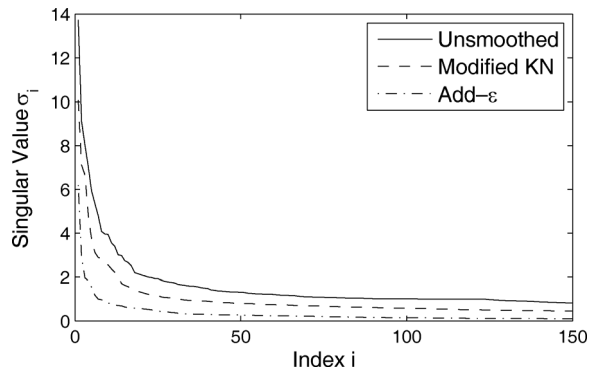

Fig. 1. Singular values of conditional probability matrix for unsmoothed (solid line), modified-KN smoothed (dashed), and add-ϵ smoothed (dash-dotted) models. Trained on 150 K words of broadcast news text, with a 5 K word vocabulary.

One can compare different smoothing methods by their effects on the properties of $\mathcal{T}$. In particular, even highly distinct approaches to smoothing have the effect of reducing, either exactly or approximately, the rank of the tensor $\mathcal{T}$. Reducing the rank implies a reduction in model complexity, yielding a model that is easier to estimate from the finite amount of training data. In the matrix case, reducing the rank of the joint probability matrix is equivalent to pushing the distributions over a vocabulary of size V, $P(\cdot|w_{t-1})$, either exactly or approximately into a subspace of $\mathbb{R}^V$. More generally, a low rank tensor implies that the set of distributions $P(\cdot|\cdot)$ are largely governed by a common set of $R \ll V$ underlying factors. Although the factors need not be interpretable, and certainly not predefined, one might envision that a set of syntactic (e.g., part-of-speech), style and/or semantic factors could account for much of the observed sequence behavior in natural language. Fig. 1 illustrates the rank-reducing phenomenon of smoothing in a conditional probability matrix. Both modified Kneser–Ney and add-ϵ [1] smoothing shrink the mass of the singular values over the unsmoothed estimate. This effect is most pronounced on small training sets, which require more smoothing.

The number of factors (the rank R) of a tensor thus provides a mechanism to control model complexity. The benefits of controlling model complexity are well-known: a model that is too expressive can overfit the training, while a model that is too simple may not be able to capture the inherent structure. By reducing the mass of singular values, existing smoothing methods effectively reduce the complexity of the model. Although they do so in a meaningful and interpretable way, it is unlikely that any fixed approach to smoothing will be *optimal*, in the sense that it may return a model whose complexity is somewhat more or somewhat less than ideal for the given training data. In this

Manuscript received April 14, 2011; revised June 09, 2011; accepted June 14, 2011. Date of publication June 27, 2011; date of current version July 07, 2011. This work was supported in part by NSF CAREER Grant ECCS-0847077 and by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA). All statements of fact, opinion or conclusions contained herein are those of the authors and should not be construed as representing the official views or policies of the NSF, IARPA, the ODNI or the U.S. Government. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Constantine L. Kotropoulos.

The authors are with the Department of Electrical Engineering, University of Washington, Seattle, WA 98195 USA (e-mail: brianhutchinson@ee.washington.edu; mo@ee.washington.edu; mfazel@ee.washington.edu).

Digital Object Identifier 10.1109/LSP.2011.2160850

letter we test the hypothesis that it is the rank-reducing behavior that is important in the generalization of smoothed language models, and propose a model and training approach better matched to this objective.

III. Background

Let R denote the tensor rank, n the order of the n-gram model, V the size of the vocabulary, and K the number of n-gram tokens in the training set. Let $\mathbb{R}^p$ denote the set of vectors of length p, and $\mathbb{R}^{p\times q}$ the set of $p \times q$ matrices. Tensors are written in script font (e.g., $\mathcal{T}$).

