# N-gram Smoothing on Expected Fractional Counts

Michael Levit, Partha Parthasarathy, Shawn Chang

Microsoft Corporation

Technical Report #MSR-TR-2018-16

June 9, 2018

**Abstract**

This is a companion document to our Interspeech 2018 paper "What to Expect from Expected Kneser-Ney Smoothing" [1]. It provides additional details and considerations regarding derivation, optimizations and extensions of the n-gram smoothing on expected fractional counts, a technique originally introduced in [5].

## 1 Kneser-Ney Smoothing Recap

The goal of n-gram language modeling is to produce smoothed probability estimates for words in contexts. Let $\boldsymbol{u}$ be the context and $w$ the next word. Also, let $c(\boldsymbol{u}w) = \#\boldsymbol{u}w$ denote the number of observed occurrences of a n-gram. The maximum likelihood probability of $w$ given $\boldsymbol{u}$ is defined as:

$$p^{\mathrm{ML}}(w|\,\boldsymbol{u}) = \frac{c(\boldsymbol{u}w)}{c(\boldsymbol{u})} = \frac{c(\boldsymbol{u}w)}{c(\boldsymbol{u}\,\bullet)} = \frac{c(\boldsymbol{u}w)}{\sum_w c(\boldsymbol{u}w)} \tag{1}$$

The formula has an obvious problem with counts of n-grams not observed in training material. Count smoothing mitigates this problem. For each $\boldsymbol{u}$, it reduces counts of all or some observed n-grams $\boldsymbol{u}w$ and redistributes the saved amount among the unobserved or all n-grams. While counts get redistributed, their cumulative is preserved unchanged:

$$\tilde{p}(w|\,\boldsymbol{u}) = \frac{\tilde{c}(\boldsymbol{u}w)}{\tilde{c}(\boldsymbol{u}\,\bullet)} = \frac{\tilde{c}(\boldsymbol{u}w)}{c(\boldsymbol{u}\,\bullet)} = \frac{\tilde{c}(\boldsymbol{u}w)}{\sum_w c(\boldsymbol{u}w)} \tag{2}$$

In absolute discounting [2] the amount by which n-gram counts get reduced is a constant $D$, which results in:

$$\tilde{c}(\boldsymbol{u}w) = \begin{cases} c(\boldsymbol{u}w) - D, & \text{if } c(\boldsymbol{u}w) > 0 \\ n_{1+}(\boldsymbol{u}\,\bullet) D q_{\boldsymbol{u}}(w) & \text{otherwise} \end{cases} \tag{3}$$

where $n_{1+}(\boldsymbol{u}\,\bullet)$ is the number of possible words $\bullet = w$ such that $c(\boldsymbol{u}w) \geq 1$. Basically, we subtract a constant amount from all counts that are larger than this amount[1] and distribute the saved probability mass among all $\boldsymbol{u}w$ that weren't so lucky in a way that depends on $\boldsymbol{u}$ via distribution $q_{\boldsymbol{u}}(w)$.

Following Eq. 1, this discounting can also be expressed in terms of probabilities:

$$\tilde{p}(w|\,\boldsymbol{u}) = \begin{cases} \frac{c(\boldsymbol{u}w) - D}{c(\boldsymbol{u}\,\bullet)}, & \text{if } c(\boldsymbol{u}w) > 0 \\ \frac{n_{1+}(\boldsymbol{u}\,\bullet) D q_{\boldsymbol{u}}(w)}{c(\boldsymbol{u}\,\bullet)} & \text{otherwise.} \end{cases} \tag{4}$$

Context-independent constant $D$ can be LOO-estimated as

$$D = \frac{n_1}{n_1 + 2n_2}. \tag{5}$$

where $n_r$ is the number of n-grams in the corpus that occurred exactly $r$ times [3]. Alternatively, a modified version [4] allows $D$ to depend on $c(\boldsymbol{u}w) \in \{1, 2, 3+\}$.

Kneser-Ney (KN) smoothing offers a specialization of $q_{\boldsymbol{u}}$ making it proportional to a lower-order probability $p^{\text{kn}}$ defined below. Assuming $\boldsymbol{u'}$ to be the largest true suffix of $\boldsymbol{u}$ ($|\,\boldsymbol{u'}\,| = |\,\boldsymbol{u}\,| - 1$), this proportionality requirement becomes:

$$q_{\boldsymbol{u}}(w) = q'(\boldsymbol{u}) p^{\text{kn}}(w|\boldsymbol{u}'). \tag{6}$$

Then, probability $p^{\text{kn}}$ is estimated from marginal constraints over the oldest word in $\boldsymbol{u}$ [3] as:

$$p^{\text{kn}}(w|\,\boldsymbol{u'}) = \frac{n_{1+}(\bullet\,\boldsymbol{u'}w)}{n_{1+}(\bullet\,\boldsymbol{u'}\,\bullet)}. \tag{7}$$

[1]In the setup with integer counts, this means we subtract some $D < 1$ from all n-grams $\boldsymbol{u}w$ that occur at least once

With that, Eq. 3 can be rewritten like this:

$$\tilde{c}(\boldsymbol{u}w) = \begin{cases} c(\boldsymbol{u}w) - D, & \text{if } c(\boldsymbol{u}w) > 0 \\ \gamma'(\boldsymbol{u}) p^{\text{kn}}(w|\,\boldsymbol{u'}) & \text{otherwise} \end{cases} \tag{8}$$

and Eq. 4 turns into:

$$\tilde{p}(w|\,\boldsymbol{u}) = \begin{cases} \frac{c(\boldsymbol{u}w) - D}{c(\boldsymbol{u}\,\bullet)}, & \text{if } c(\boldsymbol{u}w) > 0 \\ \gamma(\boldsymbol{u}) p^{\text{kn}}(w|\,\boldsymbol{u'}) & \text{otherwise} \end{cases} \tag{9}$$

where relationship between $\gamma'(\boldsymbol{u})$ and $\gamma(\boldsymbol{u})$ is simply:

$$\gamma(\boldsymbol{u}) = \frac{\gamma'(\boldsymbol{u})}{c(\boldsymbol{u}\,\bullet)} = \frac{n_{1+}(\boldsymbol{u}\,\bullet) D q'(\boldsymbol{u})}{c(\boldsymbol{u}\,\bullet)}. \tag{10}$$

