SPARSE NON-NEGATIVE MATRIX LANGUAGE MODELING FOR GEO-ANNOTATED QUERY SESSION DATA

Ciprian Chelba and Noam Shazeer

Google, Inc.,
1600 Amphitheatre Parkway
Mountain View, CA 94043, USA
Email: {ciprianchelba,noam}@google.com

ABSTRACT

The paper investigates the impact on query language modeling when using skip-grams within query as well as across queries in a given search session, in conjunction with the geo-annotation available for the query stream data. As modeling tool we use the recently proposed sparse non-negative matrix estimation technique, since it offers the same expressive power as the well-established maximum entropy approach in combining arbitrary context features.

Experiments on the `google.com` query stream show that using session-level and geo-location context we can expect reductions in perplexity of 34% relative over the Kneser-Ney N-gram baseline; when evaluating on the '"local" subset of the query stream, the relative reduction in PPL is 51%—more than a bit. Both sources of context information (geo-location, and previous queries in session) are about equally valuable in building a language model for the query stream.

Index Terms— language modeling, geo-location, query session, sparse non-negative matrix, voice search

1. INTRODUCTION

The `google.com` query stream is a rich data source for language modeling (LM) work. In addition to making available a very large amount of data, the query stream is annotated with geo-location information at various resolution levels, query order within a one day search session, as well as time stamp.

We have recently proposed a novel LM paradigm based on Sparse Non-negative Matrix (SNM) estimation [9], [9a]. When trained with n-gram features, the SNM-LMs were shown to perform almost as well as Kneser-Ney ones [7]. In addition, when pruned using a mutual information-based algorithm, the SNM n-gram LMs suffer less from aggressive pruning and significantly outperform entropy pruning for the well-established Katz [6] and interpolated Kneser-Ney models, as shown in [2]. The SNM-LM paradigm allows one to combine arbitrary features, offering the same expressive power as the well-established Maximum Entropy approach. It is thus well suited to building LMs that incorporate the rich features available with the query stream data.

A way of leveraging long distance context is to use skip-grams [8], [11]. Skip-grams are a generalization of regular n-grams where in addition to allowing adjacent word sequences, words are also allowed to be skipped, thus covering a longer context without being hampered as much by data sparsity. Previous work has revealed that training a model with skip-gram features is able to compete with neural network-based models [12]. SNM skip-grams have been shown to be as adept at modeling long distance dependencies as the RNN-LM approach, see [9]. In this work we investigate the impact on query language modeling when using within query skip-grams, as well as across queries in a given session, in conjunction with the geo-annotation available for the query stream data. We use SNM-LM as modeling tool.

In the remainder of this paper we describe the SNM-LM paradigm (Section 2), describe our approach to leverage session-level skip-gram and geo-annotation for language modeling of the `google.com` query stream (Section 3), evaluate it experimentally (Section 5) and discuss related work (Section 6). We end with conclusions and future work in Section 7.

1.1. Privacy Considerations

Before delving into the technical details, we wish to clarify the privacy aspects of our work with respect to handling user data.

All of the query data used for training and testing models is strictly anonymous; the queries bear no user-identifying information. The only data saved after training are vocabularies and n-gram counts.

2. SPARSE NON-NEGATIVE MATRIX LANGUAGE MODELING

In this section we describe our new paradigm without working out all the derivations. The interested reader can find these

in [9].

