Towards Competitive N-gram Smoothing

Moein Falahatgar[1] moein@ucsd.edu
Mesrob Ohannessian[2] mesrob@uic.edu
Alon Orlitsky[1] alon@ucsd.edu
Venkatadheeraj Pichapati[1] dheerajpv7@ucsd.edu
[1] UC San Diego
[2] University of Illinois at Chicago

Abstract

N-gram models remain a fundamental component of language modeling. In data-scarce regimes, they are a strong alternative to neural models. Even when not used as-is, recent work shows they can regularize neural models. Despite this success, the effectiveness of one of the best N-gram smoothing methods, the one suggested by Kneser and Ney (1995), is not fully understood. In the hopes of explaining this performance, we study it through the lens of competitive distribution estimation: the ability to perform as well as an oracle aware of further structure in the data. We first establish basic competitive properties of Kneser–Ney smoothing. We then investigate the nature of its backoff mechanism and show that it emerges from first principles, rather than being an assumption of the model. We do this by generalizing the Good–Turing estimator to the contextual setting. This exploration leads us to a powerful generalization of Kneser–Ney, which we conjecture to have even stronger competitive properties. Empirically, it significantly improves performance on language modeling, even matching feed-forward neural models. To show that the mechanisms at play are not restricted to language modeling, we demonstrate similar gains on the task of predicting attack types in the Global Terrorism Database.

1 Introduction

Statistical N-gram language models, that strive to predict the N^{th} word based on the preceding $N-1$ words, have a long and rich history, and it is hard to give the literature justice in such a short space. The classical works are covered in the comprehensive survey (Chen and Goodman, 1999),which empirically studied different smoothing techniques. Smoothing is critical to learning these models, since so much of the N-gram space remains unobserved. For a long time, the most successful smoothing technique was the one proposed by Kneser and Ney (1995). This led to several efforts to explain its properties, mainly its use of backoff: reverting to a simpler model when data is scarce. Some of the best forays in this direction were the Bayesian perspective described by the hierarchical Pitman-Yor language models in (Teh, 2006) and, more pertinent to this paper, the more recent developments exploring rank-reduction properties (Hutchinson et al., 2015; Parikh et al., 2013; Falahatgar et al., 2016). Despite these, there was no complete understanding of the joint mechanisms of smoothing and backoff in Kneser–Ney.

Perhaps this is due to the surge of neural networks and in particular recurrent neural language models, which led to a significant jump in performance (Mikolov et al., 2010). Neural language models have since continued to improve their results (Merity et al., 2017; Yang et al., 2017; Gong et al., 2018; Takase et al., 2018; Dai et al., 2019). Interestingly, N-gram techniques are still relevant as they usually run much faster, and, can be used in conjunction with neural models to improve performance even further (Xie et al., 2017). Moreover, for low-resource languages, non-neural methods or hybrid/ensembled methods are known to achieve the best performance (Gandhe et al., 2014).

Motivated by this continued relevance, and curious about its further potential, this paper offers a first theoretical foray into N-gram models, and in particular into the principles behind the practice of backoff. As a result of this exploration and as concrete evidence of its promise, we report on a powerful generalization of Kneser–Ney backoff, which is empirically able to compete with neural models, albeit those limited to feed-forward architectures. It is worth mentioning that the smoothing aspect of N-gram models, understood

Proceedings of the 23[rd]International Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108.

primarily as high-dimensional categorical distribution estimation, has received attention. Part of the novelty of our perspective is to study backoff through the very same lens, namely that of competitive distribution estimation. This notion was expressed most clearly by Orlitsky and Suresh (2015), where it was used to strongly justify the Good–Turing estimator (Good, 1953), which is intimately related to Kneser–Ney.

Our contributions are as follows:

- We study this problem as a *contextual distribution estimation* problem. Apart from the fact that this means we aim to learn conditional distributions, the objective function and notions of competitivity have to both be carefully set. We do this in Section 3 and show that competitivity is possible in the contextual setting, and give some evidence for the advantage that Kneser–Ney has.
- We generalize the Good–Turing estimator to the contextual setting, in Section 4. The idealized expression of this estimator cannot be used directly and needs to be smoothed, just like in the non-contextual setting. We show that with the proper smoothing, contextual Good–Turing recovers the Kneser–Ney estimator, when the tails of the distributions are power laws and are aligned.
- The idealized Good–Turing formula is much more powerful than the special case of Kneser–Ney. We conjecture that it could potentially offer competitivity versus oracles that are aware of intricate relationships between distributions in various contexts. We illustrate this potential by giving a strict generalization of Kneser–Ney backoff, which we call *Partial Low-Rank*, since it applies the rank structure only to the rare part of the data. Kneser–Ney corresponds to the rank-1 special case.
- In Section 6, we show that Partial Low-Rank uniformly improves on Kneser–Ney on various benchmarks. Furthermore, a nested trigram-level implementation of this approach meets and slightly exceeds the performance of the feed-forward neural models on the Penn Tree Bank data set. This advantage is only enhanced by considering that it can be trained with a fraction of the time and space resources required for the neural model.

We start with some preliminaries in Section 2.

2 Preliminaries

We describe the problem generally. Let the context space be $\mathcal{X}$ and the prediction space be $\mathcal{Y}$. When finite, identify these spaces with $\mathcal{X} = [K]$ and $\mathcal{Y} = [k]$ respectively. Data is modeled as n context/prediction pairs $(X_t, Y_t)_{t=1,\cdots,n}$. How is this data generated? Various scenarios may be considered. In modeling sequence data, as in the case of language modeling, the ideal context is usually the whole history. Namely, given an infinite history $X_t := (\cdots, Y_{t-2}, Y_{t-1})$, there is a conditional probability of observing the next word Y_t. When the history is truncated to $N-1$ words, this is called an N-gram model. Other history-to-context mappings $X_t := f(\cdots, Y_{t-2}, Y_{t-1})$ may also be considered, such as skip-grams or word embeddings (Mikolov et al., 2013a,b). If the data consists of just a single long sequence, such as a text $Y_1, Y_2, Y_3, \ldots, Y_n$ of n words, the context/prediction pairs that result from partitioning the text are correlated.

We simplify this by assuming that X_t are independently and identically drawn from a distribution π over $\mathcal{X}$. In this case, independently for each context X_t, we take Y_t to be drawn from the conditional distribution $P_{ij} := \mathbb{P}(Y = j | X = i)$. The matrix $C_{i,j} = \sum_t \mathbf{1}\{X_t = i, Y_t = j\}$ then summarizes the data.

Our goal can now be concisely stated as: given $(X_t, Y_t)_{t=1,\cdots,n}$ or $(C_{i,j})_{i\in\mathcal{X}, j\in\mathcal{Y}}$, estimate $(P_{ij})_{i\in\mathcal{X}, j\in\mathcal{Y}}$. We judge the performance of an *estimator* Q that maps data to contextual probabilities, according to a suitably defined statistical risk. The primary goal of contextual probability estimation is to make accurate predictions, requiring the estimated conditional probability to be close to its true value on new data.

We consider the underlying risk of an estimator as being the *KL-risk*, defined as the averaged per-context Kullback-Leibler divergence

$$D_\pi(P\|Q) := \sum_i \pi_i \sum_j P_{ij} \ln \frac{P_{ij}}{Q_{ij}}, \tag{1}$$

which captures relative closeness of estimated and true probabilities, on average. It is the risk associated with the *log*-loss and, up to the entropy, is the population cross-entropy of Q. Since Q is random, guarantees are often given in terms of the expected KL-risk $r_n(\pi, P, Q) = \mathbb{E}[D_\pi(P\|Q)]$.

Ideally, we would like to have the best Q possible, an optimal one. In the non-contextual setting, when $K = 1$, one measure of optimality is worst-case risk with respect to a class $\mathcal{P}$, $r_n(Q, \mathcal{P}) = \max_{P\in\mathcal{P}} r_n(\pi, P, Q)$. The best possible such risk is known as the minimax risk of the class, $r_n(\mathcal{P}) = \min_Q r_n(Q, \mathcal{P})$. A minimax optimal Q (either exactly or in rate) is desirable but pessimistic: minimax optimality does not capture the possibility of the truth being in a smaller class. The competitive loss with respect to a family $\mathcal{F}$, which contains many such (some small, some big) classes, is a more optimistic notion. It is defined as $\epsilon(Q, \mathcal{F}) = \max_{\mathcal{P}\in\mathcal{F}}[r_n(Q, \mathcal{P}) - r_n(\mathcal{P})]$. This is related to the notion of adaptivity in statistics.