Instead of brute-force calculations, the context-dependent factor $\gamma(\boldsymbol{u})$ can be obtained via normalization:

$$\sum_{w:c(\boldsymbol{u}w)>0} \tilde{p}(w|\,\boldsymbol{u}) + \sum_{w:c(\boldsymbol{u}w)=0} \tilde{p}(w|\,\boldsymbol{u}) = 1 \tag{11}$$

under the closed vocabulary assumption.

Taking a closer look at Eq. 7, we can treat its numerator $n_{1+}(\bullet\,\boldsymbol{u'}w)$ as a pseudo-count of lower-order n-gram $\boldsymbol{u'}w$. In its turn, the denominator is by construction the sum of these pseudo-counts over $w$. Thus:

$$\begin{array}{lcl} c^{\text{kn}}(\boldsymbol{u'}w) & := & n_{1+}(\bullet\,\boldsymbol{u'}w) \\ c^{\text{kn}}(\boldsymbol{u'}\,\bullet) = \sum_w c^{\text{kn}}(\boldsymbol{u'}w) & = & \sum_w n_{1+}(\bullet\,\boldsymbol{u'}w) = n_{1+}(\bullet\,\boldsymbol{u'}\,\bullet) \end{array} \tag{12}$$

These pseudo-counts can be discounted as well [4], leading to a discounted version of the lower-order probabilities $p^{\text{kn}}(w|\boldsymbol{u'})$:

$$\tilde{p}^{\text{kn}}(w|\,\boldsymbol{u'}) = \begin{cases} \frac{c^{\text{kn}}(\boldsymbol{u'}w) - D}{c(\boldsymbol{u'}\,\bullet)}, & \text{if } c^{\text{kn}}(\boldsymbol{u'}w) > 0 \\ \gamma(\boldsymbol{u'}) p^{\text{kn}}(w|\,\boldsymbol{u''}) & \text{otherwise} \end{cases} \tag{13}$$

where $\boldsymbol{u''}$ is the right suffix of length $|\,\boldsymbol{u''}\,| = |\,\boldsymbol{u'}\,| - 1 = |\,\boldsymbol{u}\,| - 2$. The recursion can go on all the way to unigrams where unigram KN probability Eq. 7 is estimated as:

$$p^{\text{kn}}(w) = \frac{n_{1+}(\bullet w)}{n_{1+}(\bullet\bullet)} \tag{14}$$

and the saved probability mass due to discounting can be either distributed using trivial fall-back probabilities of zerograms or dedicated unknown symbol <unk>, or spread equally (cf. Section 3.5).

Now, let us combine Eq. 7 and Eq. 13 into a single recursive formula for better readability:

$$\tilde{p}(w|\boldsymbol{u}) = \begin{cases} \frac{\hat{c}(\boldsymbol{u}w)-D}{\hat{c}(\boldsymbol{u}\bullet)}, & \text{if } \hat{c}(\boldsymbol{u}w) > 0 \\ \gamma(\boldsymbol{u}) * \tilde{p}(w|\boldsymbol{u'}) & \text{otherwise} \end{cases} \tag{15}$$

with

$$\hat{c}(\boldsymbol{u}w) = \begin{cases} c(\boldsymbol{u}w) = \#(\boldsymbol{u}w), & \text{if } |\,\boldsymbol{u}w\,| = N \text{ or } \boldsymbol{u}w = \text{"<s>}\ldots\text{"} \\ c^{\text{kn}}(\boldsymbol{u}w) = n_{1+}(\bullet\,\boldsymbol{u}w), & \text{otherwise} \end{cases} \tag{16}$$

In practice, a separate constant $D^{(n)}$ (or, in the case of modified KN, 3-vector of constants) can be estimated and used for each n-gram order $n$.

## 2 Kneser-Ney on Expected Counts

In the case of Expected KN, the data is viewed as a collection of n-grams with probabilities that can be used to generate one or many "real" corpora with integer counts [5]. Therefore, all the statistics needed for KN-smoothing can be computed from the expectations on the population of such "real" corpora. Taking expectation of Eq. 2 over observed frequency $c(\boldsymbol{u}w) = r$ and noticing that conditional probabilities are independent of full counts:

$$\tilde{p}(w|\boldsymbol{u}) = \frac{E[\tilde{c}(\boldsymbol{u}w)]}{E[c(\boldsymbol{u}\,\bullet)]} = \frac{E[\tilde{c}(\boldsymbol{u}w)]}{\sum_w E[c(\boldsymbol{u}w)]}. \tag{17}$$

The discounted counts that we used to compute via Eq. 8 for regular KN-smoothing, now need to be turned into expectations as well:

$$\begin{aligned} E[\tilde{c}(\boldsymbol{u}w)] &= \sum_{r>0} p(c(\boldsymbol{u}w)\!=\!r)(r-D) + p(c(\boldsymbol{u}w)\!=\!0)\gamma'(\boldsymbol{u})p^{\text{kn}}(w|\boldsymbol{u'}) \\ &= E[c(\boldsymbol{u}w)] - DP + p(c(\boldsymbol{u}w)\!=\!0)\gamma'(\boldsymbol{u})p^{\text{kn}}(w|\boldsymbol{u'}) \end{aligned} \tag{18}$$

