Show Some Love to Your n-grams: A Bit of Progress and Stronger n-gram Language Modeling Baselines

Ehsan Shareghi, Daniela Gerz, Ivan Vulić, Anna Korhonen
Language Technology Lab, DTAL, University of Cambridge
{es776, dsg40, iv250, alk23}@cam.ac.uk

Abstract

In recent years neural language models (LMs) have set state-of-the-art performance for several benchmarking datasets. While the reasons for their success and their computational demand are well-documented, a comparison between neural models and more recent developments in n-gram models is neglected. In this paper, we examine the recent progress in n-gram literature, running experiments on 50 languages covering all morphological language families. Experimental results illustrate that a simple extension of Modified Kneser-Ney outperforms an LSTM language model on 42 languages while a word-level Bayesian n-gram LM (Shareghi et al., 2017) outperforms the character-aware neural model (Kim et al., 2016) on average across all languages, and its extension which explicitly injects linguistic knowledge (Gerz et al., 2018a) on 8 languages. Further experiments on larger Europarl datasets for 3 languages indicate that neural architectures are able to outperform computationally much cheaper n-gram models: n-gram training is up to $15,000\times$ quicker. Our experiments illustrate that standalone n-gram models lend themselves as natural choices for resource-lean or morphologically rich languages, while the recent progress has significantly improved their accuracy.

1 Introduction

Statistical language models (LMs) are the pivot for several natural language processing tasks where a model trained on a text corpus is required to assign a probability to a given sequence $w_1 w_2 ... w_N$ (denoted by w_1^N). This probability indicates how likely is for w_1^N to belong to the corpus and is decomposed into conditional probabilities of words given their preceding contexts as $P(w_1^N) = \prod_{i=1}^{N} P(w_i|w_1^{i-1})$.

In n-gram LMs the unbounded conditional probabilities $P(w_i|w_1^{i-1})$ are approximated by imposing a finite-order Markov assumption, $P(w_i|w_1^{i-1}) \approx P(w_i|w_{i-n+1}^{i-1})$. Several smoothing techniques address the statistical sparsity issue for computing the conditional probabilities (Kneser and Ney, 1995; Chen and Goodman, 1999; Teh, 2006; Shareghi et al., 2016a), while others avoided the above approximation with unbounded hierarchical nonparametric Bayesian frameworks (Wood et al., 2011; Shareghi et al., 2017).

Alternatively, neural LMs compute $P(w_i|w_1^{i-1})$ via recurrent neural units which, in theory, are capable of encoding an unbounded context w_1^{i-1}. In recent years, neural LMs have become the prominent class of language modeling and have established state-of-the-art results on almost all sufficiently large benchmarks (Melis et al., 2018; Yang et al., 2018). While outperforming n-grams in terms of predictive accuracy, the computational shortcomings of neural LMs are well-documented: Training neural LMs is computationally expensive to the point that running experiments on large data ($\geq$ a few GiBs) is beyond the reach of academic research to this date (Chen et al., 2016; Patwary et al., 2018; Puri et al., 2018). [1] Similarly, querying is slower for neural LMs due to the required matrix-based operations, whereas most of the widely used n-gram LM toolkits rely on a few hash lookups and much cheaper scalar-based operations (Liu et al., 2018; Tang and Lin, 2018).

Nonetheless, it has been shown that the best predictive performance is still achieved by combining the two models via a basic interpolation or a mixture model (Jozefowicz et al., 2016; Neubig and Dyer, 2016): this indicates that the progress in n-gram LM should eventually be reflected in improving the

[1] For instance, n-gram LMs could be trained on 32GiB of data on a single CPU with ~32GiB of RAM in half a day (Shareghi et al., 2016b). A ballpark estimate for neural LMs, based on Puri et al. (2018), requires 26 Tesla V100 16GB GPUs to finish within the same amount of time while its financial cost is at least 100× higher.

state-of-the-art performance. Inspired by this, in this paper we shed new light on the most notable recent progress in n-gram statistical LMs which improves their predictive accuracy.