A. Tensor Rank

Tensor rank generalizes matrix rank. The rank of an nth-order tensor $\mathcal{T}$ is defined to be the smallest R for which there exist $\lambda \in \mathbb{R}^R, F^{(1)}, F^{(2)}, \ldots, F^{(n)} \in \mathbb{R}^{R\times V}$ such that

$$\mathcal{T} = \sum_{r=1}^{R} \lambda_r F_r^{(1)} \otimes F_r^{(2)} \otimes \cdots \otimes F_r^{(n)}. \tag{2}$$

Here, $\otimes$ denotes the tensor product, a generalization of the outer product: the $(i_1, i_2, \ldots, i_n)$th entry of $(F_r^{(1)} \otimes F_r^{(2)} \otimes \cdots \otimes F_r^{(n)})$ is $F_{ri_1} F_{ri_2} \cdots F_{ri_n}$. Like the singular value decomposition for matrices, (2) decomposes a tensor into the sum of rank-1 components.

B. Tensor Rank Minimization

There are two dominant approaches to estimating low rank tensors. One approach solves an optimization problem that penalizes the tensor rank, which encourages but does not impose a low rank solution:

$$\min_{\mathcal{T}\in\mathbb{T}} \quad f(\mathcal{T}) + \operatorname{rank}(\mathcal{T}). \tag{3}$$

($\mathbb{T}$ denotes the feasible set.) When the tensor is order-3 or higher, not only is this problem NP-hard [3], but there are no tractable convex relaxations of the notion of rank in (2). Recently, researchers [4], [5] have proposed a nuclear norm relaxation of a different concept of rank, built upon the tensor n-rank [6]. Under their relaxation, assuming f and $\mathbb{T}$ are convex, (3) is convex, but the approach requires $O(V^n)$ memory, which is prohibitive for reasonable size vocabularies.

Instead, one can impose a hard rank constraint:

$$\min_{\mathcal{T}\in\mathbb{T}, \operatorname{rank}(\mathcal{T})\le R} \quad f(\mathcal{T}). \tag{4}$$

In this nonconvex problem, R is a predetermined hard limit; in practice, the problem is solved repeatedly for different R and the best result is used. This approach allows one to reduce the space complexity to $O(nRV)$ by explicitly encoding the parameters in the low-rank factored form of (2), which makes scaling to real-world datasets practical.

IV. Low Rank Language Models

A. Model

Our low rank language models (LRLMs) represent n-gram probabilities in a factored tensor form:

$$P(w_{t-n+1}, \ldots, w_t = i_1 i_2 \ldots i_n) = \mathcal{T}_{i_1 i_2 \ldots i_n}$$
$$= \sum_{r=1}^{R} \lambda_r F_{ri_1}^{(1)} F_{ri_2}^{(2)} \ldots F_{ri_n}^{(n)}. \tag{5}$$

The model is parametrized by the non-negative component weights $\lambda \in \mathbb{R}^R$ and the factor matrices $F^{(i)} \in \mathbb{R}^{R\times V}$.

Because $\mathcal{T}$ denotes a joint probability distribution, we must impose that $\mathcal{T}$ is entry-wise non-negative and sums to one. We will see later that requiring our parameters to be non-negative provides substantial benefits for interpretability and leads to an efficient training algorithm. Technically, R denotes the *non-negative* tensor rank, which is never less than the tensor rank.

Because all of the parameters in (5) are non-negative, we can constrain the rows of $F^{(i)}$ to sum to one. It is then sufficient to constrain λ to sum to one for $\mathcal{T}$ to sum to one. Under these constraints, the rows of the factor matrices can be interpreted as position-dependent unigram models over our vocabulary, and the elements of λ as priors on each component:

$$P(w_{t-n+1}, \ldots, w_t = i_1 \ldots i_n) = \sum_{r=1}^{R} \lambda_r F_{ri_1}^{(1)} F_{ri_2}^{(2)} \ldots F_{ri_n}^{(n)}$$
$$= \sum_{r=1}^{R} P(r) P_r^{(1)}(w_{t-n+1}) \ldots P_r^{(n)}(w_t). \tag{6}$$

Note that when $R = 1$, $\mathcal{T}$ degenerates to a unigram model. On the other extreme, when $\mathcal{T}$ is sufficiently high rank ($R = V^{n-1}$), it can represent any possible joint probability distribution over n words. Interpolating R between these extremes permits us to carefully control model complexity, so that it can be matched to the amount of training data available.