Eq. 18 is just an expanded form of an expectation formula, weighted sum of several cases: if $\boldsymbol{u}w$ occurred one or more times, use the discounted version, otherwise fall back to $p^{\text{kn}}$. For the last transition, average discount $DP$ was introduced as

$$DP := p(c(\boldsymbol{u}w) > 0) * D \tag{19}$$

In the modified version of Expected KN, this term becomes:

$$DP := p(c(\boldsymbol{u}w) = 1) * D_1 + p(c(\boldsymbol{u}w) = 2) * D_2 + p(c(\boldsymbol{u}w) \geq 3) * D_3 \tag{20}$$

where constants $D_r$ correspond to occurrence counts $c(\boldsymbol{u}w) = r$. Substituting Eq. 18 into Eq. 17:

$$\tilde{p}(w|\,\boldsymbol{u}) = \frac{E[c(\boldsymbol{u}w)] - DP}{\sum_w E[c(\boldsymbol{u}w)]} + p(c(\boldsymbol{u}w) = 0)\gamma(\boldsymbol{u})p^{\text{kn}}(w|\,\boldsymbol{u'}) \tag{21}$$

We will return to the normalization coefficients $\gamma(\boldsymbol{u})$ later. For now though, let us focus on the KN-probability $p^{\text{kn}}(w|\,\boldsymbol{u'})$. Similar to Eq. 7 for integer-count KN, this lower-order probability is defined in terms of numbers of words $w$ for which n-grams $\boldsymbol{u'}w$ occurred at least once, except this time, we need to use expectations [5]:

$$p^{\text{kn}}(w|\,\boldsymbol{u'}) := \frac{E[n_{1+}(\bullet\,\boldsymbol{u'}w)]}{E[n_{1+}(\bullet\,\boldsymbol{u'}\,\bullet)]} \tag{22}$$

Now let us compute all the other statistics needed for Eqs. 19, 20, 21 and 22, such as $D_r$, $E[c(\boldsymbol{u}w)]$, $p(c(\boldsymbol{u}w) = r)$ and $p(c(\boldsymbol{u}w) > r)$ as well as $E[n_{1+}(\bullet\,\boldsymbol{u'}w)]$ and $E[n_{1+}(\bullet\,\boldsymbol{u'}\,\bullet)]$.

First, assume that all n-grams come with weights $0 < p \leq 1$. As mentioned in the beginning of the section, instead of dealing with a single training set of n-grams, we pretend to be given a distribution of training sets where each n-gram is provided with a probability. In a particular instance of training set drawn from this distribution, the n-gram might or might not occur (its $c(\boldsymbol{u}w)$ could be 1 or 0). The expected number of its occurrences in the distribution of training sets will obviously be $p$.

Next, acknowledge that a n-gram $\boldsymbol{u}w$ can occur not once but $K > 1$ times; in the most general case, each occurrence will have its own probability (weight) $p_k, k = \overline{1,K}$[2]. The expected value of the cumulative frequency of the n-gram in a set drawn from the training set distribution is

$$E[c(\boldsymbol{u}w)] = \sum_{k=1}^{K} p_k. \tag{23}$$

[2]It should also be understood that sets $\{p_k\}$ are different for each n-gram

Probabilities of specific cumulative frequencies can be computed recursively:

$$p(c(\boldsymbol{u}w)=r) = s_K^r \tag{24}$$

where

$$s_k^r = \begin{cases} s_{k-1}^r(1-p_k) + s_{k-1}^{r-1}p_k, & \text{if } 0 \leq r \leq k \\ 1, & \text{if } k = r = 0 \\ 0, & \text{otherwise.} \end{cases} . \tag{25}$$

Note that since we will only need these quantities for $r \leq 4$, the overall computational effort can be considered linear in $K$.

The expected value of the number of all n-grams with cumulative frequency $r$ is computed by simply summing over all n-grams:

$$E[n_r] = \sum_{\boldsymbol{u}w} p(c(\boldsymbol{u}w)=r) \tag{26}$$

which, in its turn, allows us to define the expectation-version of the absolute discount $D$ [5]:

$$D = \frac{E[n_1]}{E[n_1] + 2E[n_2]}, \tag{27}$$

or a set thereof, for the modified Expected KN:

$$\begin{aligned} D_1 &= 1 - 2D\tfrac{E[n_2]}{E[n_1]} \\ D_2 &= 2 - 3D\tfrac{E[n_3]}{E[n_2]} \\ D_3 &= 3 - 4D\tfrac{E[n_4]}{E[n_3]} \end{aligned} \tag{28}$$

Similar to Eq. 26, we can also define marginals:

$$E[n_r(\boldsymbol{u}\,\bullet)] = \sum_{w} p(c(\boldsymbol{u}w)=r) \tag{29}$$

$$E[n_r(\bullet\,\boldsymbol{u'}w)] = \sum_{v} p(c(v\,\boldsymbol{u'}w)=r) \tag{30}$$

One-sided probabilities are similar to Eq. 24 but need to be protected against rounding errors:

$$p(c(\boldsymbol{u}w)\geq r) = \max\left\{ s_K^r, 1 - \sum_{r'<r} s_K^{r'} \right\} \tag{31}$$

and the corresponding marginals:

$$E[n_{r+}(\boldsymbol{u}\,\bullet)] = \sum_w p(c(\boldsymbol{u}w) \geq r) \tag{32}$$

$$E[n_{r+}(\bullet\,\boldsymbol{u'}w)] = \sum_v p(c(v\,\boldsymbol{u'}w) \geq r) \tag{33}$$