We demonstrate that under a recent massively multilingual experimental setup of Gerz et al. (2018a), more recent extensions of Kneser-Ney family of n-gram LMs (Shareghi et al., 2016a, 2017) are highly competitive with neural LMs. More specifically, we experiment on 50 languages from different morphological families, and illustrate that a word-level Bayesian n-gram LM (Shareghi et al., 2017) outperforms the character-level informed neural counterpart (Kim et al., 2016) on average, and its linguistically informed variant (Gerz et al., 2018a) on 8 languages. On larger Europarl datasets we find that n-gram models cannot reach the performance peaks of computationally much more expensive neural models, but a 2× decrease in perplexity comes at the cost of 15,000× longer training time. Our work reveals that recent n-gram LMs should be used as strong baselines, especially in resource-lean LM data and for morphologically rich languages.

Additionally, n-gram LMs offer a stringent way of dealing with Out-of-Vocabulary (OOVs) and rare words in a *full vocabulary* setting without relying on any pruning (Heafield, 2013). However, in neural LMs this remains an open question (Kawakami et al., 2017; Kim et al., 2016; Cotterell et al., 2018), while a common practice is pruning the training corpus and imposing *closed vocabulary* assumption (Mikolov et al., 2010) where rare words at training and unseen words at test are treated as an UNK token. We provide the mathematical underpinnings of n-gram models and highlight how this popular treatment works in favor of neural LMs (in comparative studies), and enforces n-gram LMs to perform much worse than their full potential.

2 n-gram Language Models: Smoothing

We now provide an overview of established smoothing techniques for n-gram LMs and their recent extensions. Smoothing is typically achieved by interpolation, where the probability of seeing a word w_i after a context w_{i-n+1}^{i-1} $P(w_i|w_{i-n+1}^{i-1},\Theta)$ is smoothed by its probability after a shorter context, w_{i-n+2}^{i-1}, and follows the following general form:

$$\beta(w_i|w_{i-n+1}^{i-1},\Theta)+\gamma(w_{i-n+1}^{i-1},\Theta)P(w_i|w_{i-n+2}^{i-1},\Theta). \quad (1)$$

The term $\beta(\cdot)$ represents the existing mass for the n-gram (e.g. via maximum likelihood estimation),

	$\beta(w_i\|w_{i-n+1}^{i-1},\Theta)$	Θ
KN	$\frac{\left[c(w_{i-n+1}^{i})-D_n\right]^+}{c(w_{i-n+1}^{i-1})}$	D_n
MKN	$\frac{\left[c(w_{i-n+1}^{i})-D_n^{c(w_{i-n+1}^{i})}\right]^+}{c(w_{i-n+1}^{i-1})}$	$D_n^{i\in\{1,2,3+\}}$
GKN	$\frac{\left[c(w_{i-n+1}^{i})-D_n^{c(w_{i-n+1}^{i})}\right]^+}{c(w_{i-n+1}^{i-1})}$	$D_n^{i\in\{1,\ldots,10+\}}$
BKN	$\frac{\left[c(w_{i-n+1}^{i})-D_{w_{i-n+1}}t_{w_{i-n+1}}^{w_i}\right]^+}{c(w_{i-n+1}^{i-1})+\theta_{w_{i-n+1}}}$	$D_{w_{i-n+1}}, \theta_{w_{i-n+1}}, t_{w_{i-n+1}}^{w_i}$

Table 1: Top-level interpolation for n-gram LMs smoothings and its parameters, and $[x]^+ \stackrel{\text{def}}{=} \max\{x, 0\}$.

γ is the weight used for redistributing the preserved mass (e.g. via discounting), and Θ are the parameters of the smoothing technique. The recursion stops at the unigram level where the conditioning context is empty. The recursion at lower levels relies on different quantities (e.g. pseudo-counts) but for brevity we focus on the top level of recursion and only the first term, $\beta(w_i|w_{i-n+1}^{i-1},\Theta)$, which suffices to highlight the key differences between smoothing techniques (see Table 1).