Think of a family as an *oracle* that can determine exactly which $\mathcal{P}$ in $\mathcal{F}$ we have, and does the best for it. When an estimator has small ϵ it means it manages to do as well as this oracle itself. To see the reason for optimism, say $\mathcal{P}$ is nice, with a small $r_n(\mathcal{P})$. The estimator will achieve this small risk, even if $\mathcal{F}$ contains such large classes for which $r_n(\mathcal{P})$ is enormous. If an estimator has ϵ that is of the same order as the minimax risk, we call it *competitive*. It effectively discovers the true $\mathcal{P}$. We currently have a nascent theory for non-contextual estimators that are competitive with respect to rich families. These include the works that show that the Good–Turing estimator combined with the empirical estimator has dimension-free competitivity, (Orlitsky and Suresh, 2015), and that the absolute discounting estimator competes with oracles aware of the effective alphabet size of the distribution, (Falahatgar et al., 2017). A similar notion can also be found in (Valiant and Valiant, 2015).

In the bigram setting, $K = k$, Kneser–Ney backoff can be described as follows. For every context / row of C, perform absolute discounting, defined, for $\alpha < 1$ as:

$$Q_{ij} = (C_{ij} - \alpha)/n_i, \qquad \text{when } C_{ij} \geq 1,$$

where $n_i = \sum_j C_{ij}$. Note that the α subtractions discount a total probability mass of α/n_i times the number of distinct predictions that appear in context i. This constitutes the estimate of the *missing mass*, the total probability of unseen predictions, i.e. j with $C_{ij} = 0$. One has then to figure out how this total mass is spread out to specific predictions. The simplest approach is row-wise absolute discounting: spread this estimated missing mass evenly over unseen predictions. It is roughly equivalent to row-wise Good–Turing estimation (Falahatgar et al., 2017). Instead of this, Kneser–Ney *backs-off* to an alternate distribution: it spreads the missing mass proportionally to backoff counts $b_j = \sum_i \mathbf{1}\{C_{ij} > 0\}$. The variant proposed by Chen-Goodman (Chen and Goodman, 1999) performs a further absolute discounting on b, and spreads proportionally to $(b_j - \alpha)/\sum_{j'} b_{j'}$. This generates a secondary missing mass within the b, which is itself evenly spread over the b_j that are zero. Despite being such a simple estimator, Kneser–Ney backoff, and especially the Chen-Goodman variant held state-of-the-art performance for more than a decade.

3 Theoretical Insights

We now give a tentative theoretical exploration of the advantage of backoff through the lens of competitive distribution estimation. Let us first define the contextual competitive loss of an estimator Q, with respect to a family $\mathcal{F}$ of classes. In general $\mathcal{F}$ contains classes $\mathcal{C}$, which are sets containing pairs (π, P). To simplify, we take π to be arbitrary, or equivalently each $\mathcal{C}$ is of the form $\Delta_K \times \mathcal{P}$ where $\mathcal{P}$ is a class of Ps only. The competitive loss of an estimator Q is then:

$$\epsilon_n(Q, \mathcal{F}) := \max_{\mathcal{C}\in\mathcal{F}} \max_{(\pi,P)\in\mathcal{C}} \left[r_n(\pi, P, Q) - r_n(\mathcal{P})\right], \tag{2}$$

$$= \max_{\Delta_K\times\mathcal{P}\in\mathcal{F}} \max_{\pi\in\Delta_K} \max_{P\in\mathcal{P}} \left[r_n(\pi, P, Q) - r_n(\mathcal{P})\right], \tag{3}$$

where the risk of the estimator Q is its expected KL-risk,

$$r_n(\pi, P, Q) = \mathbb{E}_{(x^n,y^n)\sim\pi P}\big[\textstyle\sum_i \pi_i \sum_j P_{ij} \log \frac{P_{ij}}{Q_{ij}}\big], \tag{4}$$

and the minimax risk of the class $\mathcal{C} = \Delta_K \times \mathcal{P}$ achieved by an optimal estimator $Q^{\mathcal{C}}$,

$$r_n(\mathcal{P}) := \min_Q \max_{\pi\in\Delta_K} \max_{P\in\mathcal{P}} r_n(\pi, P, Q). \tag{5}$$

The choice of family $\mathcal{F}$ is again equivalent to competing with an oracle/genie that can determine the true (π, P) up to a class $\mathcal{C}$ which it belongs to. We are in particular considering oracles that are uninformed about π, but know that P belongs to some class $\mathcal{P}$ allowed by $\mathcal{F}$.

Basic competitivity Consider an oracle $\mathcal{F}_1$ that knows each row of P up to permutation. Is it possible to compete with it? Intuitively, one ought to be able to, by reducing to the non-contextual competitive estimator in each context. There are some subtle points to consider, however. One is the fact that the number of samples that each context receives is random. More importantly, the number K of contexts plays a role in how competitive we can be.

Let Q^{GT} be the per-context Good–Turing estimator. This analysis also describes absolute-discounting applied to data from each context separately. Out of n total samples, let n_i denote those that fall in context i. In the non-contextual case, the Good–Turing estimator with n samples has a competitive loss of $\mathcal{O}(\min\{\frac{1}{\sqrt{n}}, \frac{k}{n}\})$. Note that it is dimensionless in the high-dimensional regime. So we intuitively expect the same to hold per context. We formally extend this to the overall contextual case.

Theorem 1 *We have*

$$\epsilon_n(Q^{\mathsf{GT}}, \mathcal{F}_1) \leq \mathcal{O}\left(\min\left\{1, \sqrt{\frac{K}{n}}, \frac{K\cdot k}{n}\right\}\right).$$

This implies three distinct regimes:

$$\epsilon_n(Q^{\mathsf{GT}}, \mathcal{F}_1) \leq \begin{cases} \mathcal{O}(\frac{K\cdot k}{n}) & n > K\cdot k^2 \\ \mathcal{O}(\sqrt{\frac{K}{n}}) & K < n < K\cdot k^2 \\ \mathcal{O}(1) & n < K \end{cases}$$

The proof is in the supplements (Appendix A). Theorem 1 thus generalizes non-contextual results in a data-diluted form: effectively replacing n by n/K. The first case is the low-dimensional regime. Perhaps the most relevant is the middle high-dimensional regime. This often holds in the case of bigrams ($K = k$) and trigrams ($K = k^2$). In this case we recover the prediction-dimensionless (in k) bound. For large K and k, this loss is negligible compared to the minimax risk (Falahatgar et al., 2016), implying true competitivity (for more on minimax risks see the supplements, Appendix B). Not that in the third (extreme high-dimensional) regime, the unobserved contexts give no advantage to this oracle, leading to a competitivity that does not decay but also does not depend on the number of contexts.

Stronger competitivity In this paper, we conjecture that the advantage of backoff is in providing a much stronger form of competitivity. We use the following intuition. The competitivity of the Good–Turing estimator shows that the difficulty of the problem is *not* in estimating the multiset of probabilities *as much as it is* in identifying in which permutation they map to the categories, the only task that the oracle has to perform, given data. One can think of this as aligning the tail of the distribution. In the contextual setting, this intuition still persists. But another joins it: tails are often related across contexts, and since the identities of the categories are shared across contexts, the oracle then ought to be able to better align within each context too. To make this intuition concrete, consider the following idealized scenario.

Consider an oracle $\mathcal{F}_2$, that knows P has exactly $m \leq k$ non-zero columns, but not which ones they are. Thus $\mathcal{P} \in \mathcal{F}_2$ are indexed simply by m. This idealizes two aspects of the problem. First, there is a non-ambiguous tail (the zeros) in each context. And second, all these tails are clearly correlated across contexts by being aligned. It turns out that the Kneser–Ney backoff estimator Q^{KN} strongly competes with this oracle.

Theorem 2 *If $n \gg k$ then*

$$\epsilon(Q^{\mathsf{KN}}, \mathcal{F}_2) \leq \mathcal{O}\left(\frac{k}{n}\right).$$

Consider the regime where $n > k^2/K$. Usually $K \geq k$, think of N-grams, in which case it would suffice that $n > K$. It is easy to verify that in this regime the bound of Theorem 2 is strictly better than that of Theorem 1, and has the distinct benefit of not scaling with the dimension of contexts. It is also worth mentioning that the proof of this result (supplements, Appendix A) gives the finer class-by-class competitive loss of $\mathcal{O}(m/n)$. Q^{KN} achieves this without prior knowledge of m.