2.1. Model definition

In the Sparse Non-negative Matrix (SNM) paradigm, we represent the training data as a sequence of events $E = e_1, e_2, ...$ where each event $e \in E$ consists of a sparse non-negative feature vector $\mathbf{f}$ and a sparse non-negative target word vector $\mathbf{t}$. Both vectors are binary-valued, indicating the presence or absence of a feature or target word, respectively. The size of $\mathbf{f}$ depends on the total amount of features over all events, whereas the size of $\mathbf{t}$ corresponds to the size of the vocabulary $\mathcal{V}$. Hence, the training data consists of $|E||Pos(\mathbf{f})|(|\mathcal{V}|)$ training examples, where $Pos(\mathbf{f})$ denotes the number of positive elements in the vector $\mathbf{f}$. Of these, $|E||Pos(\mathbf{f})|$ are positive (presence of target word) and $|E||Pos(\mathbf{f})|(|\mathcal{V}| - 1)$ are negative (absence of target word),

A language model is represented by a non-negative matrix $\mathbf{M}$ that, when applied to a given feature vector $\mathbf{f}$, produces a dense prediction vector $\mathbf{y}$:

$$\mathbf{y} = \mathbf{M}\mathbf{f} \approx \mathbf{t} \tag{1}$$

Upon evaluation, we normalize $\mathbf{y}$ such that we end up with a conditional probability distribution $P_{\mathbf{M}}(\mathbf{t}|\mathbf{f})$ for a model $\mathbf{M}$. For each word $w \in \mathcal{V}$ that corresponds to index j in $\mathbf{t}$, and its history that corresponds to feature vector $\mathbf{f}$, the conditional probability $P_{\mathbf{M}}(t_j|\mathbf{f})$ then becomes:

$$\begin{aligned} P_{\mathbf{M}}(t_j|\mathbf{f}) &= \frac{y_j}{\sum_{u=1}^{|\mathcal{V}|} y_u} \\ &= \frac{\sum_{i \in Pos(\mathbf{f})} M_{ij}}{\sum_{i \in Pos(\mathbf{f})} \sum_{u=1}^{|\mathcal{V}|} M_{iu}} \end{aligned} \tag{2}$$

For convenience, we will write $P(t_j|\mathbf{f})$ instead of $P_{\mathbf{M}}(t_j|\mathbf{f})$ in the rest of the paper.

As required by the denominator in Eq. (2), this computation involves summing over all of the present features for the entire vocabulary. However, if we precompute the row sums $\sum_{u=1}^{|\mathcal{V}|} M_{iu}$ and store them together with the model, the evaluation can be done very efficiently in only $|Pos(\mathbf{f})|$ time. Note also that the row sum precomputation involves only few terms due to the sparsity of $\mathbf{M}$.

2.2. Adjustment function and metafeatures

We let the entries of $\mathbf{M}$ be a slightly modified version of the relative frequencies:

$$M_{ij} = e^{A(i,j)} \frac{C_{ij}}{C_{i*}} \tag{3}$$

where $\mathbf{C}$ is a feature-target count matrix, computed over the entire training corpus and $A(i, j)$ is a real-valued function, dubbed *adjustment function*. For each feature-target pair (f_i, t_j), the adjustment function computes a sum of weights $\theta_k(i, j)$ corresponding to k new features, called *metafeatures*:

$$A(i, j) = \sum_k \theta_k(i, j) \tag{4}$$

From the given input features, such as regular n-grams and skip-grams, we construct the metafeatures as conjunctions of any or all of the following elementary metafeatures:

- feature identity, e.g. `[the quick brown]`
- feature type, e.g. 4-gram
- feature count C_{i*}
- target identity, e.g. `fox`
- feature-target count C_{ij}

Note that the seemingly absent feature-target identity is represented by the conjunction of the feature identity and the target identity. Since the metafeatures may involve the feature count and feature-target count, in the rest of the paper we will write $A(i, j, C_{i*}, C_{ij})$ when necessary. This will become important in Section 2.5 where we discuss leave-one-out training.

Each elementary metafeature is joined with the others to form more complex metafeatures which in turn are joined with all the other elementary and complex metafeatures, ultimately ending up with all $2^5 - 1$ possible combinations of metafeatures.

2.3. Model estimation

Estimating a model $\mathbf{M}$ corresponds to finding optimal weights θ_k for all the metafeatures for all events in such a way that the average loss over all events between the target vector $\mathbf{t}$ and the prediction vector $\mathbf{y}$ is minimized, according to some loss function L.