Kneser-Ney (KN). The key parameters of KN (Kneser and Ney, 1995) are the k-gram specific discounts D_k which control the amount of preserved and redistributed mass at kth level of recursion. While learning the discounts (on held-out data) is a possibility, the following estimation is shown to work well in practice:

$$D_k = 1 - 2\frac{n_2(k)}{n_1(k)}.\frac{n_1(k)}{n_1(k)+2n_2(k)}. \quad (2)$$

It captures the characteristics of different k-gram orders by looking at the number of unique k-grams which occurred once, $n_1(k)$, or twice, $n_2(k)$ in the training data, defined as follows:

$$n_j(k) = \begin{cases} |\{\alpha \text{ s.t. } |\alpha|=k, c(\alpha)=j\}|, & \text{if } k=n \\ \left|\{\alpha \text{ s.t. } |\alpha|=k, N_{1+}(\bullet\alpha)=j\}\right|, & \text{if } k<n \end{cases} \quad (3)$$

where $N_{1+}(\bullet\alpha) = |\{w : c(w\alpha) > 0\}|$, and is referred to as a form of pseudo-count, and $c(\alpha)$ denotes the frequency of sequence α. KN considers one discount value at each level of recursion and the discounts are bounded, $0 \le D_k < 1$.

Modified Kneser-Ney (MKN). Similarly, Modified Kneser-Ney (Chen and Goodman, 1999) is

defined with modifications applied to the discounting mechanism in order to make them sensitive to the *existing* mass j. The discounts are estimated as:

$$D_k^j = \begin{cases} 0, & \text{if } j = 0 \\ j-(j+1)\frac{n_{j+1}(k)}{n_j(k)}\frac{n_1(k)}{n_1(k)+2n_2(k)}, & \text{if } j < \mathcal{B} \\ \mathcal{B}-(\mathcal{B}+1)\frac{n_4(k)}{n_3(k)}\frac{n_1(k)}{n_1(k)+2n_2(k)}, & \text{if } j \geq \mathcal{B} \end{cases} \quad (4)$$

where $n_j(k)$ is defined in Eq. (3), and $\mathcal{B} = 3$. This leads to three discount parameters $\{D_k^1, D_k^2, D_k^{3+}\}$ for each recursion level, and allows for discount values to be as large as $c(w_{i-n+1}^i)$. MKN is widely accepted as the state-of-the-art n-gram LM and is very frequently confused with KN in the literature.

Generalized Modified Kneser-Ney (GKN). In conditions where statistical sparsity is more severe a more refined approach to model the distributions is necessary. Motivated by this, Shareghi et al. (2016a) provided the mathematical proof (based on leave-one-out log-likelihood) of the discount bounds used in MKN and proposed a natural extension of its discount binning to $\mathcal{B} = 10$. This was shown to be effective for further perplexity reduction in out-of-domain setting where OOV ratio is naturally high.

Bayesian Kneser-Ney (BKN). The Bayesian generalization of KN is the Hierarchical Pitman-Yor Process (HPYP) language model (Teh, 2006). This can be interpreted as a richer parameterization of KN and MKN, where the additional parameters (introduced shortly) allow the model to have more flexibility in capturing the desired distribution.

The Pitman-Yor Process (Pitman et al., 1997), $\text{PYP}(D, \theta, H)$, is a distribution defined over probability vectors (each draw from a PYP is a multinomial distribution) and has three parameters: a base probability vector H which is the expected value of a draw from a PYP, the concentration parameter $-D < \theta$ which controls the variation of draws from a PYP around H, and the discount parameter $0 \leq D < 1$ which allows the drawn vectors to capture the power-law behavior.

In the LM context, given a sequence of words w_{i-n+1}, a draw from a PYP is a multinomial distribution over the words following this sequence, denoted by $G_{w_{i-n+1}}$. This distribution can be captured by a vector of two counts $\{t_{w_{i-n+1}}^{w_i}, n_{w_{i-n+1}}^{w_i}\}^{w_i \in \sigma}$ which defines a partitioning arrangement of $G_{w_{i-n+1}}$, while different partitioning correspond to different multinomial draws from PYP. Here, $n_{i-n+1}^{w_i}$ is the total number of evidence [2] for word w_i after the context w_{i-n+1}, $t_{i-n+1}^{w_i}$ is the total number of partitions dedicated to w_i constrained by $0 \leq t_{i-n+1}^{w_i} \leq n_{i-n+1}^{w_i}$, and σ is the vocabulary.