We construct the probability of a word sequence using the standard n-gram Markov assumption:

$$P(w_1, \ldots, w_T)$$
$$= \prod_{t=1}^{T} P(w_t | w_{t-n+1}^{t-1})$$
$$= \prod_{t=1}^{T} \frac{\sum_{r=1}^{R} P(r) P_r^{(1)}(w_{t-n+1}) \cdots P_r^{(n)}(w_t)}{\sum_{s=1}^{R} P(s) P_s^{(1)}(w_{t-n+1}) \cdots P_s^{(n-1)}(w_{t-1})} \tag{7}$$

where for notational convenience we assume that $w_{1-n+1}, \ldots, w_0$ are a designated sentence start token. Note that in (7), unlike traditional language mixture models, $P(w_t | w_{t-n+1}^{t-1})$ does not take the form of a sum of conditional distributions. By learning joint probabilities directly, we can capture higher-order multilinear behavior.

The connection between non-negative tensor factorization and latent variable models has been previously explored in the literature (e.g., in [7], [8]). Non-negative tensor factorization models have also been applied to other language processing applications, including subject-verb-object selectional preference induction [9] and learning semantic word similarity [10]. Without drawing the connection to low rank tensors, Lowd and Domingos [11] propose Naive Bayes models for estimating arbitrary probability distributions that can be seen as a generalization of (6).

B. Training

Our criterion for language model training is to maximize the log-likelihood of the n-grams appearing in the training data. Formally, we find a local solution to the problem:

$$\max_{\mathcal{T}\in\mathbb{P},\mathrm{rank}(\mathcal{T})\leq R} \log P_{\mathcal{T}}(D) \tag{8}$$

where $\mathbb{P}$ denotes the set of element-wise non-negative tensors whose entries sum to one, i.e., the set of tensors corresponding to valid joint probability distributions; $D = \{d_1, d_2, \ldots, d_K\}$ are the n-grams in the training data (obtained by sliding a window of size n over each sentence); and $P_{\mathcal{T}}$ is the probability distribution given by $\mathcal{T}$.[1] For traditional n-gram models, the maximum likelihood objective yields models that are highly overfit to the data; in particular, they are plagued with zero probability n-grams. The parameter tying implied by the low rank form greatly reduces the risk of introducing zero probabilities into the model; in practice, some additional smoothing is still required.

The low-rank language model can be interpreted as a mixture model, where each component is a joint distribution that decomposes into a product of position-dependent unigram models over words: $P_r(w_{t-n+1}, \ldots, w_t) = P_r^{(1)}(w_{t-n+1}) \cdots P_r^{(n)}(w_t)$. Using this interpretation, we propose an expectation-maximization (EM) approach to training our models, iterating:

1) Given model parameters, assign the responsibilities γ_{rk} of each component r to the k-th n-gram instance $d_k = (w_1^{(k)}, w_2^{(k)}, \ldots, w_n^{(k)})$:

$$\gamma_{rk} = \frac{P(d_k|r)P(r)}{P(d_k)} = \frac{P(r)P_r^{(1)}(w_1^{(k)}) \cdots P_r^{(n)}(w_n^{(k)})}{P(w_1^{(k)}, w_2^{(k)}, \ldots, w_n^{(k)}).} \tag{9}$$

2) Given responsibilities γ, re-estimate $F^{(i)}, \lambda$:

$$P_r^{(p)}(w) = \frac{\sum_{k=1}^{K} \gamma_{rk}\delta(w_p^{(k)} = w)}{\sum_{k=1}^{K} \gamma_{rk}} \tag{10}$$

$$P(r) = \frac{1}{K}\sum_{k=1}^{K} \gamma_{rk} \tag{11}$$

where δ is an indicator function. Iterations continue until perplexity on a held-out development set begins to increase.

The above training is only guaranteed to converge to a local optimum, which means that proper initialization can be important. A simple initialization is reasonably effective for bigrams: randomly assign each training sample to one of the R mixture components and estimate the component statistics similar to step 2. To avoid zeroes in the component models, a small count mass weighted by the global unigram probability is added to each distribution $P_r^{(p)}(w)$ in (10).

[1]By using overlapping n-grams, the samples are no longer independent, and the estimation is not strictly maximum likelihood of the original data. While no token is double counted in a distribution $P_r^{(i)}$, each will be counted in $P_r^{(i)}$ for multiple i. The implication is that the distribution is not consistent with respect to marginalization; e.g., the probability of the start symbol is position-dependent (a desirable property).