In [9] we suggested a loss function based on the Poisson distribution: we consider each t_j in $\mathbf{t}$ to be Poisson distributed with parameter y_j. The conditional probability of $P_{Poisson}(\mathbf{t}|\mathbf{f})$ then is:

$$P_{Poisson}(\mathbf{t}|\mathbf{f}) = \prod_{j \in \mathbf{t}} \frac{y_j^{t_j} e^{-y_j}}{t_j!} \tag{5}$$

and the corresponding Poisson loss function is:

$$\begin{aligned} L_{Poisson}(\mathbf{y}, \mathbf{t}) &= -log(P_{Poisson}(\mathbf{t}|\mathbf{f})) \\ &= -\sum_{j \in \mathbf{t}} [t_j \log(y_j) - y_j - log(t_j!)] \\ &= \sum_{j \in \mathbf{t}} y_j - \sum_{j \in \mathbf{t}} t_j \log(y_j) \end{aligned} \tag{6}$$

where we dropped the last term, since t_j is binary-valued[1]. Although this choice is not obvious in the context of language modeling, it is well suited to gradient-based optimization and, as we will see, the experimental results are in fact excellent. Moreover, the Poisson loss also lends itself nicely for multiple target prediction which might be useful in e.g. subword modeling.

The adjustment function is learned by applying stochastic gradient descent on the loss function. That is, for each feature-target pair (f_i, t_j) in each event we need to update the weights of the metafeatures by calculating the gradient with respect to the adjustment function.

$$\frac{\partial(L_{Poisson}(\mathbf{Mf}, \mathbf{t}))}{\partial(A(i,j))} = f_i M_{ij}(1 - \frac{t_j}{y_j}) \tag{7}$$

For the complete derivation we refer to [9].

We then use the Adagrad [3] adaptive learning rate procedure to update the metafeature weights. Rather than using a single fixed learning rate, Adagrad uses a separate adaptive learning rate $\eta_{k,N}(i,j)$ for each weight $\theta_k(i,j)$ at the Nth occurrence of (f_i, t_j):

$$\eta_{k,N}(i,j) = \frac{\gamma}{\sqrt{\Delta_0 + \sum_{n=1}^{N} \partial_n(ij)^2}} \tag{8}$$

where γ is a constant scaling factor for all learning rates, Δ_0 is an initial accumulator constant and $\partial_n(ij)$ is a shorthand notation for the Nth gradient of the loss with respect to $A(i,j)$.

2.4. Optimization

If we were to apply the gradient in Eq. (7) to each (positive and negative) training example, it would be computationally too expensive, because even though the second term is zero for all the negative training examples, the first term needs to be computed for all $|E||Pos(\mathbf{f})||\mathcal{V}|$ training examples.

However, since the first term does not depend on y_j, we are able to distribute the updates for the negative examples over the positive ones by adding in gradients for a fraction of the events where $f_i = 1$, but $t_j = 0$. In particular, instead of adding the term $f_i M_{ij}$, we add $f_i t_j \frac{C_{i*}}{C_{ij}} M_{ij}$ which lets us update the gradient only on positive examples. This is based on the observation that, over the entire training set, it amounts to the same thing. For the complete derivation we refer to [9].

We note that this update is only strictly correct for batch training, and not for online training since M_{ij} changes after each update. Nonetheless, we found this to yield good results as well as seriously reducing the computational cost. The online gradient applied to each training example then becomes:

$$\frac{\partial(L_{Poisson}(\mathbf{Mf}, \mathbf{t}))}{\partial(A(i,j))} = f_i t_j \frac{C_{i*} - C_{ij}}{C_{ij}} M_{ij} + f_i t_j (1 - \frac{1}{y_j}) M_{ij} \tag{9}$$

[1]In fact, even in the general case where t_j can take any non-negative value, this term will disappear in the gradient, as it is independent of $\mathbf{M}$.

which is non-zero only for positive training examples, hence speeding up computation by a factor of $|\mathcal{V}|$.