The HPYP ties PYP distributions through their base and offers the statistical mean to smooth over infinitely long contexts (Wood et al., 2011). For instance, $\text{PYP}(D_u, \theta_u, \text{PYP}(D_{\pi(u)}, \theta_{\pi(u)}, H))$ is a two-level HPYP where a draw from a *child* distribution u, is smoothed by a draw from its *parent* $\pi(u)$. Here, $\pi(u)$ is u with its most earliest word dropped (e.g., $u = abc$, $\pi(u) = bc$).

KN can be seen as a special case for BKN, when the concentration is 0, and $\{t_{w_{i-n+1}}^{w_i}=1\}^{w_i \in \sigma}$. BKN can be considered as a richer parameterization of MKN: The product $D_{w_{i-n+1}} t_{w_{i-n+1}}^{w_i}$ (see Table 1) allows ≥ 1 discounting, simulating the discount range of MKN, while an additional parameter $\theta_{w_{i-n+1}}$ permits further adjustments of the distribution. BKN is shown to outperform MKN (Shareghi et al., 2017) but relies on expensive parameter sampling.

Out-of-Vocabulary (OOV). To complete the definitions we now explain how unseen words or contexts are handled during the computation without resorting to pruning or closed vocabulary setting. This treatment is the same for both non-Bayesian and Bayesian methods described in this paper. Let us consider the generic interpolation form of Eq. (1) and the maximum likelihood estimation of the $\beta(.)$ term (hence Θ is dropped) in the top level of the interpolation, $\beta(w_i|w_{i-n+1}^{i-1}) = \frac{c(w_{i-n+1}^i)}{c(w_{i-n+1}^{i-1})}$. Regardless of the level of interpolation and the paradigm used for computing the $\beta(.)$ term, $\beta(.)$ and $\gamma(.)$ always share the same denominator (normalizing factor).

An unseen word can appear as the target word w_i, which results in $\beta(.) = 0$ as $c(w_{i-n+1}^i) = 0$. It can also appear as a part of the prediction context w_{i-n+1}^{i-1} in which case both the $\beta(.)$ and $\gamma(.)$ terms will be undefined (and ignored) as the denominator $c(w_{i-n+1}^{i-1}) = 0$. This procedure is applied to all levels of interpolation without loss of generality, and as can be seen it only relies on the basic mathematical property of the involved computations rather than any other presumptions about data preprocessing or vocabulary. [3]

[2] This is equal to $c(w_{i-n+1}^i)$ at the top level of recursion, hence not mentioned as a part of Θ in Table 1.

[3] See Shareghi (2017) for a comprehensive explanation of the models covered in this section.

3 Experiments and Results

As our main large-scale experiment we use a typologically diverse set of 50 languages. These LM datasets cover many languages which are challenging in terms of data size, as well as the type-token ratio. In a less challenging setup, we experiment on 3 languages with larger training data from the Europarl (Koehn, 2005) to compare the two classes of LMs based on perplexity reduction and training time. For full data statistics, actual sampling of languages and data curation see Gerz et al. (2018a,b).

The common practice of setting a frequency threshold and mapping training data unigrams ($(k = 1)$-gram) to an UNK token degrades the performance of n-gram LMs by discarding a range of discount parameters: e.g., using threshold (< 3) results in $n_1(1), n_2(1) = 0$, both included in Eq. (2) and Eq. (4), increases the average perplexity score of 5-gram KN in our experiments by 11%. Motivated by its significance, we base our comparison on reported results by Gerz et al. (2018a): they deal with the task in the full vocabulary setting (Adams et al., 2017) with word-level predictions, and follow a relatively comparable treatment of unseen words with both n-gram and neural LM families (although not identical) at test time without enforcing any threshold over the training data.

The benchmarked neural LMs include three models with word-level predictions: a standard LSTM (Zaremba et al., 2014), a Char-CNN-LSTM (denoted as CNN) (Kim et al., 2016) which incorporates character-level information in the input, and the Attract-Preserve model (denoted as AP) (Gerz et al., 2018a) which injects further subword-level information. All benchmarked n-gram LMs are 5-grams, with the exception of BKN which is an ∞-gram model trained via 5 samples[4] following the recipe of Shareghi et al. (2017). GKN results are based on $\mathcal{B} = 5$, tuned on a development set.