TABLE I
LANGUAGE MODEL EXPERIMENT DATA

Dataset	Size (words)
BN Train	3.2M
BC Train	99K
BC Dev	189K
BC Test	136K

V. EXPERIMENTS

Our expectation is that the LRLM will be good for applications with small training sets. The experiments here first evaluate the LRLM by training on a small set of conversational speech transcripts and then in a domain adaptation context, which is another common approach when there is data sparsity in the target domain. The adaptation strategy is the standard approach of static mixture modeling, specifically linearly interpolating a large general model trained on out-of-domain data with the small domain-specific model.

A. Experimental Setup

Our experiments use LDC English broadcast speech data,[2] with broadcast conversations (BC) or talkshows as the target domain. This in-domain data is divided into three sets: training, development and test. For the out-of-domain data we use a much larger set of broadcast news speech, which is more formal in style and less conversational. Table I summarizes the data sets.

We train several bigram low rank language models (LR2) on the in-domain (BC) data, tuning the rank (in the range of 25 to 300). Because the initialization is randomized, we train models for each rank ten times with different initializations and pick the one that gives the best performance on the development set. As baselines, we also train in-domain bigram (B2) and trigram (B3) standard language models with modified Kneser-Ney (mKN) smoothing. Our general trigram (G3), trained on BN, also uses mKN smoothing. Finally, each of the in-domain models is interpolated with the general model. We use the SRILM toolkit [12] to train the mKN models and to perform model interpolation. The vocabulary consists of the top 5 K words in the in-domain (BC) training set.

B. Results

The experimental results are presented in Table II. As expected, models using only the small in-domain training data have relatively high perplexities. Of the in-domain-only models, however, the LRLM gives the best perplexity, 3.6% lower than the best baseline. Notably, the LR bigram outperforms the mKN trigram. The LR trigram gave no further gain; extensions to address this are described later. The LRLM results are similar to mKN when training on the larger BN set.

Benefiting from a larger training set, the out-of-domain model alone is much better than the small in-domain models. Interpolating the general model with any of the in-domain models yields an approximately 15% reduction in perplexity over the general model alone, highlighting the importance of in-domain data. However, the different target-domain models are contributing complementary information: when the in-domain models are combined performance further improves. In particular, combining the baseline trigram and LRLM gives the largest relative reduction in perplexity.

[2]http://www.ldc.upenn.edu

TABLE II
IN-DOMAIN TEST SET PERPLEXITIES. B DENOTES IN-DOMAIN BASELINE MODEL, G DENOTES GENERAL MODEL, AND LR DENOTES IN-DOMAIN LOW RANK MODEL. EACH MODEL IS SUFFIXED BY ITS n-GRAM ORDER

Model	Perplexity
B2	166.7
B3	169.1
LR2	**162.9**
B2+LR2	154.5
G3	98.7
G3+B2	83.7
G3+B3	83.6
G3+LR2	83.6
G3+B2+LR2	83.1
G3+B3+LR2	**82.6**

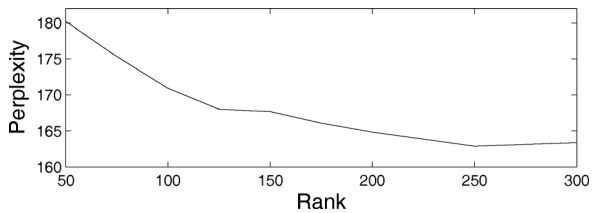

Fig. 2. Low rank language model perplexity by rank.

TABLE III
SAMPLES DRAWN RANDOMLY FROM LRLM MIXTURE COMPONENTS

r = 53 λ_r = 1.29e-02	r = 33 λ_r = 1.41e-02	r = 236 λ_r = 1.26e-02	r = 122 λ_r = 1.26e-03
he was	down and	should be	we civilians
he goes	over and	can be	defense security
he says	people and	will make	syrian armed
he faces	one and	would affect	iraqi prison

Fig. 2 reports LRLM perplexity for the LR2 model by rank R (the number of mixture components). For an in-domain bigram model, using approximately $R = 250$ mixture components is optimal, which corresponds to roughly 10% as many parameters as a full bigram joint probability matrix.

C. Discussion

Each component in the model specializes in some particular language behavior; in this light, the LRLM is a type of mixture of experts. To gain insight into what the different LRLM components capture, we investigated likely n-grams for different mixture components. We find that components tend to specialize in one of four ways: 1) modeling the distribution of words following a common word, 2) modeling the distribution of words preceding a common word, 3) modeling sets of n-grams where the words in both position are relatively inter-changeable with the other words in the same position, and 4) modeling semantic related n-grams. Table III illustrates these four types, showing sample n-grams randomly drawn from different components of a trained low rank model.