We believe this simple case reinforces the idea that tail alignment *across contexts* is a fruitful avenue for competitivity in the contextual case, just as tail alignment *within contexts* was a fruitful one in the non-contextual case. Classes that ease the latter alignment, such as power law decay or small effective support size, enjoy lower competitive loss by Q^{GT}. These factors are invariant under permutations. This suggests that unlike the oracle $\mathcal{F}_1$ that permutes within each context separately, the natural notion of invariance in the contextual case ought to be under simultaneous permutation across contexts, i.e. permutation of entire *columns* of P.

4 Contextual Good–Turing

Motivated by this theoretical foray, and with the goal of giving a principled underpinning to Kneser–Ney smoothing and the hope of deriving estimators with more favorable competitive properties, we revisit the derivation of the original Good–Turing estimator and extend it to the contextual case.

Good–Turing is based upon an empirical Bayes construction. To parallel it in the contextual setting, assume the multiset of the columns of P is known and that P is instanced via a uniformly random permutation of these columns. Let x be some context and y be some prediction, such that our ultimate goal is to estimate P_{xy}. The chance that y is any particular $j \in [k]$ is a priori $1/k$, and thus $\mathbb{E}[P_{xy}] = 1/k$. But having made some observations, we would like to determine the conditional expectation of P_{xy} given that there are $\mathbf{n} := (n_i)_{i\in[K]}$ samples in each context and given that y has been observed in each context $\mathbf{c} := (C_{i,y})_{i\in[K]}$ times[1]. Starting with the simple observation that

$$\Pr\{\mathbf{c} \mid \mathbf{n},\ y = j\} = \prod_i \binom{n_i}{C_{ij}} P_{ij}^{C_{ij}} (1 - P_{ij})^{n_i - C_{ij}},$$

one can show that

$$\begin{aligned}\mathbb{E}[P_{xy} \mid \mathbf{n}, \mathbf{c}] &= \frac{\sum_j \prod_i P_{ij}^{C_{ij} + \mathbf{1}\{i=x\}} (1 - P_{ij})^{n_i - C_{ij}}}{\sum_j \prod_i P_{ij}^{C_{ij}} (1 - P_{ij})^{n_i - C_{ij}}} \\ &\equiv \frac{c_x + 1}{n_x + 1} \frac{\mathbb{E}[K_{\mathbf{n}+\mathbf{1}_x, \mathbf{c}+\mathbf{1}_x} | \mathbf{n}]}{\mathbb{E}[K_{\mathbf{n},\mathbf{c}} | \mathbf{n}]}. \end{aligned} \tag{6}$$

Here $K_{\mathbf{n},\mathbf{c}}$ is the number of columns that have exactly the $\mathbf{c}$ count pattern. The expectation of this quantity is column-permutation invariant, thus can be computed from the multiset. The Good–Turing approach is to use this expression as an estimator, substituting the expectations with their empirical counterparts (with

[1]Note that this is *not* the entire information useful to determine the permutation, just local information that makes the task tractable.

the shift of $n_x + 1$ to n_x, since the additional sample is not available empirically):

$$\hat{P}_{xy} = \frac{c_x + 1}{n_x} \frac{K_{\mathbf{n},\mathbf{c}+\mathbf{1}_x}}{K_{\mathbf{n},\mathbf{c}}}. \tag{7}$$

The challenge is that, even in the non-contextual case, these can be highly unreliable, and one needs to smooth them, such as by combining with the empirical distribution in the abundant range or by using absolute discounting. In the contextual case, even more smoothing is needed: $\mathbf{c}$ may be observed, but it's very unlikely that $\mathbf{c} + \mathbf{1}_x$ is, and the estimator degenerates. How can we remedy this?

From Contextual Good–Turing to Classical Back-off Back-off is an intuitive notion, but was originally proposed in an ad hoc fashion. We now show that contextual Good–Turing naturally gives rise to backoff. We start by observing that if one sums the total probability assigned to all symbols that appear μ times in context x, the estimator (7) gives us back the non-contextual Good–Turing estimate of that probability:

$$\begin{aligned}
&\sum_{y:\mathbf{c}(y),c_x(y)=\mu} K_{\mathbf{n},\mathbf{c}(y)} \hat{P}_{xy} \\
&= \frac{\mu+1}{n_x+1} \sum_{y:\mathbf{c}(y),c_x(y)=\mu} K_{\mathbf{n}+\mathbf{1}_x,\mathbf{c}(y)+\mathbf{1}_x} \\
&= \frac{\mu+1}{n_x+1} K_{n_x,\mu+1}(x)
\end{aligned}$$

This shows that (7) simply redistributes this mass. This is the main premise of Kneser–Ney backoff. Does it redistribute it similarly to Q^{KN}? In general, no. But we can identify when exactly it does. We give the following general smoothing strategy, which we can think of as binning. For a given x, choose an equivalence $\sim$, compatible with the contextual Good–Turing estimator, namely that satisfies (1) $\mathbf{c} \sim \mathbf{c}'$ implies $\mathbf{c}_x = \mathbf{c}'_x$ (fixes x) and (2) if $c \sim c'$ then $\mathbf{c}_\sigma \sim \mathbf{c}'_\sigma$ for any permutation σ of $[K]$ (invariant under permutations of contexts). We smooth by spreading probability within each equivalence bin and counting all equivalent $\mathbf{c}$ as being identical:

$$\tilde{P}_{xy} = \frac{c_x + 1}{n_x} \frac{K_{\mathbf{n}+\mathbf{1}_x,\sim\mathbf{c}+\mathbf{1}_x}}{K_{\mathbf{n},\sim\mathbf{c}}}. \tag{8}$$

We can verify that this preserves the mass redistribution property. Let $\mathsf{nnz}(\mathbf{c}) = \sum_i \mathbf{1}\{c_i > 0\}$ count the number of non-zero entries of $\mathbf{c}$. Then the following defines a possible equivalence class:

$$\mathbf{c} \sim \mathbf{c}' \iff \begin{cases} c_x = c'_x \\ \mathsf{nnz}(c) = \mathsf{nnz}(c') \end{cases} \tag{9}$$

In this case we can characterize the redistribution accurately, at least in its idealized form.

Theorem 3 *Use the equivalence relation of Equation* (9) *in the smoothed contextual Good–Turing estimator* (8)*, where the counts K are substituted by their idealized expectations. Let $\mu := c_x$. Then:*

$$\check{P}_{xy} = \frac{\mu+1}{n_x+1} \frac{\sum_j \binom{n_x+1}{\mu+1} P_{xj}^{\mu+1} (1-P_{xj})^{n_x-\mu} \rho_{bj}}{\sum_j \binom{n_x}{\mu} P_{xj}^{\mu} (1-P_{xj})^{n_x-\mu} \rho_{bj}},$$

where $b = \mathsf{nnz}(\mathbf{c})$ and

$$\rho_{bj} = \sum_{S \subset [K]\setminus\{x\} : |S| = b} \prod_{i \in S} [1 - (1-P_{xj})^{n_i}] \prod_{i \in S^c \setminus \{x\}} (1-P_{xj})^{n_i}.$$

We omit the proof of this result, since it's straightforward manipulations. It is more important to observe that, apart from ρ, this is exactly the non-contextual Good–Turing expression. Thus ρ acts as a redistribution coefficient. In general, it does not quite redistribute like Kneser–Ney: unlike it, ρ depends on the context x. Observe however that only the small values (of the order of $1/n_i$) of P_{xj} contribute to ρ. Let us assume that these values are aligned across rows (do not depend on x), that they have a power law decay of index α, and that the n_i are roughly uniform. We can then show that the effect of ρ is asymptotically approximately given by (see supplements, Appendix A):

$$\check{P}_{xy} \sim \frac{\mu + b_y - \alpha}{n + b_y}. \tag{10}$$

For the unseen symbols, when $\mu = 0$, this recovers the Chen-Goodman version of Kneser–Ney smoothing (see Section 2, and note that b_y is negligible with respect to n.)

5 Partial Low-Rank N-gram Backoff

It is enlightening that contextual Good–Turing, an empirical Bayes estimator derived from column-permutation invariance, when properly smoothed, recovers classical forms of backoff under the kind of tail alignment conditions that make these competitive in the first place. It is then natural to ask whether contextual Good–Turing's competitive properties extend further than simple alignment, especially that it is not explicitly aware of it. Indeed, column-permutation invariance has the potential to capture a much richer family of tail structures: the rank of P, its sparsity, the dimension of the manifold on which each row of P lies, such as the embedding dimension in typical neural embeddings, and many other classical structures, are all invariant under such permutation.