Results and Discussion. The main results on the 50-languages benchmark are summarized in Table 2. The results for the more recent n-gram LMs indicate that these n-gram models are highly competitive with neural LMs in this challenging resource-lean setup.[5] For instance, for 26/50 languages all n-gram models outperform a regular LSTM, while GKN and BKN extend the lead to 42/50

[4]Marginal improvements achieved with more sampling.

[5]The size of each dataset is $\approx$ 40K sentences, which is at the level of the standard Penn Treebank dataset often used for LM evaluation in English (Marcus et al., 1993).

		n-gram LMs pplx				neural LMs pplx		
Lang	OOV %	KN	MKN	GKN	BKN	LSTM	CNN	AP
♣ am	15.2	1289	1252	1101	941†	1535	981	817
♣ ar	11.1	2241	2156	1871	1746	2587	1659	1604
♡ bg	9.9	636	610	524	470	651	415	409
♡ ca	5.2	369‡	358‡	307	280	318	241	238
♡ cs	11	1724	1658	1433	1331	2200	1252	1131
♡ da	9.6	690	668	582	526	710	466	442
♡ de	10.6	964‡	930‡	814	721	903	602	551
♡ el	8.2	627‡	607‡	526	499	538	405	389
♡ en	6.1	553‡	533‡	454	391	494	371	349
♡ es	5.9	430‡	415‡	356	319	366	275	270
♠ et	12.9	1655	1609	1422†	1223†	2564	1478	1388
♠ eu	9.1	571‡	560‡	496	456	533	347	309
♡ fa	4.7	362‡	355‡	308‡	283‡	263	208	205
♠ fi	17.6	2709	2611	2318	2143†	4263	2236	1858
♡ fr	6.1	361‡	350‡	298‡	268	294	231	220
♣ he	8.9	1870	1797	1531	1373§	2189	1519	1375
♡ hi	7.1	488‡	473‡	408	378	426	326	299
♡ hr	10.6	1345	1294	1124	972†	1665	1014	906
♠ hu	12	1188	1151	1011	877†	1595	929	819
◇ id	6	469‡	454‡	388‡	358	359	286	263
♡ it	6	582‡	567‡	489	465	493	349	350
♠ ja	4.7	174‡	169‡	142	129†	156	136	125
♠ jv	12.2	1462‡	1387	1217	1138†	1443	1158	1003
♠ ka	11.7	1422	1370	1198	1115	1827	1097	939
◇ km	4.8	608	586	501§	451§	637	522	535
♠ kn	17.7	2378	2315	2051§	1914§	5310	2558	2265
♠ ko	19.3	5332	5146	4492†	4019†	10063	4778	3821
♡ lt	11	1187	1155	1001	891	1415	854	827
♡ lv	12.7	1518	1452	1255	1144	1967	1129	969
♠ mng	10.8	1441	1392	1215	1055§	1716	1165	1091
◇ ms	7.2	807‡	776‡	659	572	725	525	513
◇ my	2	216‡	209	178§	168§	212	182	180
◇ nan	3.8	63‡	61‡	52‡	45‡	43	39	38
♡ nl	5.6	407‡	397‡	347‡	305	340	267	248
♡ no	9	551‡	534‡	465	435	513	379	346
♡ pl	12	1810	1741	1514	1363†	2641	1491	1328
♡ pt	5	351‡	342‡	290‡	261	272	214	202
♡ ro	7	395‡	384‡	330	290	359	256	247
♡ ru	10.1	1160	1128	977	906	1309	812	715
♡ sk	13.1	1633	1560	1352	1234†	2062	1275	1151
♡ sl	10.9	1160	1114	969	856	1308	776	733
♡ sr	9.3	812	790	683	637	961	582	547
♡ sv	10	873‡	843‡	734	634	832	583	543
♠ ta	18.2	3469†	3342†	2920†	2635§	6234	3496	2768
◇ th	3.3	238	233	199§	181§	241	206	199
◇ tl	7.7	393‡	379‡	321‡	293	298	219	211
♠ tr	12.3	1784	1724	1497	1416	2267	1350	1290
♡ uk	12.8	1707	1639	1418	1338	1893	1283	1091
◇ vi	2.7	202‡	197‡	170	145§	190	158	165
◇ zh	3.8	1110‡	1064‡	899‡	777†	826	797	762
◇ avg	4.6	456‡	440‡	374	332	392	326	318
♡ avg	8.8	873	842	729	661	969	618	566
♠ avg	13.2	1965	1898	1665†	1510†	3164	1727	1473
♣ avg	11.7	1800	1735	1501	1354†	2104	1386	1265
+ avg	9.3	1116	1077	936	847†	1460	878	781