VI. CONCLUSIONS

Language model smoothing techniques can be viewed as operations on joint probability tensors over words; in this space, it is observed that one common thread between many smoothing methods is to reduce, either exactly or approximately, the tensor rank. This letter introduces a new approach to language modeling that more directly optimizes the low rank objective, using a factored low-rank tensor representation of the joint probability distribution. Using a novel approach to parameter-tying, the LRLM is better suited to modeling domains where training resources are scarce. On a genre-adaptation task, the LRLM obtains lower perplexity than the baseline (modified Kneser–Ney-smoothed) models.

The standard file formats used for interpolating different language models cannot compactly represent the low rank parameter structure of LRLMs. Thus, despite having relatively few free parameters, storing LRLMs in standard formats can result in prohibitively large files when the n-gram order or vocabulary size is large. To interpolate higher n-gram order LRLMs will require either development of a new format or implementation of interpolation within LRLM training itself.

In addition to implementation issues, our initial experiments did not obtain gains for trigrams as for bigrams. Possible improvements that may address this include alternative initialization methods to find better local optima (since training optimizes a nonconvex objective), exploration of smoothing in combination with regularization, and other low-rank parameterizations of the model (e.g., the Tucker decomposition [6]). For domain adaptation, there are many other approaches that could be leveraged [13], and the LRLM might be useful as the filtering LM used in selecting data from out-of-domain sources [14]. Finally, it would be possible to incorporate additional criteria into the LRLM training objective, e.g., minimizing distance to a reference distribution.

REFERENCES

[1] S. F. Chen and J. Goodman, "An empirical study of smoothing techniques for language modeling," *Comput. Speech Lang.*, vol. 13, no. 4, pp. 359–394, Oct. 1999.
[2] C. Chelba, T. Brants, W. Neveitt, and P. Xu, "Study on interaction between entropy pruning and Kneser–Ney smoothing," in *Proc. Interspeech*, 2010, pp. 2422–2425.
[3] C. J. Hillar and L.-H. Lim, "Most tensor problems are NP hard," in *Proc. CORR*, 2009.
[4] M. Signoretto, L. D. Lathauwer, and J. A. K. Suykens, Nuclear Norms for Tensors and Their Use for Convex Multilinear Estimation ESAT-SISTA, K. U. Leuven, Belgium, 2010, Tech. Rep. 10-186.
[5] R. Tomioka, K. Hayashi, and H. Kashima, "On the extension of trace norm to tensors," in *Proc. NIPS Workshop on Tensors, Kernels and Machine Learning*, Dec. 2010.
[6] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Rev.*, vol. 51, no. 3, pp. 455–500, Sep. 2009.
[7] A. Shashua and T. Hazan, "Non-negative tensor factorization with applications to statistics and computer vision," in *Proc. ICML*, New York, 2005, pp. 792–799.
[8] M. Shashanka, B. Raj, and P. Smaragdis, "Probabilistic latent variable models as nonnegative factorizations," *Comput. Intell. Neurosci.*, 2008.
[9] T. Van de cruys, "A non-negative tensor factorization model for selectional preference induction," *Nat. Lang. Eng.*, vol. 16, pp. 417–437.
[10] P. D. Turney, Empirical Evaluation of Four Tensor Decomposition Algorithms Institute for Information Technology, NRC Canada, 2007, Tech. Rep. ERB-1152.
[11] D. Lowd and P. Domingos, "Naive bayes models for probability estimation," in *Proc. ICML*, 2005, pp. 529–536.
[12] A. Stolcke, "SRILM – An extensible language modeling toolkit," in *Proc. ICSLP*, 2002, pp. 901–904.
[13] J. R. Bellegarda, "Statistical language model adaptation: Review and perspectives," *Speech Commun.*, vol. 42, pp. 93–108, 2004.
[14] I. Bulyko, M. Ostendorf, M. Siu, T. Ng, A. Stolcke, and O. Çetin, "Web resources for language modeling in conversational speech recognition," *ACM Trans. Speech Lang. Process.*, vol. 5, no. 1, pp. 1–25, Dec. 2007.