What is needed to achieve this generality is a more flexible smoothing of the idealized contextual Good–Turing formula of Equation (6). Based on this idea, we now give a direct generalization of bigram Kneser–Ney smoothing. First, refine the equivalence relation given by (9), and use instead

$$\mathbf{c} \sim \mathbf{c}' \Longleftrightarrow \begin{cases} c_x = c'_x \\ \forall i \ \mathbf{1}\{c_i > 0\} = \mathbf{1}\{c'_i > 0\} \end{cases} \tag{11}$$

Two columns are thus considered equivalent if their non-zero patterns align. This is clearly a coarser binning than maintaining the full identity of $\mathbf{c}$, but is a significant refinement of the partition induced by (9). For (9), apart from the identity of c_x, the partition could be determined fully through $(b_j)_{j\in[k]}$, where $b_j = \mathsf{nnz}(C_{\cdot j}) = \sum_i \mathbf{1}\{C_{ij} > 0\}$. To determine this finer partition one needs the full matrix of non-zero indicators $B = (B_{i,j})_{i\in[X],j\in[k]}$, $B_{i,j} = \mathbf{1}\{C_{ij} > 0\}$. Indeed, this may still be too fine to effectively smooth $K_{\mathbf{n},\mathbf{c}}$ in general. It does however allow us to create a hierarchy of refinements of which itself is one extreme, and Kneser–Ney is another, a projection onto a subspace of one dimension.

The key idea is the following: allow this subspace to be of a larger dimension, say m. Namely, represent the non-zero indicator matrix B by a rank-m approximation. It is not hard to see that for the case $m = 1$, such a representation collapses to $b_j = \sum_i B_{ij}$ and recovers Kneser–Ney backoff. It is also important to appreciate that this is *not* a low-rank representation of the raw count matrix C. That would be similar to a topic model. It is instead a low-rank representation only of C's rare component. We dub it *partial low-rank* (PLR).

PLR can be intuitively thought of as follows:

- Split counts C into two, $C_{\mathsf{abundant}} + C_{\mathsf{rare}}$.
- C_{abundant} is a α-discounted C, call its normalized version A. Use it as is.
- C_{rare} consists of the discounts themselves, equal to α-scaled B. *Don't use it as is.* Instead, factorize it into WH: a thin matrix W and a short one H.
- Combine the A and WH components, using the estimates ν of the missing mass in each row.

If W and H are 1-thin/short, we get Kneser–Ney because the backoff distributions, all rows of WH, become the same. In contrast, PLR allows varying backoff distributions across rows/contexts.

The main fine-print here is that it is desirable to smooth the backoff/rare component too. As mentioned in Section 2, this was one of Chen and Goodman's main improvements on Kneser–Ney: they smoothed the backoff counts b_j [2]. A general simultaneous rank-reduction and smoothing technique was recently proposed by Falahatgar et al. (2017), as a simple modification of the multiplicative-updates algorithm for non-negative matrix factorization. That is what we propose to apply to B for the general rank case, and it once again recovers the Chen-Goodman version when $m = 1$. PLR is detailed in Algorithm 1.

Algorithm 1 Partial Low-Rank (PLR)

inputs
  $K \times k$ count matrix C, rank m, discount α, number of iterations T, (optionally) W_0 and H_0
outputs
  Conditional distribution matrix Q^{PLR}, (optionally) its components A, ν, W, and H
Do α absolute-discounting on each row of C:
  Get abundant component A, $A_{ij} = (C_{ij} - \alpha)/n_i$
  Get missing mass vector ν, $\nu_i = d_i\alpha/n_i$, where $d_i = \sum \mathbf{1}\{C_{ij} > 0\}$
Generate indicator matrix B, $B_{i,j} = \mathbf{1}\{C_{ij} > 0\}$
If W_0 and H_0 are *not* provided,
  Initialize $K \times m$ matrix W_0:
    Set it to be uniform $1/m$ in each row
  Initialize $m \times k$ matrix H_0:
    Split B into m random row-blocks
    Collapse each to get $m \times k$ soft-counts $\tilde{H}_0$
    Do α absolute-discounting on each row of $\tilde{H}_0$
for $t = 1$ to T **do**
  $\tilde{W}_t \longleftarrow \left[(B \oslash W_{t-1}H_{t-1})H_{t-1}^{\intercal}\right] \otimes W_{t-1}$
  Add $1/2$ to each row of the soft-count matrix $\tilde{W}$ and normalize each row to obtain W_t
  $\tilde{H}_t \longleftarrow [W_t^{\intercal}(B \oslash W_tH_{t-1})] \otimes H_{t-1}$
  Do α absolute-discounting on each row of the soft-count matrix $\tilde{H}_t$ to obtain H_t
end for
return $Q^{\mathsf{PLR}} = A + \mathsf{diag}(\nu) \cdot W_T H_T$

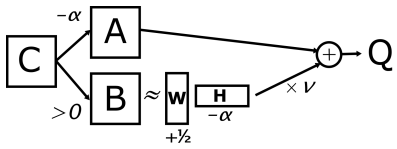


Figure 1: A graphical representation of the Partial Low-Rank (PLR) algorithm, Algorithm 1.

Algorithm 1 is illustrated graphically in Figure 1. The notations $\otimes$ and $\oslash$ refer to element-wise multiplication and division. The for-loop in which they appear is the modified multiplicative updates to factorize B, see details in (Falahatgar et al., 2016). When $K = k$, the PLR algorithm applies as-is to perform bigram smoothing. With $m = 1$, it is identical to Kneser–Ney. With larger m, it describes a much richer set of tail alignments. One could extend this to N-grams in two ways, directly by setting $K = k^{N-1}$, or by nesting like

[2] This contribution surprisingly appears just as a tucked away mention in Section 4.1 of (Chen and Goodman, 1999).

Chen-Goodman's recursive application of backoff, by fixing sub-contexts (Chen and Goodman, 1999). This general Nested Partial Low-Rank algorithm, NPLR, along with other technical details of both algorithms, is presented in the supplements (Appendix C.) Here we merely illustrated it in the trigram ($N = 3$) case graphically, in Figure 2. Note that all iterations can be very efficiently implemented using either sparse matrix manipulations or dictionaries. They run in linear time in n, as only observed contexts need to be tracked.

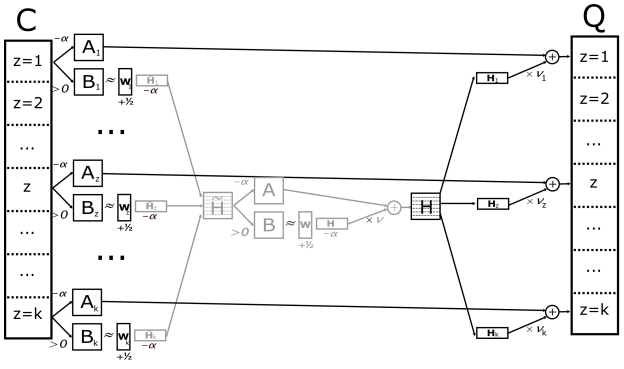


Figure 2: A graphical representation of the Nested Partial Low-Rank (NPLR) algorithm for the case of trigrams. Note how the PLR backoffs in each subcontext z are combined to create an effective "augmented" bigram model (we say this because there are mk contexts instead of the k of Chen-Goodman). A nested backoff is then applied to this augmented bigram. When $m > 1$, the iterations of the trigram and bigram PLRs are themselves nested: the trigram H is updated only after the bigram H is updated. When $m = 1$, each iteration converges in one step, recovering the Chen-Goodman algorithm.

6 Experiments

In this section, we reinforce the theory and concepts so far with experiments. The goal is to show both that we can improve traditional smoothing techniques and that we can better aid neural language models. Toward the first goal, we show improved performance not only in language modeling, that is predicting the next word given its history, but also the Global Terrorism Database, where we predict the target type for the next attack for a given city.