For recursive formulation of adjusted probability in Eq. 15, we treated numerator and denominator of Eq. 7 as KN-counts. Similarly, we can treat numerator and denominator in Eq. 22 as expectations of KN-counts:

$$p^{\text{kn}}(w|\,\boldsymbol{u'}) = \frac{E[c^{\text{kn}}(\boldsymbol{u'}w)]}{E[c^{\text{kn}}(\boldsymbol{u'}\,\bullet)]} \tag{34}$$

with

$$\begin{aligned} E[c^{\text{kn}}(\boldsymbol{u'}w)] &= E[n_{1+}(\bullet\,\boldsymbol{u'}w)] = \textstyle\sum_v p(c(v\,\boldsymbol{u'}w) > 0) \\ E[c^{\text{kn}}(\boldsymbol{u'}\,\bullet)] &= \textstyle\sum_w E[c^{\text{kn}}(\boldsymbol{u'}w)] = \sum_w \sum_v p(c(v\,\boldsymbol{u'}w) > 0) \end{aligned} \tag{35}$$

It is easy to see that regular and KN counts $c(\boldsymbol{u}w)$ and $c^{\text{kn}}(\boldsymbol{u}w)$ can only be zero at the same time. Therefore, probabilities of either count being zero are equal:

$$p^{\text{kn}}(c(\boldsymbol{u}w)=0) = p(c(\boldsymbol{u}w)=0) = s_K^0. \tag{36}$$

Hence, the recursive version of Eq. 21 can be written as:

$$\tilde{p}(w|\,\boldsymbol{u}) := \frac{E[\hat{c}(\boldsymbol{u}w)] - DP}{\sum_w E[\hat{c}(\boldsymbol{u}w)]} + p(\hat{c}(\boldsymbol{u}w)=0)\gamma(\boldsymbol{u})\tilde{p}(w|\,\boldsymbol{u'}) \tag{37}$$

with

$$E[\hat{c}(\boldsymbol{u}w)] = \begin{cases} E[c(\boldsymbol{u}w)] = \sum_{k=1}^K p_k \text{ (Eq. 23)}, & \text{if } |\,\boldsymbol{u}w\,| = N \\ & \text{or } \boldsymbol{u}w = \text{"<s>} \ldots\text{"} \\ E[c^{\text{kn}}(\boldsymbol{u}w)] = \sum_v p(c(v\,\boldsymbol{u}w) > 0) \text{ (Eq. 35)} & \text{otherwise} \end{cases} \tag{38}$$

and $p(\hat{c}(\boldsymbol{u}w) = 0) = s_K^0$, due to Eqs. 24 and 36. Again, we can decide to estimate separate absolute discounts $DP^{(n)}$ for each n-gram order $n = |\,\boldsymbol{u}w\,|$, though for simplicity, we will skip this index in most of the formulae below.

At last, just like it was done for regular KN-smoothing in Eq. 11, we compute normalization coefficients $\gamma(\boldsymbol{u})$ for each $\boldsymbol{u}$ from stochastic conditions on probabilities (sum over $w$ is 1.0).

$$\gamma(\boldsymbol{u}) = \frac{1 - \sum_w \frac{E[\hat{c}(\boldsymbol{u}w)] - DP}{\sum_v E[\hat{c}(\boldsymbol{u}\,v)]}}{\sum_w p(\hat{c}(\boldsymbol{u}w)=0) * \tilde{p}(w|\,\boldsymbol{u'})} \tag{39}$$

If the language model is to be built without n-gram count cutoffs, the above formula can be simplified:

$$\gamma(\boldsymbol{u}) = \frac{\sum_w DP}{\sum_w E[\hat{c}(\boldsymbol{u}w)] * \sum_w p(\hat{c}(\boldsymbol{u}w)=0)\tilde{p}(w|\,\boldsymbol{u'})} \tag{40}$$

(for impact of cutoffs, see Section 3.6). According to that, summation has to be conducted over all words $w$ in the vocabulary for all contexts $\boldsymbol{u}$. This is often infeasible for realistic models with  100M n-grams and vocabulary size on the order of millions. The good news is that only n-grams $\boldsymbol{u}w$ that actually occur in the training data get to contribute to the numerator sum and the first denominator sum, but what about the other denominator sum: $\sum_w p(\hat{c}(\boldsymbol{u}w)=0)\tilde{p}(w|\,\boldsymbol{u'})$? This sum can be split into two: one for $w$'s such that $\boldsymbol{u}w$ exists in the training data, the other for the ones where it doesn't. The first sub-sum is computed as prescribed, the addends of the second all have in common that their $p(\hat{c}(\boldsymbol{u}w)=0)$ is always 1.0. Therefore, the second sub-sum is just: $\sum_{w:c(\boldsymbol{u}w)=0} \tilde{p}(w|\,\boldsymbol{u'})$. On the other hand, due to stochastic conditions on the lower-order probabilities $\tilde{p}(w|\,\boldsymbol{u'})$:

$$\sum_{w:c(\boldsymbol{u}w)=0} \tilde{p}(w|\,\boldsymbol{u'}) = 1 - \sum_{w:c(\boldsymbol{u}w)>0} \tilde{p}(w|\,\boldsymbol{u'}) \tag{41}$$

and so we are back to just the observed n-grams. Then, the entire sum becomes:

$$\begin{aligned} \sum_w p(\hat{c}(\boldsymbol{u}w)=0)\tilde{p}(w|\,\boldsymbol{u'}) &= \\ \sum_{w:c(\boldsymbol{u}w)>0)} p(\hat{c}(\boldsymbol{u}w)=0)\tilde{p}(w|\,\boldsymbol{u'}) + 1 - \sum_{w:c(\boldsymbol{u}w)>0} \tilde{p}(w|\,\boldsymbol{u'}) &= \\ 1 + \sum_{w:c(\boldsymbol{u}w)>0)} (\hat{p}(c(\boldsymbol{u}w)=0) - 1)\tilde{p}(w|\,\boldsymbol{u'}) & \end{aligned} \tag{42}$$

and a more practical version of Eq. 39:

$$\gamma(\boldsymbol{u}) = \frac{\sum_{w:c(\boldsymbol{u}w)>0} DP}{\sum_{w:c(\boldsymbol{u}w)>0} E[\hat{c}(\boldsymbol{u}w)] * \left(1 - \sum_{w:c(\boldsymbol{u}w)>0}(1 - p(\hat{c}(\boldsymbol{u}w)=0))\tilde{p}(w|\,\boldsymbol{u'})\right)} \tag{43}$$

We now have everything to compute discounted probabilities.