Table 2: Data Statistics and Perplexity scores. OOV denotes the percentage of unseen words at test time. For detailed data stats see (Gerz et al., 2018a). Suit symbols denote morphological types: ◇ Isolating, ♡ Fusional, ♠ Agglutinative, ♣ Introflexive. Color codes denote comparative performance: ‡ is outperformed by LSTM, † outperforms CNN, § outperforms AP.

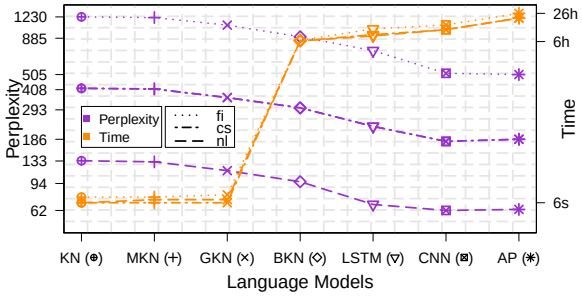

Figure 1: Time-Perplexity for *n*-gram and neural LMs across 3 languages fi, cs, nl. Timings done on a single core of AMD Ryzen 1900X for *n*-grams, and on a Nvidia TITAN X Pascal GPU for neural models.

and 48/50 languages, respectively. For certain morphologically rich languages (e.g., Tamil, Mongolian, Hebrew), the *n*-gram LMs are able to outscore even character-aware neural LMs. On average we observe *n*-grams succeed, especially the BKN model, for introflexive and agglutinative languages which are known to have productive morphological systems: as reported in Table 2, they have the higher OOV ratio compared to the other language families. Overall, the best performing *n*-gram model, BKN, outperforms both the LSTM (42% reduction in perplexity) and CNN models (3% reduction in perplexity), while falling behind AP by 8%.

These results highlight that *n*-gram models can serve as strong baselines for such morphologically rich languages with high OOV rates and type-to-token ratios, which are especially problematic in scarce data setups. Additionally, they suggest that more sophisticated *n*-gram variants such as GKN or BKN should be used to provide adequate comparison points with *n*-gram LMs than the commonly used KN or MKN.

As expected, experiments on 10× larger Europarl datasets for 3 languages show that neural models outperform *n*-gram models in less challenging data-intensive scenarios. However, training on large datasets comes at the expense of training efficiency for neural models: e.g., according to Figure 1 training non-Bayesian *n*-grams is around 15,000× quicker than training neural models. We leave a full-fledged investigation on the relation of training corpus size and efficacy of *n*-gram vs. neural LM training for future work. In addition, motivated by these preliminary insights, we advocate investing further efforts in future work into coupling the ideas behind *n*-gram and neural LMs towards improved language modeling.

4 Conclusion

We provided an overview of previous work and very recent progress in *n*-gram LMs. The recent developments, when tested on a challenging set of 50 languages, demonstrated superior or highly competitive performance compared with neural LMs, while being substantially cheaper to train. We also shed light on a common issue in the experimental setups, concerning OOV or rare words handling, when comparing *n*-grams and neural LMs.

While being non-trivial, investigating any correlation between cheap-to-compute heuristics (e.g., basic data statistics) and the choice of the most suitable model for a given dataset is worth exploring. Also, motivated by our findings, we will work on utilizing continuous space representations as side information in sampling the parameters of BKN, i.e. similar to Zhao et al. (2018), which potentially can reduce the gap between BKN and neural models.

Acknowledgments

This work is supported by the ERC Consolidator Grant LEXICAL (648909). The authors would like to thank the anonymous reviewers for their helpful suggestions. The first author would like to also thank the members of the Language Technology Lab for their comments on the presentation of this work.