N-gram language models We perform word-level language modeling on the Penn TreeBank (PTB) data set (Mikolov, 2012) using standard splits (929k training tokens, 73k validation tokens, and 82k test tokens.) The vocabulary size is 10k. We compare different models in terms of their perplexity, the exponentiated *exp*(cross-entropy). These express the uncertainty in prediction, therefore, the lower, the better. We compare the original Kneser–Ney (KN) with the NPLR algorithm (nested version of PLR). NPLR allows us to control rank at every back-off level (m_2 at bigram, m_3 at trigram, $\cdots$). We look at two variants, both of which are overall 5-gram models: **(1)** partial low-rank applied either at only the bigram level PLR ($m_2 = 30$, $m_3 = m_4 = m_5 = 1$), or **(2)** at both the trigram *and* bigram levels ($m_2 = 18$, $m_3 = 4$, $m_4 = m_5 = 1$). This means in particular that at higher levels we maintain the KN structure in both cases, i.e. $m_4 = m_5 = 1$. We set $(\alpha_2, \cdots, \alpha_5) = (0.8, 0.9, 0.9, 0.9)$, close to estimates on held-out data (Ben Hamou et al., 2017; Ohannessian and Dahleh, 2012).

We also include three neural models, two 5-grams, one feedforward and one LSTM, and one 13-gram feedforward, all reported in (Chelba et al., 2017). Table 1 summarizes these results. We show significant improvement over KN and a modest improvement over the 5-gram feedforward model. The gap with the 5-gram LSTM is expected, considering its extensive weight-sharing, which we surmise latches to additional structure in language. To the best of our knowledge, no other direct N-gram smoothing technique, and in particular none of the attempts to explain and generalize backoff including hierarchical Bayesian models (Teh, 2006), have reported such clear gains.

Table 1: Perplexity on PTB — NPLR smoothing surpasses KN and competes with feedforward NNs — see (Chelba et al., 2017) for details

	Method	Test Perplexity
5-gram	KN	143
5-gram	NPLR **variant (1)**	**131**
5-gram	Feedforward NN	127
5-gram	NPLR **variant (2)**	**126**
13-gram	Feedforward NN	125
5-gram	LSTM	103

LSTM language models with data noising Recently, Xie et al. (2017) utilized smoothing techniques as a data noising method for LSTM language models. Replacing some words in the input data changes the counts of N-grams in a way that applying an empirical estimator to the noised data is similar to applying N-gram smoothing techniques to the non-noisy data. In Table 2 we show that PLR, if used as a data noising technique, improves the perplexity of an LSTM language model more than all the other techniques, even the best noising based on Kneser–Ney. The LSTM setup is the same as the large-network used in (Xie et al., 2017) (2-layer LSTM with 1500 hidden units) and we trained our models using the same setup as

in (Xie et al., 2017; Zaremba et al., 2014). The noising parameter is tuned based on the validation data and the result is reported for the best noising parameter.

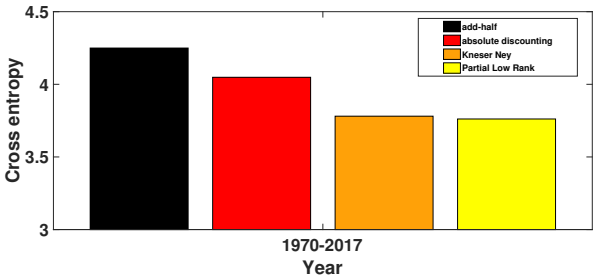
Table 2: Smoothing as data noising for LSTMs — Effects on perplexity for the Penn Tree Bank data set — see (Xie et al., 2017) for details

Noise scheme	Validation	Test
none	81.6	77.5
unigram	79.4	76.1
bigram Kneser–Ney	76.2	73.4
PLR	**75.5**	**72.7**

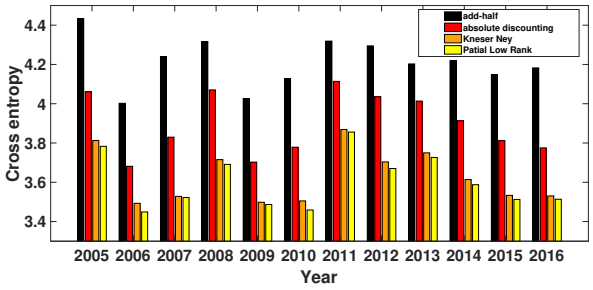
Global terrorism target prediction Language modeling is *the* flagship application of smoothing techniques and most of the new techniques are put to test there. However, the challenge of predicting rarely seen elements conditioned on some context, is present in a multitude of other natural applications. Even the power-law structure of language is also present in many natural phenomena. Here we explore one such alternative application: predicting terror incidents. We use the Global Terrorism Database (GTD) (START, 2018), which includes systematic data about more than 180,000 cases of domestic and international terrorist events from 1970 through 2017 for more than 36,000 cities around the globe. The task that we consider is to estimate the probability of the next attack in a given city has a specific target. This is a contextual probability estimation: the context is the city and the prediction is the target type. There are 114 different target sub-types, such as restaurants, banks, hotels, and etc.

We predict the type of the next attack in each city based on the prior incidents that happened in that year and compare four different estimators: row-wise add-half, row-wise absolute discounting, Kneser–Ney (KN), and PLR. Figures 3(b) and 3(a) show the benefit of using data from different cities (different rows of the count matrix) when predicting for the target type in a particular city. PLR and KN always have significantly better performances in predicting the next attack's type than row-wise estimators such as add-half and absolute discounting. Also, PLR shows an edge over KN in terms of generalization power. In all experiments, PLR is set to use rank $m = 5$ and the discount factor $\alpha = 0.9$, and is run for 100 iterations.

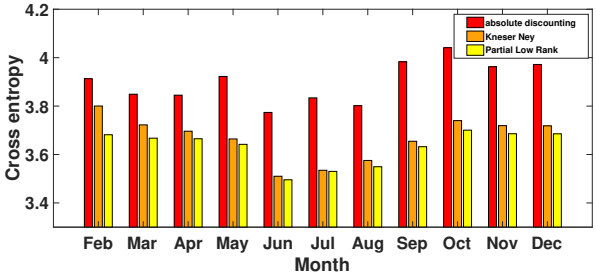
Lastly, we study how performance changes as the amount of data available for estimation varies. For this, we base our predictions for a particular year and predict the target type for two weeks in a month, using all the incidents before that time period in that year. As we move towards the end of the year, the amount of data available for estimation increases. Figure 3(c) shows how cross-entropy changes for different months in the year 2017. It is clear that the relative gain of PLR is more in the case when data is not abundant. More experiments are reported in the supplements.



(a) All data



(b) Yearly data



(c) Effect of training size

Figure 3: Test cross-entropy with training / test periods: (a) dates before / after 2015, (b) the first 10 months / the last two months of the year, and (c) all dates prior to a month / that month of the year

7 Conclusion

We initiated a theoretical exploration of N-gram smoothing, through the lens of competitivity. This allowed us to understand backoff from first principles and discover powerful new generalizations. Beyond matching the performance of feed-forward neural networks, these new algorithms enable better data augmentation for training general neural language models, and even show gains in applications beyond language modeling. We hope this provides momentum towards a mature theory and practice of competitive contextual distribution estimation.

References

Ben Hamou, A., Boucheron, S., and Ohannessian, M. I. (2017). Concentration Inequalities in the Infinite Urn Scheme for Occupancy Counts and the Missing Mass, with Applications. *Bernoulli*.

Braess, D. and Sauer, T. (2004). Bernstein polynomials and learning theory. *Journal of Approximation Theory*, 128(2):187–206.

Chelba, C., Norouzi, M., and Bengio, S. (2017). N-gram language modeling using recurrent neural network estimation. *arXiv preprint arXiv:1703.10724*.

Chen, S. F. and Goodman, J. (1999). An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–394.

Dai, Z., Yang, Z., Yang, Y., Cohen, W. W., Carbonell, J., Le, Q. V., and Salakhutdinov, R. (2019). Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.

Falahatgar, M., Ohannessian, M. I., and Orlitsky, A. (2016). Near-optimal smoothing of structured conditional probability matrices. In *NIPS*, pages 4860–4868.

Falahatgar, M., Ohannessian, M. I., Orlitsky, A., and Pichapati, V. (2017). The power of absolute discounting: all-dimensional distribution estimation. In *NIPS*, pages 6660–6669.

Gandhe, A., Metze, F., and Lane, I. (2014). Neural network language models for low resource languages. In *Fifteenth Annual Conference of the International Speech Communication Association*.

Gong, C., He, D., Tan, X., Qin, T., Wang, L., and Liu, T.-Y. (2018). Frage: frequency-agnostic word representation. In *NIPS*, pages 1334–1345.

Good, I. J. (1953). The population frequencies of species and the estimation of population parameters. *Biometrika*, pages 237–264.