## 3 Practical Considerations

In this section several practical topics of working with Expected KN are addressed.

### 3.1 Back-off LM

Formula 37 can be easily split into two cases:

- "Out of Training" case where n-gram $\boldsymbol{u}w$ is not present in the training material at all. This means $p(c(\boldsymbol{u}w) = 0) = 1.0$ and only a simplified version of the second addend survives
- the most general case where formula 37 is used in its original form.

This means that the model defines probability of $w$ following $\boldsymbol{u}$ as:

$$\begin{cases} \tilde{p}(w|\,\boldsymbol{u}) & \text{(from 37)} \quad \text{if } \boldsymbol{u}w \text{ observed in input data set} \\ \gamma(\boldsymbol{u}) * \tilde{p}(w|\,\boldsymbol{u'}) & \text{otherwise} \end{cases} \tag{44}$$

In Eq. 44, it is easy to recognize Katz back-off model with adjusted n-gram probability $\tilde{p}(w|\,\boldsymbol{u})$ and back-off penalty $\gamma(\boldsymbol{u})$, which means that the Expected KN smoothing can be saved in the familiar ARPA format.

### 3.2 Implementation

Let's summarize the algorithm to compute discounted probabilities $\tilde{p}(w|\,\boldsymbol{u})$ and $\gamma(\boldsymbol{u})$.

1. assume for now that we have already accumulated statistics $s_K^r$ for $r = \overline{0,4}$ for each n-gram $\boldsymbol{u}w$
2. compute occurrence probabilities $p(c(\boldsymbol{u}w) = r)$ (Eq. 24), $p(c(\boldsymbol{u}w) \geq r)$ (Eq. 31) and $p(c(\boldsymbol{u}w) > 0) = 1 - p(c(\boldsymbol{u}w) = 0)$. Also compute count expectations $E[c(\boldsymbol{u}w)]$ (Eq. 23)
3. compute Expected KN-counts $E[c^{\text{kn}}(\boldsymbol{u'}w)]$ using Eq. 35
4. estimate per-n-gram-order frequency expectations $E[n_r]$ with Eq. 26 and absolute discounts *DP* (or $DP^{(n)}$) with Eqs. 19 and 27 for regular Expected KN and Eqs. 20 and 28 for the modified version.

5. recursively, starting with n-gram order n=1 and increasing to full order $N$, do the following:
   (a) compute count expectations for each observed n-gram $\boldsymbol{u}w$ using Eq. 38 (while relying on statistics precomputed in Steps 2 or 3)
   (b) estimate $\gamma(\boldsymbol{u})$ via Eq. 43 (assuming that the lower-order discounted KN-probabilities $\tilde{p}(w|\,\boldsymbol{u'})$ are already available from the previous iteration)
   (c) compute discounted version of KN-probability $\tilde{p}(w|\,\boldsymbol{u})$ using Eq. 37

## 3.3 Repeated Patterns

Now, let us focus on statistics $s_K^r$ accumulated individually for each n-gram. These statistics can be computed in a single pass over the training corpus. As we advance in the training data set and encounter new instances of an n-gram, we can use Eq. 25 to update five probabilities $s^r$ of observing this n-gram $r = \overline{0,4}$ times in a "real" (non-fractional) corpus sampled from probabilities defined by these instances. However, a significant speed-up can be achieved if "compact" representations of the n-grams are provided in format:

*n-gram* *probability* *#repetitions*

For instance, let us estimate all $s^r$ for unigram "hello" from this data:

| | | |
|---|---|---|
| hello world | 0.8 | 14 |
| hello world | 0.3 | 9 |
| hello baby | 0.9 | 11 |

Having only seen the first sentence (probability $p = 0.8$ and number of repetitions $m = 14$) we can already generate a sample of a training corpus. The probability of not observing "hello" a single time in this sample is $s_0 = (1-0.8)^{14}$ (none of the 14 coin tosses with success probability 0.8 actually succeeded). By comparison, observing this n-gram exactly once means that only one coin toss succeeded and the rest failed. The general formula is k-combination:

$$\begin{aligned} s_m^0 &= (1-p)^m \\ s_m^1 &= m(1-p)^{m-1} * p \\ s_m^2 &= \frac{m(m-1)}{2}(1-p)^{m-2} * p^2 \\ s_m^3 &= \frac{m(m-1)(m-2)}{6}(1-p)^{m-3} * p^3 \\ s_m^4 &= \frac{m(m-1)(m-2)(m-3)}{24}(1-p)^{m-4} * p^4 \end{aligned} \tag{45}$$

and remember that we only need to compute these statistics for $r \leq 4$. As we take into consideration the second sentence, probability $s^0$ of not seeing "hello" in the corpus generated from these two sentences is the probability of generating this n-gram neither from the $m_1 = 14$ occurrences of probability $p_1 = 0.8$ nor from $m_2 = 9$ occurrences of probability $p_2 = 0.3$. Summarizing updates for all five $s^r$:

$$\begin{aligned} s_{m_1+m_2}^0 &= s_{m_1}^0 * s_{m_2}^0 \\ s_{m_1+m_2}^1 &= s_{m_1}^1 * s_{m_2}^0 + s_{m_1}^0 * s_{m_2}^1 \\ s_{m_1+m_2}^2 &= s_{m_1}^2 * s_{m_2}^0 + s_{m_1}^1 * s_{m_2}^1 + s_{m_1}^0 * s_{m_2}^2 \\ s_{m_1+m_2}^3 &= s_{m_1}^3 * s_{m_2}^0 + s_{m_1}^2 * s_{m_2}^1 + s_{m_1}^1 * s_{m_2}^2 + s_{m_1}^0 * s_{m_2}^3 \\ s_{m_1+m_2}^4 &= s_{m_1}^4 * s_{m_2}^0 + s_{m_1}^3 * s_{m_2}^1 + s_{m_1}^2 * s_{m_2}^2 + s_{m_1}^1 * s_{m_2}^3 + s_{m_1}^0 * s_{m_2}^4 \end{aligned} \tag{46}$$