2.5. Leave-one-out training

A model with a huge amount of parameters is prone to overfitting the training data. The preferred way to deal with this issue is to use held-out data to estimate the parameters. Unfortunately the aggregated gradients in Eq. (9) do not allow us to use additional data to train the adjustment function, since they tie the update computation to the relative frequencies $\frac{C_{i*}}{C_{ij}}$ in the training data. Instead, we have to resort to leave-one-out training to prevent the model from overfitting. We do this by excluding the event that generates the gradients from the counts used to compute those gradients. So, for each positive example (f_i, t_j) of each event $e = (\mathbf{f}, \mathbf{t})$, we compute the gradient, excluding 1 from C_{i*} and C_{ij}. For the gradients of the negative examples on the other hand we only exclude 1 from C_{i*}, because we did not observe t_j. In order to keep the aggregate computation of the gradients for the negative examples, we distribute them uniformly over all the positive examples with the same feature; each of the C_{ij} positive examples will then compute the gradient of $\frac{C_{i*} - C_{ij}}{C_{ij}}$ negative examples.

To summarize, when we do leave-one-out training we apply the following gradient update rule on all positive training examples:

$$\begin{aligned}
&\frac{\partial(L_{Poisson}(\mathbf{Mf}, \mathbf{t}))}{\partial(A(i,j))} \\
&= f_i t_j \frac{C_{i*} - C_{ij}}{C_{ij}} e^{A(i,j,C_{i*}-1,C_{ij})} \frac{C_{ij}}{C_{i*} - 1} \\
&+ f_i t_j (1 - \frac{1}{y'_j}) e^{A(i,j,C_{i*}-1,C_{ij}-1)} \frac{C_{ij} - 1}{C_{i*} - 1}
\end{aligned} \tag{10}$$

where y'_j is the product of leaving one out for all the relevant features:

$$\begin{aligned}
y'_j &= (\mathbf{M'f})_j \\
\mathbf{M}'_{ij} &= e^{A(i,j,C_{i*}-1,C_{ij}-1)} \frac{C_{ij} - 1}{C_{i*} - 1}
\end{aligned}$$

3. SKIP-GRAM LANGUAGE MODELING

In our approach, a skip-gram feature extracted from the context W_{k-1} is characterized by the tuple (r, s, a) where:

- r denotes the number of remote context words
- s denotes the number of skipped words
- a denotes the number of adjacent context words

relative to the target word w_k being predicted. For example, in the sentence `<S> The quick brown fox jumps`

Model	Training Set	Test Set Perplexity							
		all		all/local		all/geo		all/geo/local	
		abs	rel (%)	abs	rel (%)	abs	rel (%)	abs	rel (%)
Katz 5-gram	10B	91.1	-	95.8	-	88.9	-	94.3	-
+ DMA 5-gram	10B	85.9	6	73.5	23	80.2	10	57.3	39
Katz 5-gram	100B	79.1	13	84.6	12	77.2	12	83.9	11
+ DMA 5-gram	100B	73.7	19	64.1	33	68.2	23	49.9	48
Kneser-Ney 5-gram	10B	86.0	-	90.9	-	83.9	-	89.7	-
+ DMA 5-gram	10B	80.8	6	69.4	24	75.3	10	54.1	40
Kneser-Ney 5-gram	100B	76.3	11	82.8	9	74.6	11	82.3	8
+ DMA 5-gram	100B	70.9	18	62.6	31	65.6	22	48.6	46

Table 1. N-gram perplexity for Katz and Kneser-Ney models trained on 10B and 100B words, with and without geo-location information.

`over the lazy dog </S>` a $(1, 2, 3)$ skip-gram feature for the target word `dog` is:
`[brown skip-2 over the lazy]`

For performance reasons, it is recommended to limit s and to limit either $(r + a)$ or limit both r and s; not setting any limits will result in events containing a set of skip-gram features whose total representation size is quintic in the length of the sentence.