References

Oliver Adams, Adam Makarucha, Graham Neubig, Steven Bird, and Trevor Cohn. 2017. Cross-lingual word embeddings for low-resource language modeling. In *EACL*, pages 937–947.

Stanley F. Chen and Joshua Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–393.

Wenlin Chen, David Grangier, and Michael Auli. 2016. Strategies for training large vocabulary neural language models. In *ACL*, pages 1975–1985.

Ryan Cotterell, Sebastian J. Mielke, Jason Eisner, and Brian Roark. 2018. Are all languages equally hard to language-model? In *NAACL*, pages 536–541.

Daniela Gerz, Ivan Vulić, Edoardo Ponti, Jason Naradowsky, Roi Reichart, and Anna Korhonen. 2018a. Language modeling for morphologically rich languages: Character-aware modeling for word-level prediction. *TACL*, 6:451–465.

Daniela Gerz, Ivan Vulić, Edoardo Maria Ponti, Roi Reichart, and Anna Korhonen. 2018b. On the relation between linguistic typology and (limitations of) multilingual language modeling. In *EMNLP*, pages 316–327.

Kenneth Heafield. 2013. *Efficient Language Modeling Algorithms with Applications to Statistical Machine Translation*. Ph.D. thesis, Carnegie Mellon University.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

Kazuya Kawakami, Chris Dyer, and Phil Blunsom. 2017. Learning to create and reuse words in open-vocabulary neural language modeling. In *ACL*, pages 1492–1502.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *AAAI*, pages 2741–2749.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *ICASSP*, pages 181–184.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT Summit*.

Xuan Liu, Di Cao, and Kai Yu. 2018. Binarized LSTM language model. In *NAACL*, pages 2113–2121.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Gabor Melis, Chris Dyer, and Phil Blunsom. 2018. On the state of the art of evaluation in neural language models. In *ICLR*.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*, pages 1045–1048.

Graham Neubig and Chris Dyer. 2016. Generalizing and hybridizing count-based and neural language models. In *EMNLP*, pages 1163–1172.

Mostofa Patwary, Milind Chabbi, Heewoo Jun, Jiaji Huang, Gregory Diamos, and Kenneth Church. 2018. Language modeling at scale. *arXiv preprint arXiv:1810.10045*.

Jim Pitman, Marc Yor, et al. 1997. The two-parameter poisson-dirichlet distribution derived from a stable subordinator. *The Annals of Probability*, 25(2):855–900.

Raul Puri, Robert Kirby, Nikolai Yakovenko, and Bryan Catanzaro. 2018. Large scale language modeling: Converging on 40GB of text in four hours. *arXiv preprint arXiv:1808.01371*.

Ehsan Shareghi. 2017. *Scalable Non-Markovian Sequential Modelling for Natural Language Processing*. Ph.D. thesis, Monash University.

Ehsan Shareghi, Trevor Cohn, and Gholamreza Haffari. 2016a. Richer interpolative smoothing based on modified Kneser-Ney language modeling. In *EMNLP*, pages 944–949.

Ehsan Shareghi, Gholamreza Haffari, and Trevor Cohn. 2017. Compressed nonparametric language modelling. In *IJCAI*, pages 2701–2707.

Ehsan Shareghi, Matthias Petri, Gholamreza Haffari, and Trevor Cohn. 2016b. Fast, small and exact: Infinite-order language modelling with compressed suffix trees. *TACL*, pages 477–490.

Raphael Tang and Jimmy Lin. 2018. Progress and tradeoffs in neural language models. *arXiv preprint arXiv:1811.00942*.

Yee Whye Teh. 2006. A Bayesian interpretation of interpolated Kneser-Ney. Technical report, NUS School of Computing.

Frank Wood, Jan Gasthaus, Cédric Archambeau, Lancelot James, and Yee Whye Teh. 2011. The sequence memoizer. *Communications of the ACM*, 54(2):91–98.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. 2018. Breaking the softmax bottleneck: A high-rank RNN language model. In *ICLR*.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

He Zhao, Lan Du, Wray L. Buntine, and Mingyuan Zhou. 2018. Inter and intra topic structure learning with word embeddings. In *ICML*, pages 5887–5896.