In exactly the same way we continue updating running values $s_{m_1+m_2+m_3\ldots}^r$ for all other input sentences containing "hello". As a result, the overall complexity is linear in the number of unique n-gram/probability combinations.

## 3.4 Input as a Number of Weighted Alternatives

One of the most common reasons for fractional counts is training on uncertain data. The uncertainty can be expressed via weighted graphs (e.g. speech lattices) or a number of linear alternatives (ASR n-best). What makes this setup interesting is that n-grams from alternative paths (explicit or through graph) cannot be considered independently. For instance, if there are two alternatives with weights:

| | |
|---|---|
| hello world | 0.8 |
| hello baby | 0.2 |

they do not contribute to the unigram “hello” twice (once with p=0.8 and another time with p=0.2), but rather a single time with p=1.0. The difference is significant: in the first version there is a positive chance that the sample corpus drawn from the training data will have no “hello” at all. In the second, this chance is nil. Therefore, probabilities of each n-grams in all paths need to be accumulated (sometimes, for convenience we need to assume that each path contains at most one). This is, for instance, how Good-Turing approach on uncertain data works in [6].

In this report, we are only concerned with a simplified setup with a number of discrete linear alternatives for each sentence weighted by their posteriors. In that case, one approximate solution is to merge these into a single word trie while accumulating posteriors on each arc[3]. Then, all paths of length up to $N$ from the root of this trie can be extracted with weights computed as products of probabilities along each path.

## 3.5 Dealing with the Unknown

Language models can contain a token explicitly representing all unknown words. Usually, this token is only present as a unigram and never occurs in higher order n-grams. Therefore, when computing discounted probability $\tilde{p}(w|\boldsymbol{u})$, $\boldsymbol{u}$ is truncated (from the “older” side) to contain only known words without incurring any penalty. On the other hand, if $w$ itself is unknown, this probability automatically turns into unigram probability $p_{\text{unk}} = p(\text{<unk>})$. If this probability is very small, its effect is negligible. However, for larger $p_{\text{unk}}$, the already discounted n-gram probability estimates $\tilde{p}(w|\boldsymbol{u})$ from Eq. 37 need to be further reduced to accommodate for an extra addend in the stochastic condition.

The easiest way to do this is by computing $\gamma(\boldsymbol{u})$ and $\tilde{p}(w|\boldsymbol{u})$ just like we did before in Eqs. 37, 43 and 44, and then retrospectively renormalize the probabilities to sum up to $1 - p_{\text{unk}}$:

$$\tilde{p}'(w|\boldsymbol{u}) = (1 - p_{\text{unk}})\tilde{p}(w|\boldsymbol{u}) \tag{47}$$

$$\gamma'(\boldsymbol{u}) = (1 - p_{\text{unk}})\tilde{\gamma}(\boldsymbol{u}) \tag{48}$$

[3]Compare this with the proper way where forward/backward statistics from the full graph would be considered.

Next, just like with higher orders, unigram probability estimates need to be discounted as:

$$\frac{E[\hat{c}(\boldsymbol{u}w)] - DP}{\sum_w E[\hat{c}(\boldsymbol{u}w)]} \tag{49}$$

However, KN smoothing does not offer a clear guidance as to how to redistribute spared unigram probability mass in this case. If we want to continue rely on Eq. 37, some kind of replacement for zerogram KN-probability needs to be agreed upon. For instance, the spared probability can be redistributed among all unigrams equally (SRILM approach) or according to the ML probability estimates (which amounts to doing no smoothing at all), or following some compromise solution. For instance, in our experiments we found it advantageous to apply square root flattening to ML unigram probability estimates for this purpose.

## 3.6 N-gram Frequency Cuttoffs

Oftentimes, minimum frequency requirements (cutoffs) are imposed on n-grams while estimating language models. Together with entropy-based pruning, cutoffs are useful to keep in check language model size. Specifically for KN-smoothing, such cutoffs appear to be working better when applied to KN-counts rather than the original counts [7]. For Expected KN, the thresholds can be real-valued.

What is the effect of cutoffs on the formulae we devised earlier and, specifically, on how these formulae can be applied in practice? For instance, in the numerator of Eq. 39, we assumed the sum over all present n-grams $\boldsymbol{u}v$ to be the same as the sum over all present and considered n-grams $\boldsymbol{u}w$, and this allowed to simplify this formula into Eq. 40. With cutoffs, this simplification does not work, and as a result an additional quantity needs to be maintained for each context:

$$\hat{C}(\boldsymbol{u}) := \sum_w \hat{c}(\boldsymbol{u}w). \tag{50}$$

For regular counts, values $\hat{C}(\boldsymbol{u})$ are identical to counts $c(\boldsymbol{u})$, and for KN-counts, they are the sum of KN-counts over all $\boldsymbol{u}w$ before cutoff thresholds are applied. Going over to expectations, Eq. 39 can be rewritten as:

$$\gamma(\boldsymbol{u}) = \frac{1 - \sum_w \frac{E[\hat{c}(\boldsymbol{u}w)] - DP}{E[\hat{C}(\boldsymbol{u})]}}{\sum_w p(\hat{c}(\boldsymbol{u}w) = 0) * \tilde{p}(w|\, \boldsymbol{u'})} = \frac{E[\hat{C}(\boldsymbol{u})] - \sum_w (E[\hat{c}(\boldsymbol{u}w)] - DP)}{E[\hat{C}(\boldsymbol{u})] * \sum_w p(\hat{c}(\boldsymbol{u}w) = 0) \tilde{p}(w|\, \boldsymbol{u'})} \tag{51}$$