We configure the skip-gram feature extractor to produce all features $\mathbf{f}$, defined by the equivalence class $\Phi(W_{k-1})$, that meet constraints on the minimum and maximum values for the number of :

- context words used $r + a$;
- remote words r;
- adjacent words a;
- words skipped s.

We also allow the option of not including the exact value of s in the feature representation; this may help with smoothing by sharing counts for various skip features. Tied skip-gram features will look like:
`[curiousity skip-* the cat]`

In order to build a good probability estimate for the target word w_k in a context W_{k-1} we need a way of combining an arbitrary number of skip-gram features $\mathbf{f}_{k-1}$, which do not fall into a simple hierarchy like regular n-gram features. The following section describes a simple, yet novel approach for combining such predictors in a way that is computationally easy, scales up gracefully to large amounts of data and as it turns out is also very effective from a modeling point of view.

3.1. Extension to Query Sessions

For query sessions, we extend the above skip-gram feature extractor to allow skips across previous query boundaries. A skip-gram is now defined by the number of :

- context words used $r + a$;
- remote words r;
- adjacent words a;
- previous query boundaries skipped q;
- skipped words s counted in reverse order from the end of the landing query.

4. GEO-LOCATION N-GRAM LANGUAGE MODELING

A simple way of making use of geo-location information in an N-gram language model is to split the data according to geo-tagged partition of the query stream, train a geo-tagged N-gram language model for each and then interpolate the relevant components when predicting the words of a geo-tagged query, see [1].

The SNM modeling approach allows for a more elegant approach: N-gram features can be augmented with either POSTAL CODE or DMA geo-tag, and used along the regular N-grams features for predicting the next word in a query. Experiments reported in Section 5.3 show that this approach is as effective as the one described above.

5. EXPERIMENTS

5.1. Query Benchmark

Our experiments use an internal benchmark corpus generated from English mobile query sessions—non-overlapping 24 hour time spans. For about 60% of queries we also have available geo-location annotation at postal code (ZIP), and designated marketing area (DMA) resolution.

The benchmark contains two training sets:

- one hundred billion (100B) word set
- ten billion (10B) word set

counting sentence begining and end boundary markers; the 10B set is a subset of the 100B one, at the tail end in chronological order.

The test set consists of sessions containing a total of about 7.7 million queries. The training data is selected from months prior to the test data. Test queries are also annotated with a "local" bit, signaling the fact that the search results page for that query contains results that are of local interest: local points of interest, businesses, restaurants, etc.

Since we aim our experiments at voice-search, both training and test data was pre-processed as commonly done for speech recognition.

To evaluate the full impact of geo-location on the quality of our LMs, we evaluate on all four test subsets: all, all/local, all/geo, and all/geo/local. Only the results on the subsets of the test set with geo annotation fully evaluate the impact of geo-location on LM; when measuring PPL on the full test set we are mixing almost equally predictions that use geo tags, with predictions that do not, so it is not an accurate reflection of the potential that geo-location information holds for improved LM.

5.2. N-gram Experiments

We have built and evaluated both Katz and Interpolated Kneser-Ney 5-gram LMs as baselines for both the 10B and the 100B training sets, respectively. The vocabulary was chosen to contain the most frequent 1 million words (978565), with an out-of-vocabulary (OoV) rate of 0.6% and 0.4% on the all and all/local test sets, respectively; the OoV rate is not affected by restricting either subset to the queries that have detailed geo annotation.

A simple way to make use of the DMA geo-location information is to interpolate the LM built from the entire training data with one built from the data for a specific DMA.

The results are presented in Table 2.5. Increasing the amount of training data ten-fold does reduce the PPL of the model by about 10% relative. As noted before, the performance gap between Katz and Interpolated Kneser-Ney is shrinking as the amount of data increases. Adding geo-location to the LM is about as productive in reducing PPL as increasing the amount of data ten-fold; for the local subsets it is particularly beneficial, reducing the PPL by about 40% relative over the regular N-gram LM. The relative gain holds as more training data is used for the LM.