Hutchinson, B., Ostendorf, M., and Fazel, M. (2015). A sparse plus low-rank exponential language model for limited resource scenarios. *IEEE Transactions on Audio, Speech, and Language Processing*, 23(3):494–504.

Kamath, S., Orlitsky, A., Pichapati, D., and Suresh, A. T. (2015). On Learning Distributions from their Samples. In *COLT*, pages 1066–1100.

Kneser, R. and Ney, H. (1995). Improved backing-off for m-gram language modeling. In *1995 International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 181–184. IEEE.

Merity, S., Keskar, N. S., and Socher, R. (2017). Regularizing and optimizing LSTM language models. *arXiv preprint arXiv:1708.02182*.

Mikolov, T. (2012). Statistical language models based on neural networks. *Presentation at Google, Mountain View, 2nd April*, 80.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Karafiát, M., Burget, L., Černockỳ, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Ohannessian, M. I. and Dahleh, M. A. (2012). Rare Probability Estimation under Regularly Varying Heavy Tails. In *COLT*, pages 21–1.

Orlitsky, A. and Suresh, A. T. (2015). Competitive distribution estimation: Why is Good–Turing good. In *NIPS*, pages 2143–2151.

Paninski, L. (2004). Variational Minimax Estimation of Discrete Distributions under KL Loss. In *NIPS*, pages 1033–1040.

Parikh, A. P., Saluja, A., Dyer, C., and Xing, E. P. (2013). Language modeling with power low rank ensembles. *arXiv preprint arXiv:1312.7077*.

START (2018). National consortium for the study of terrorism and responses to terrorism, Global Terrorism Database. `https://www.start.umd.edu/gtd`.

Takase, S., Suzuki, J., and Nagata, M. (2018). Direct output connection for a high-rank language model. *arXiv preprint arXiv:1808.10143*.

Teh, Y. W. (2006). A hierarchical bayesian language model based on pitman-yor processes. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 985–992. Association for Computational Linguistics.

Valiant, G. and Valiant, P. (2015). Instance optimal learning. *arXiv preprint arXiv:1504.05321*.

Xie, Z., Wang, S. I., Li, J., Lévy, D., Nie, A., Jurafsky, D., and Ng, A. Y. (2017). Data noising as smoothing in neural network language models. *arXiv preprint arXiv:1703.02573*.

Yang, Z., Dai, Z., Salakhutdinov, R., and Cohen, W. W. (2017). Breaking the softmax bottleneck: A high-rank rnn language model. *arXiv preprint arXiv:1711.03953*.

Zaremba, W., Sutskever, I., and Vinyals, O. (2014). Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

A Deferred Proofs

A.1 Proof of Theorem 1

We will need the following lemma.

Lemma 4 *Let $X \sim Poi(\lambda)$, then, $\mathbb{E}[\frac{1}{\sqrt{x+1}}] \leq \frac{1}{\sqrt{\lambda}}$ and $\mathbb{E}[\frac{1}{X}] \leq \frac{1}{\lambda}$.*

Proof

$$\begin{aligned}
\mathbb{E}[\frac{1}{\sqrt{X+1}}] &= \sum_{x=0}^{\infty} e^{-\lambda}\frac{\lambda^x}{x!}\frac{1}{\sqrt{x+1}} \\
&= \sum_{x=0}^{\infty} \frac{\sqrt{x+1}}{\lambda} e^{-\lambda}\frac{\lambda^{x+1}}{(x+1)!} \\
&= \sum_{x=0}^{\infty} \frac{\sqrt{x}}{\lambda} e^{-\lambda}\frac{\lambda^{x}}{x!} \\
&= \frac{\mathbb{E}[\sqrt{X}]}{\lambda} \leq \frac{\sqrt{\mathbb{E}[X]}}{\lambda} = \frac{1}{\sqrt{\lambda}}
\end{aligned}$$

where the last line is by Jensen's inequality. Proof of the other part is similar and thus omitted.

Proof [Theorem 1] First note that if we do not observe any samples from a row, the competitive loss will be zero as we do same as the oracle.

Rewriting the competitive loss in (4):

$$\begin{aligned}
&\epsilon(Q^{\mathsf{GT}}, \mathcal{F}_1) \\
&\overset{(a)}{\leq} \max_{\pi} \max_{\mathcal{P}} \max_{P\in\mathcal{P}} \Bigg[\mathbb{E}_{(x^n,y^n)\sim\pi P} \sum_i \pi_i \sum_j P_{ij} \log \frac{P_{ij}}{Q^{\mathsf{GT}}_{ij}((x^n,y^n))} \\
&\quad - \max_{\pi} \max_{P\in\mathcal{P}} \mathbb{E}_{(x^n,y^n)\sim\pi P} \sum_i \pi_i \sum_j P_{ij} \log \frac{P_{ij}}{Q^{\mathcal{C}}_{ij}((x^n,y^n))}\Bigg] \\
&\overset{(b)}{\leq} \max_{\pi} \max_{\mathcal{P}} \max_{P\in\mathcal{P}} \Bigg[\mathbb{E}_{(x^n,y^n)\sim\pi P} \sum_i \pi_i \sum_j P_{ij} \log \frac{P_{ij}}{Q^{\mathsf{GT}}_{ij}((x^n,y^n))} \\
&\quad - \max_{P\in\mathcal{P}} \mathbb{E}_{(x^n,y^n)\sim\pi P} \sum_i \pi_i \sum_j P_{ij} \log \frac{P_{ij}}{Q^{\mathcal{C}}_{ij}((x^n,y^n))}\Bigg] \\
&\overset{(c)}{=} \max_{\pi} \mathbb{E}_{x^n} \sum_i \pi_i \Bigg[\max_{\mathcal{P}_i} \max_{p_i\in\mathcal{P}_i} \mathbb{E}_{y^n|x^n} \sum_j P_{ij} \log \frac{P_{ij}}{Q^{\mathsf{GT}}_{ij}((x^n,y^n))} \\
&\quad - \max_{p_i\in\mathcal{P}_i} \mathbb{E}_{y^n|x^n} \sum_j P_{ij} \log \frac{P_{ij}}{Q^{\mathcal{C}}_{ij}((x^n,y^n))}\Bigg] \\
&\overset{(d)}{\leq} \max_{\pi} \mathbb{E}_{n_i\sim\pi} \sum_{i=1}^{K} \pi_i \min\{\frac{1}{\sqrt{n_i}}, \frac{k}{n_i}\} \\
&\overset{(e)}{\leq} \max_{\pi} \sum_{i=1}^{K} \pi_i \mathcal{O}(\min\{1, \frac{1}{\sqrt{n\pi_i}}, \frac{k}{n\pi_i}\}) \\
&\overset{(f)}{\leq} \mathcal{O}(\min\{1, \sqrt{\frac{K}{n}}, \frac{K\cdot k}{n}\})
\end{aligned}$$

where (a) is because any estimator gives us an upper bound on the competitive loss, (b) is by removing $\max_\pi$ from the second term in the bracket and make the term inside the bracket larger. Also, (c) follows from the

independence of rows' distributions where $\mathcal{P}_i$ indicates the class of distributions for the ith row induced by $\mathcal{P}$, and p_i is ith row distribution. Results of Orlitsky and Suresh (2015) yields (d), (e) is by Lemma 4 and (f) follows from concavity of $\sqrt{x}$ function. Note that adding $\mathcal{O}(1)$ is because the competitive loss for each row is always less than a constant and this trivial bound yields a better upper bound for the range $n < K$.

A.2 Proof of Theorem 2

We show that Kneser–Ney estimator is competitive with respect to this oracle. Consider the case where $n \gg m$. Since the oracle knows there are m columns with non-zero entries, with enough samples, it gets to identify all those columns. Estimating each row of the transition probability then becomes an easier task for the oracle in the sense that the effective support size for each row is $m \ll k$. For example, if there is no observation from a particular row, the oracle will assign unifrom distribution over m non-zero columns, whereas without knowing the fact that the effective support size is m, one needs to assign uniform distribution over $k \gg m$ elements. We show that using Kneser–Ney estimator, the mass we assign to $k - m$ zero columns in each row is negligible and therefore we can compete with the oracle. To show competitivity we only need to bound the mass our estimator assigns to the zero columns, since the oracle does not have any extra knowledge about rest of the elements.

Lemma 5 *If an estimator q for a given row assigns an expected mass η to the zero columns, then*

$$\epsilon(q) \leq \mathcal{O}\left(\log \frac{1}{1-\eta}\right),$$

where ϵ is the contribution of that row to the competitive loss.