Since summation happens only over the n-grams that survived cutoff application, the difference to Eq. 39 is significant ($E[\hat{C}(\boldsymbol{u})] \neq \sum_w E[\hat{c}(\boldsymbol{u}w)]$). Other variants of Eq. 39 need to be adjusted accordingly as well. For instance, Eq. 43 turns into:

$$\gamma(\boldsymbol{u}) = \frac{E[\hat{C}(\boldsymbol{u})] - \sum_{w:c(\boldsymbol{u}w)>0}(E[\hat{c}(\boldsymbol{u}w)] - DP)}{E[\hat{C}(\boldsymbol{u})] * (1 - \sum_{w:c(\boldsymbol{u}w)>0}(1 - p(\hat{c}(\boldsymbol{u}w)=0))\tilde{p}(w|\,\boldsymbol{u'}))} \tag{52}$$

Similarly, the discounted probability from Eq. 37 is now

$$\tilde{p}(w|\,\boldsymbol{u}) := \frac{E[\hat{c}(\boldsymbol{u}w)] - DP}{E[\hat{C}(\boldsymbol{u})]} + p(\hat{c}(\boldsymbol{u}w)=0)\gamma(\boldsymbol{u})\tilde{p}(w|\,\boldsymbol{u'}) \tag{53}$$

## 4 Interpolated Modified Expected Kneser-Ney Smoothing

In [4] it was noted that interpolated version of the KN smoothing generally leads to better results. The difference to the back-off KN lies in distributing the saved probability mass among all words in the vocabulary and not just among the words that never occurred in the context $\boldsymbol{u}$. Not only does this version lead to better results, it also happens to be easier to implement: we just need to drop $p(\hat{c}(\boldsymbol{u}w)=0)$ from all formulae for $\gamma(\boldsymbol{u})$ and $\tilde{p}(w|\,\boldsymbol{u})$. For instance, instead of Eq. 53 we will have:

$$\tilde{p}(w|\,\boldsymbol{u}) := \frac{E[\hat{c}(\boldsymbol{u}w)] - DP}{E[\hat{C}(\boldsymbol{u})]} + \gamma(\boldsymbol{u})\tilde{p}(w|\,\boldsymbol{u'}) \tag{54}$$

and in place of Eq. 51, there will be

$$\gamma(\boldsymbol{u}) = \frac{E[\hat{C}(\boldsymbol{u})] - \sum_w(E[\hat{c}(\boldsymbol{u}w)] - DP)}{E[\hat{C}(\boldsymbol{u})] * \sum_w 1 * \tilde{p}(w|\,\boldsymbol{u'})} = \frac{E[\hat{C}(\boldsymbol{u})] - \sum_w(E[\hat{c}(\boldsymbol{u}w)] - DP)}{E[\hat{C}(\boldsymbol{u})]} \tag{55}$$

The interpolated version can also be expressed as a back-off language model (ARPA format) with discounted probability of an observed and preserved n-gram $\boldsymbol{u}w$ defined as:

$$\alpha(\boldsymbol{u}w) = \log_{10}(\tilde{p}(w|\,\boldsymbol{u}) + \gamma(\boldsymbol{u}) * \tilde{p}(w|\,\boldsymbol{u'})) \tag{56}$$

and the back-off penalty associated with context $\boldsymbol{u}$ being

$$\beta(\boldsymbol{u}) = \log_{10}(\gamma(\boldsymbol{u})) \tag{57}$$

## 5 Witten-Bell Smoothing Recap

In Witten-Bell Discounting, probability of falling back on the shortened context is the ML estimate of how often during training, previously unseen tokens had been observed following given history. The interpolation version is:

$$\tilde{p}(w|\boldsymbol{u}) = \frac{c(\boldsymbol{u}w)}{\sum_w c(\boldsymbol{u}w) + n_{1+}(\boldsymbol{u}\bullet)} + \frac{n_{1+}(\boldsymbol{u}\bullet)}{\sum_w c(\boldsymbol{u}w) + n_{1+}(\boldsymbol{u}\bullet)}\tilde{p}(w|\boldsymbol{u'}) \qquad (58)$$

and the back-off modification:

$$\tilde{p}(w|\boldsymbol{u}) = \begin{cases} \frac{c(\boldsymbol{u}w)}{\sum_w c(\boldsymbol{u}w)+n_{1+}(\boldsymbol{u}\bullet)}, & \text{if } c(\boldsymbol{u}w) > 0 \\ \gamma(\boldsymbol{u})\tilde{p}(w|\boldsymbol{u'}), & \text{otherwise} \end{cases} \qquad (59)$$

Summing over all words in the lexicon, realizing that cumulative probability of the observed tokens is just one minus probability of the unseen ones and, finally, taking probability of unknown into account:

$$1.0 = p_{\text{unk}} + \frac{\sum_w c(\boldsymbol{u}w)}{\sum_w c(\boldsymbol{u}w) + n_{1+}(\boldsymbol{u}\bullet)} + \gamma(\boldsymbol{u})(1.0 - p_{\text{unk}} - \sum_w \tilde{p}(w|\boldsymbol{u'})) \qquad (60)$$

with summation carried out over words $w$ that have been observed after $\boldsymbol{u}$. This leads to an estimate for $\gamma(\boldsymbol{u})$:

$$\gamma(\boldsymbol{u}) = \frac{\frac{n_{1+}(\boldsymbol{u}\bullet)}{\sum_w c(\boldsymbol{u}w)+n_{1+}(\boldsymbol{u}\bullet)} - p_{\text{unk}}}{1 - \sum_w \tilde{p}(w|\boldsymbol{u'}) - p_{\text{unk}}} \qquad (61)$$