5.3. Geo-tagged and Session-skip N-gram Experiments Using SNM Estimation

Similar to the expressive power of maximum entropy models, SNM LM estimation allows us to integrate geo-location information at various resolution levels, along with skip N-grams, constructed by either limiting the skip to the current query, or skipping query boundaries within the current query session.

Our current SNM LM implementation does not yet scale to 100B words of training data, so we have only run experiments on the 10B training set. The results are presented in Table 5.3.

We highlight below what we consider the most interesting results:

- N-gram: the SNM 5-gram PPL of 89 is better than the Katz 5-gram baseline (91), and slightly worse than the KN 5-gram baseline (86)
- geo-location N-gram:
  - DMA yields about 6%/23% relative gain over SNM 5-gram on all/local test sets, just as it does for Katz and KN n-grams; when evaluating on subset with detailed geo tags (both DMA and POSTAL CODE are present, besides COUNTRY) the relative reduction in PPL is 11%/39%
  - POSTAL CODE: about same results as DMA
  - DMA together with POSTAL CODE yields about 8%/28% relative gain over SNM 5-gram on all/local test sets; when evaluating on subset with detailed geo tags (both DMA and POSTAL CODE are present, besides COUNTRY) the relative reduction in PPL is 13%/46%
- skip N-gram:
  - within query: 4% relative reduction in PPL over SNM 5-gram
  - within session: 26% rel reduction in PPL over SNM 5-gram
- geo-location and skip N-gram: geo-location (DMA and POSTAL CODE) combined with session-level skip N-gram features yield 33%/41% relative gain over SNM 5-gram on all/local test sets; when evaluating on the subset with detailed geo tags the relative reduction in PPL is 36%/53% on the all/geo and all/geo/local test sets, respectively.

6. RELATED WORK

SNM estimation is closely related to all N-gram LM smoothing techniques that rely on mixing relative frequencies at various orders. Unlike most of those, it combines the predictors at various orders without relying on hierarchical nesting of the contexts, setting it closer to the family of maximum entropy (ME) [11], or exponential models.

We are not the first ones to highlight the effectiveness of skip-grams at capturing dependencies across longer contexts, similar to RNN-LMs; previous such results were reported in [12]. Recently, [10] also showed that a backoff generalization using single skips yields significant perplexity reductions. We note that our SNM models are trained using both

Model	Test Set Perplexity							
	all		all/local		all/geo		all/geo/local	
	abs	rel (%)	abs	rel (%)	abs	rel (%)	abs	rel (%)
SNM 5-gram	89.4	-	94.9	-	87.3	-	93.6	-
SNM 5-gram + GEO								
GEO=DMA	83.5	6	73.1	23	77.4	11	56.8	39
GEO=POSTAL CODE	83.5	6	72.7	23	77.4	11	56.1	40
GEO=DMA + POSTAL CODE	82.5	8	68.6	28	75.7	13	50.2	46
SNM 5-gram + skip								
skip=within query	85.8	4						
skip=within session	63.6	29	73.3	23	62.3	29	73.8	21
SNM 5-gram + skip + GEO								
skip=within session, GEO=DMA + POSTAL CODE	59.6	33	55.9	41	55.5	36	43.7	53

Table 2. SNM LM perplexity using various feature extraction configurations on the 10B words training set.

single and longer skips and that our method of estimating the feature weights is, as far as we know, completely original.

The speed-ups to ME, and RNN LM training provided by hierarchically predicting words at the output layer [5], and subsampling [13] still require updates that are linear in the vocabulary size times the number of words in the training data, whereas the SNM updates in Eq. (10) for the much smaller adjustment function eliminate the dependency on the vocabulary size.