Proof Since the oracle does not have any extra knowledge about the probabilities except that the support size is m instead of k, we can do as well as the oracle, except for the fact that we are assigning $1 - \eta$ mass to the m non-zero columns. Therefore the loss for that particular row will be $\mathcal{O}(\log \frac{1}{1-\eta})$.

Lemma 6 (Falahatgar et al. (2017)) *Let q^{GT} be the Good–Turing estimator applied to estimate the backoff distribution over k elements using n samples and let $\mathbb{E}[\Phi_1]$ be the expected number of elements appearing exactly once in the sample set. Then the expected mass assigned to the unobserved elements in the samples is*

$$\eta \propto \frac{\mathbb{E}[\Phi_1]}{n}$$

Proof [Theorem 2] We first bound the amount of mass the Kneser–Ney estimator assigns to the zero-columns. The missing mass assigned to unobserved columns is the same as the mass assigned to the unobserved elements by Good–Turing estimator applied to the vector of the number of distinct elements seen in each column. By Lemma 6 we can bound the missing mass assigned by Kneser–Ney by $\frac{\mathbb{E}[\Phi_1]}{n}$, where $\mathbb{E}[\Phi_1]$ is the expected number of elements appeared once in the vector of column distinct elements. But $\mathbb{E}[\Phi_1] < m$ trivially. Therefore, using Lemma 5 we have $\epsilon(Q^{\mathrm{KN}}, \mathcal{F}_2) \leq \mathcal{O}\left(\log \frac{1}{1-\frac{m}{n}}\right) \leq \mathcal{O}\left(\frac{m}{n}\right)$ for every $m \leq k$, and the claim follows by choosing the loosest bound.

A.3 Derivation of Equation (10)

Recall the expression of the smoothed Good–Turing estimator from Theorem 3, with the common factors simplified:

$$\check{P}_{xy} = \frac{\sum_j P_{xj}^{\mu+1}(1-P_{xj})^{n_x-\mu}\rho_{bj}}{\sum_j P_{xj}^{\mu}(1-P_{xj})^{n_x-\mu}\rho_{bj}},$$

where $b = \mathsf{nnz}(\mathbf{c})$ and

$$\rho_{bj} = \sum_{S\subset[K]\setminus\{x\}\,:\,|S|=b} \prod_{i\in S}[1-(1-P_{xj})^{n_i}] \prod_{i\in S^{\mathsf{c}}\setminus\{x\}} (1-P_{xj})^{n_i}.$$

Without ρ, this is the non-contextual Good–Turing estimator. In order to characterize the effect of ρ, let us make the following simplifying assumption:

- Let $n_i = m := n/K =$ for all i.
- Let all P_{xj} (at least at contributing values near $1/m$) be approximated by the same p_j.
- We have a power-law p_j, that is $p_j \propto j^{-1/\alpha}$.

Then, we can write:

$$\begin{aligned}
\rho_{bj} &= \sum_{S\subset[K]\setminus\{x\}\,:\,|S|=b} \prod_{i\in S}[1-(1-P_{xj})^{n_i}] \prod_{i\in S^c\setminus\{x\}} (1-P_{xj})^{n_i} \\
&\approx \sum_{S\subset[K]\setminus\{x\}\,:\,|S|=b} \prod_{i\in S}[1-(1-p_j)^{m}] \prod_{i\in S^c\setminus\{x\}} (1-p_j)^{m} \\
&= \binom{K-1}{b}[1-(1-p_j)^m]^b(1-p_j)^{m(K-1-b)} \\
&= \binom{K-1}{b}[(1-p_j)^{-m}-1]^b(1-p_j)^{m(K-1)} \\
&\approx \binom{K-1}{b}(mp_j)^b(1-p_j)^{m(K-1)}
\end{aligned}$$

It follows that:

$$\begin{aligned}
\check{P}_{xy} &\approx \frac{\sum_j p_j^{\mu+b+1}(1-p_j)^{n-\mu}}{\sum_j p_j^{\mu+b}(1-p_j)^{n-\mu}}, \\
&= \frac{\mu+b+1}{n+b+1}\frac{\sum_j \binom{n+b+1}{\mu+b+1} p_j^{\mu+b+1}(1-p_j)^{n-\mu}}{\sum_j \binom{n+b}{\mu+b} p_j^{\mu+b}(1-p_j)^{n-\mu}}, \\
&\sim \frac{\mu+b-\alpha}{n+b} \quad \text{asymptotically, for } \mu+b>0.
\end{aligned}$$

The last asymptotic expression follows from the typical analysis using regular variation / power-laws (see, for example, Ohannessian and Dahleh (2012) and Ben Hamou et al. (2017)). Since the b term in the denominator will be generally negligible compared to n (for power-laws, the *overall* number of distinct symbols grows as n^α with $\alpha \in (0,1)$), it is clear that whenever $\mu = 0$ and $b \geq 1$, the mass is distributed proportionally to $b-\alpha$. This doesn't inform about the $b = 0$ case, but by redistributing it uniformly one recovers a single-depth version of Kneser–Ney backoff. More generally, one could iterate this as is done by Chen-Goodman and as generalized in Section 5.

B Minimax risks

To place the competitivity results in context, we also provide some minimax analysis for the estimation of the conditional probability matrix of size $K \times k$ using n samples. The minimax risk for the class of *all* distributions with K contexts and prediction space size k is:

$$r_n(\Delta_{K,k}) = \min_Q \max_{\pi,P} \mathbb{E}_{(x^n,y^n)\sim\pi P} \sum_{i=1}^K \pi_i \sum_{j=1}^k P_{ij} \log \frac{P_{ij}}{Q_{ij}}$$

For the special case of $K = 1$ this definition will be the same as one-dimensional minimax risk (see Kamath et al. (2015)). The non-contextual minimax risk has been widely studied and fully resolved in the most general case, namely all *i.i.d.* distributions. In particular, Braess and Sauer (2004) showed that for the range $n \gg k$,

$$r_n(\Delta_{1,k}) = \frac{k-1}{2n}\left(1+o(1)\right),$$

and Paninski (2004) showed that for the range $n \ll k$,

$$r_n(\Delta_{1,k}) = \log\frac{k}{n}.$$

A more refined non-contextual minimax risk was defined in Falahatgar et al. (2017), capturing the dependence of minimax risk based on the number of distinct elements D observed in n samples. For a given n and k, let $\mathcal{P}_d$ be the set of all distributions for which $\mathbb{E}[D] \leq d$. Then for some constant c,

$$r_n(\mathcal{P}_d) \leq \frac{d}{n} \log \frac{k - \frac{d}{2}}{\frac{d}{2}} + c \cdot \frac{d}{n}$$

Using an estimator for each row independently based on the samples observed from that particular row upper bounds the minimax risk, namely,

$$\begin{aligned}
& r_n(\Delta_{K,k}) \\
& \leq \max_{\pi} \mathbb{E}_{x^n \sim \pi} \sum_{i=1}^{K} \pi_i \min_{Q_i} \max_{P_i} \mathbb{E}_{y^n | x^n \sim P_i} \sum_{j=1}^{k} P_{ij} \log \frac{P_{ij}}{Q_{ij}} \\
& \leq \max_{\pi} \mathbb{E}_{x^n \sim \pi} \sum_{i=1}^{K} \pi_i r_{n_i}(1, k) \\
& \overset{(b)}{\leq} \max_{\pi} \mathbb{E}_{x^n \sim \pi} \sum_{i=1}^{K} \pi_i \left(\frac{d_i}{n_i} \log \frac{k - \frac{d_i}{2}}{\frac{d_i}{2}} + c \cdot \frac{d_i}{n_i} \right) \qquad (12)
\end{aligned}$$

where d_i is the expected number of distinct elements observed from n_i observations of row i, maximum over all distributions possible for that row. While equation (12) is convoluted, it is insightful in the extreme ranges. In the following examples, we elucidate some of these case.

Example 7 *Consider the case where $n \gg K \cdot k$, therefore, in each row, there will be enough samples and $d_i = k$. In this case, based on equation* (12)*,*

$$r_n(\Delta_{K,k}) \leq \max_{\pi} \mathbb{E}_{\pi} \sum_{i=1}^{K} \pi_i c \cdot \frac{k}{n_i} \leq \mathcal{O}(\frac{K \cdot k}{n})$$

Theorem 8 *For the range $n \gg K * k$,*

$$r_n(\Delta_{K,k}) \geq \Omega \left(\frac{K \cdot k}{n} \right)$$

Proof We fix π to be uniform, then with high probability each row will have $\frac{n}{K} \pm \sqrt{\frac{n}{K}}$ samples and then the proof follows from the non-contextual distribution estimation lower bound.