## 6 Expected Witten-Bell Smoothing

Similarly to Expected KN, we can define conditional probability via expectations:

$$\begin{aligned} \tilde{p}(w|\boldsymbol{u}) &= \sum_{r>0} p(c(\boldsymbol{u}w) = r)\frac{c(\boldsymbol{u}w)}{\sum_w c(\boldsymbol{u}w) + n_{1+}(\boldsymbol{u}\bullet)} + p(c(\boldsymbol{u}w)=0)\gamma(\boldsymbol{u})\tilde{p}(w|\boldsymbol{u'}) \\ &= E\left[\frac{c(\boldsymbol{u}w)}{\sum_w c(\boldsymbol{u}w) + n_{1+}(\boldsymbol{u}\bullet)}\right] + p(c(\boldsymbol{u}w)=0)\gamma(\boldsymbol{u})\tilde{p}(w|\boldsymbol{u'}) \qquad (62) \end{aligned}$$

Unlike Expected Kneser-Ney, taking expectation over $c(\boldsymbol{u})$ under the fraction can only be considered a first order approximation [8]. In the absence of a better option, let's proceed with that approximation.

$$\tilde{p}(w|\,\boldsymbol{u}) \approx \frac{E[c(\boldsymbol{u}w)]}{\sum_w E[c(\boldsymbol{u}w)] + E[n_{1+}(\boldsymbol{u}\,\bullet)} + p(c(\boldsymbol{u}w)\!=\!0)\gamma(\boldsymbol{u})\tilde{p}(w|\,\boldsymbol{u'}) \qquad (63)$$

Summing up over all words $w$ in the vocabulary (and taking $p_{\text{unk}}$ into account):

$$1.0 = p_{\text{unk}} + \frac{\sum_w E[c(\boldsymbol{u}w)]}{\sum_w E[c(\boldsymbol{u}w)] + E[n_{1+}(\boldsymbol{u}\,\bullet)]} + \sum_w p(c(\boldsymbol{u}w)\!=\!0)\gamma(\boldsymbol{u})\tilde{p}(w|\,\boldsymbol{u'}) \quad (64)$$

Solving for $\gamma(\boldsymbol{u})$:

$$\gamma(\boldsymbol{u}) = \frac{1 - \frac{\sum_w E[c(\boldsymbol{u}w)]}{\sum_w E[c(\boldsymbol{u}w)] + E[n_{1+}(\boldsymbol{u}\,\bullet)]} - p_{\text{unk}}}{\sum_w p(c(\boldsymbol{u}w)\!=\!0)\tilde{p}(w|\,\boldsymbol{u'})} \qquad (65)$$

Next, following the same arguments we used in Eq. 42 (and not forgetting $p_{\text{unk}}$), we notice:

$$\sum_w p(\hat{c}(\boldsymbol{u}w)\!=\!0)\tilde{p}(w|\,\boldsymbol{u'}) = 1 - p_{\text{unk}} - \sum_{w:c(\boldsymbol{u}w)>0)} \tilde{p}(w|\,\boldsymbol{u'})(1 - \hat{p}(c(\boldsymbol{u}w)\!=\!0)) \quad (66)$$

Besides, since we are only dealing with ordinary counts, the sum $\sum_w E[c(\boldsymbol{u}w)]$ in the numerator that goes over all observed n-grams (incl.those that will be removed after cutoffs) can be replaced by $E[c(\boldsymbol{u})]$. With that:

$$\gamma(\boldsymbol{u}) = \frac{1 - \frac{\sum_{w:c(\boldsymbol{u}w)>0)} E[c(\boldsymbol{u}w)]}{E[c(\boldsymbol{u})] + E[n_{1+}(\boldsymbol{u}\,\bullet)]} - p_{\text{unk}}}{1 - p_{\text{unk}} - \sum_{w:c(\boldsymbol{u}w)>0)} \tilde{p}(w|\,\boldsymbol{u'})(1 - \hat{p}(c(\boldsymbol{u}w)\!=\!0))} \qquad (67)$$

## 7 Experiments and Results

For experimental evaluation of the above techniques please see our Interspeech paper [1].

## References

[1] Levit, Michael, and Parthasarathy, Sarangarajan, and Chang, Shuangyu. ”What to Expect from Expected Kneser-Ney Smoothing.” Interspeech 2018.

[2] Ney, Hermann, and Ute Essen. ”On smoothing techniques for bigram-based natural language modelling.” Acoustics, Speech, and Signal Processing, 1991. ICASSP-91., 1991 International Conference on. IEEE, 1991.

[3] Kneser, Reinhard, and Hermann Ney. ”Improved backing-off for m-gram language modeling.” Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on. Vol. 1. IEEE, 1995.

[4] Chen, Stanley F., and Joshua Goodman. ”An empirical study of smoothing techniques for language modeling.” Computer Speech & Language 13.4 (1999): 359-394.

[5] Zhang, Hui, and David Chiang. ”Kneser-Ney Smoothing on Expected Counts.” ACL (1). 2014.

[6] Kuznetsov, Vitaly, Hank Liao, Mehryar Mohri, Michael Riley, and Brian Roark ”Learning n-gram language models from uncertain data.” Proceedings of Interspeech (2016).

[7] SRILM: http://www.speech.sri.com/projects/srilm/

[8] http://www.stat.cmu.edu/h̃seltman/files/ratio.pdf