The computational advantages of SNM over both ME and RNN-LM estimation are probably its main strength, promising an approach that has the same flexibility in combining arbitrary features effectively and yet should scale to very large amounts of data as gracefully as N-gram LMs do.

The benefits of using geo-location information in building N-gram LMs for the query stream have been investigated in [1].

7. CONCLUSIONS AND FUTURE WORK

We have investigated the impact on query language modeling of using skip-grams within query as well as across queries in a given search session, in conjunction with the geo-annotation available for the query stream data. As modeling tool we use the recently proposed sparse non-negative matrix estimation technique, since it offers the same expressive power as the well-established Maximum Entropy approach in combining arbitrary context features.

Experiments on the `google.com` query stream show that using session-level and geo-location context we can expect reductions in perplexity of 34% relative over the Kneser-Ney N-gram baseline; when evaluating on the '"local" subset of the query stream, the relative reduction in PPL is 51%—more than a bit. Both sources of context information (geo-location, and previous queries in session) are about equally valuable in building a language model for the query stream.

As for future work, we would like to compare the ability of making use of the rich contextual information available in the query stream across all modeling approaches: SNM, ME, as well as RNN-LM.

8. REFERENCES

[1] Ciprian Chelba, Xuedong Huang and Keith Hall. "Geo-location for Voice Search Language Modeling," *Interspeech*, to appear, 2015.

[2] Joris Pelemans, Ciprian Chelba and Noam Shazeer. "Pruning Sparse Non-negative Matrix N-gram Language Models," *Interspeech*, to appear, 2015.

[3] John Duchi, Elad Hazan and Yoram Singer. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization," *Journal of Machine Learning Research*, 12, pp. 2121–2159, 2011.

[4] Joshua T. Goodman. "A Bit of Progress in Language Modeling, Extended Version," *Technical Report MSR-TR-2001-72*, 2001.

[5] Joshua T. Goodman. "Classes for Fast Maximum Entropy Training," *Proceedings of ICASSP*, pp. 561–564, 2001.

[6] Slava M. Katz. "Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer," *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-35, 3, pp. 400–401, 1987.

[7] Reinhard Kneser and Hermann Ney. "Improved Backing-Off for M-Gram Language Modeling," *Proceedings of ICASSP*, pp. 181–184, 1995.

[8] Hermann Ney, Ute Essen, and Reinhard Kneser. "On Structuring Probabilistic Dependences in Stochastic Language Modeling," *Computer Speech and Language*, 8, pp. 1–38, 1994.

[9] Noam Shazeer, Joris Pelemans and Ciprian Chelba. "Skip-gram Language Modeling Using Sparse Non-negative Matrix Probability Estimation," *CoRR*, abs/1412.1454, 2014. [Online]. Available: http://arxiv.org/abs/1412.1454.

[9a] Noam Shazeer, Joris Pelemans and Ciprian Chelba. "Sparse Non-negative Matrix Language Modeling For Skip-grams," *Interspeech*, to appear, 2015.

[10] Rene Pickhardt, Thomas Gottron, Martin Körner, Paul G. Wagner, Till Speicher, and Steffen Staab. "A Generalized Language Model as the Combination of Skipped n-grams and Modified Kneser-Ney Smoothing," *Proceedings of ACL*, pp. 1145–1154, 2014.

[11] Ronald Rosenfeld. "Adaptive Statistical Language Modeling: A Maximum Entropy Approach," *Ph.D. Thesis, Carnegie Mellon University*, 1994.

[12] Mittul Singh and Dietrich Klakow. "Comparing RNNs and Log-linear Interpolation of Improved Skip-model on Four Babel Languages: Cantonese, Pashto, Tagalog, Turkish," *Proceedings of ICASSP*, pp. 8416–8420, 2013.

[13] Puyang Xu, Asela Gunawardana, and Sanjeev Khudanpur. "Efficient Subsampling for Training Complex Language Models," *Proceedings of EMNLP*, pp. 1128–1136, 2011.