Based on Theorem 8, in the range where $n \gg K \cdot k$, an $\mathcal{O}((K \cdot k)/n)$ competitivity gives optimal rates, whereas $\mathcal{O}(\sqrt{K/n})$ competitivity implies an optimal estimator, even to the constant.

Example 9 *Consider the case where $n \ll k$, namely, the sample size is much smaller than the alphabet size. Therefore, the worst case in* (12) *happens when all d_is are equal to n_is and* (12) *gives us the upper bound of*

$$r_n(\Delta_{K,k}) \leq \max_{\pi} \mathbb{E}_{\pi} \sum_{i=1}^{K} \pi_i \log \frac{k}{n_i} \leq \mathcal{O} \left(\log \frac{K \cdot k}{n} \right)$$

Theorem 10 *For the range $n \ll k$,*

$$r_n(\Delta_{K,k}) \geq \Omega \left(\log k \right)$$

Proof If $K \gg n$, we don't get to see most of the rows and since for the unobserved rows the minimax risk is $\log k$, we have the Theorem. In the other case, where we get to observe most of the rows, the number of observations from each row is $\leq n \ll k$ and the loss in each row will be $\Omega(\log k)$, hence the Theorem.

Based on Theorem 10, in the regime where $n \ll k$, an $\mathcal{O}(1)$ competitivity is acceptable.

Algorithm 2 Nested Partial Low-Rank (NPLR)

global parameters
  Partial ranks at every level $m_2, \cdots, m_N$
  Discounts at every level $\alpha_2, \cdots, \alpha_N$
inputs
  N-gram $k^{N-1} \times k$ count matrix C
  Number of iterations T
  persistent variables
    W_z and H_z for all subcontexts $z \in [k]^{N-2}$
outputs
  Distribution matrix Q^{NPLR}
for $t = 1$ to T **do**
  for each subcontext $z \in [k]^{N-2}$ **do**
    Call PLR on the z-restricted bigram of C:
      Use m_N, α_N, and just $T' = 1$ iteration
      If $t > 1$, initialize with W_z and H_z
    Recover the components A_z, ν_z, W_z [update persistent copy, and $\tilde{H}_z$
  end for
  if $N > 2$ **then**
    Call $N-1$ NPLR with rank m_{N-1} on the combined $(\tilde{H}_z)_{z \in [k]^{N-2}}$ for $T' = 1$ iteration
    Recover the distribution matrix $(H_z)_{z \in [k]^{N-2}}$ [update persistent copy]
  end if
end for
Assemble $Q_z = A_z + \mathsf{diag}(\nu_z) \cdot W_z H_z$ into Q^{NPLR}
return Q^{NPLR}

C Nested PLR (NPLR) Algorithm

The nested-recursive version NPLR for applying PLR to smooth N-grams is given by Algorithm 2.

It is worth noting that when all m's are set to 1, NPLR reduces to the nested version of Kneser–Ney backoff suggested by Chen and Goodman (see Section 4.1.6 in Chen and Goodman (1999)). It is therefore a strict generalization of that approach. On the other hand, when $m > 1$, NPLR performs the factorization iterations recursively from the higher-level N-gram level, all the way to the bigram level and back (see Figure 2 in the main text).

Further details

Similarly to most local-search algorithms, though its objective function is implicit, multiplicative updates in both PLR and NPLR can benefit from acceleration and noising. We did not include these in the pseudocodes in order not to clutter them, but they are easy to describe. Acceleration can be achieved by moving further in the direction of the update. As for noising, we found that a very effective way to do noising is to periodically reset H by random blocking, just like in the initialization, and then perform several (of the order of the rank) updates of H. This has the effect of bringing H back towards the previous iteration (since W is fixed), while potentially pulling it away from local optima or slowing saddle points.

Another subtle comment is that when NPLR calls PLR, it passes through *soft* counts, which can take values between 0 and 1. A soft count version of absolute discounting was proposed in Falahatgar et al. (2016), which can then be used as follows:

$$\hat{A}_{ij} = \begin{cases} (C_{ij} - \alpha)/n_i & C_{ij} \geq 1 \\ C_{ij}(1-\alpha)/n & C_{ij} < 1 \end{cases}$$

which means that the total subtracted missing mass is still $\nu_i = d_i \alpha / n_i$, but where the number of distinct elements now also incorporates the fractional counts:

$$d_i = \sum \mathbf{1}\{C_{ij} \geq 1\} + C_{ij} \mathbf{1}\{C_{ij} < 1\}$$

The only aspect to clarify is that this also requires to modify the indicator matrix B to account for the fractional counts, as follows (one can easily check that this is equivalent to $\mathbf{1}\{C_{ij>0}\}$ when the counts are whole):

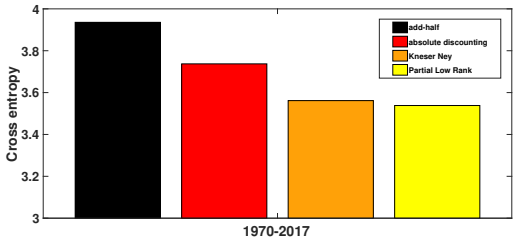
$$B_{ij} = (C_{ij} - n_i A_{ij})/\alpha.$$

Finally, it is worth noting that we can ensemble the learned models by averaging. We find that blindly averaging a couple of runs with random restarts performs better than choosing the best by validation.

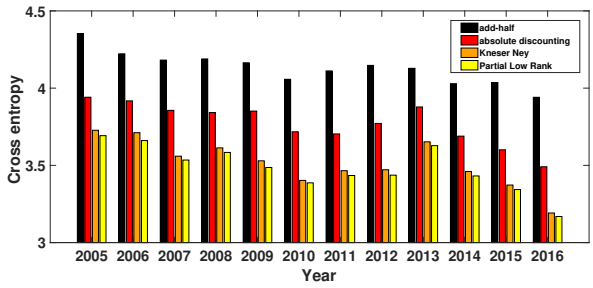
D Additional Experiments

D.1 Non-temporal data split

We perform the target type prediction, with the difference that the data split between train (used in forming the estimator) and test is by a random permutation and split, rather than temporal. Put differently, we randomly select 83% of the data pairs to be used in forming the estimator and the rest is used for testing. Figures 4(b) and 4(a) are equivalent to the ones in Figure 3 for the non-temporal data split case. The results are consistent with the ones in 6.



(a) All data



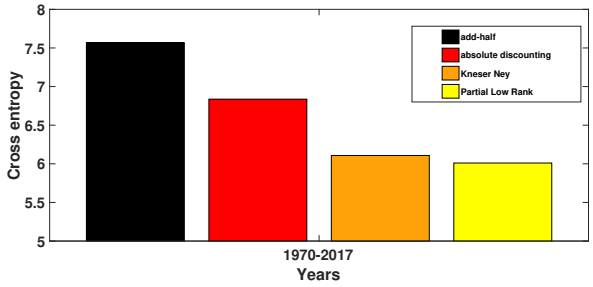
(b) Yearly data

Figure 4: Test cross entropy for a non-temporal split of (a) the whole data, and (b) a year's data

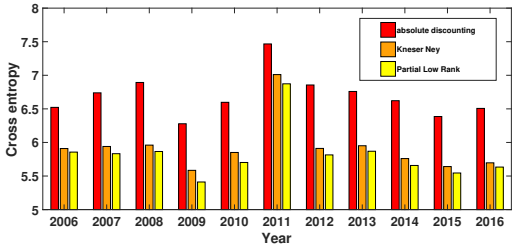
D.2 Joint prediction of target and weapon types on GDT

We repeat our experiments on a slightly modified prediction task. Instead of just predicting the target type for the next attack, we simultaneously predict both the target and weapon types in each city. Weapon type is a field recorded in the GDT and contains information about the weapon used in an attack. This joint prediction task has a bigger output space (around 3000).

Figures 5(a) and 5(b) show the benefit of using data from different cities (different rows of the count matrix) when predicting the target and weapon types in a particular city. Similar to the experiments mentioned in Section 6, PLR and KN always have significantly better performances in predicting the next incident's attack and weapon types than row-wise estimators such as add-half and absolute discounting.



(a) All data



(b) Yearly data

Figure 5: Test cross entropy for the joint prediction of target and weapon types when (a) trained on data prior to 2005 and tested on dates after that, and (b) trained on the first 10 months of a year and tested on the last two months