Kneser-Ney Smoothing on Expected Counts

Hui Zhang
Department of Computer Science
University of Southern California
hzhang@isi.edu

David Chiang
Information Sciences Institute
University of Southern California
chiang@isi.edu

Abstract

Widely used in speech and language processing, Kneser-Ney (KN) smoothing has consistently been shown to be one of the best-performing smoothing methods. However, KN smoothing assumes integer counts, limiting its potential uses—for example, inside Expectation-Maximization. In this paper, we propose a generalization of KN smoothing that operates on fractional counts, or, more precisely, on *distributions* over counts. We rederive all the steps of KN smoothing to operate on count distributions instead of integral counts, and apply it to two tasks where KN smoothing was not applicable before: one in language model adaptation, and the other in word alignment. In both cases, our method improves performance significantly.

1 Introduction

In speech and language processing, smoothing is essential to reduce overfitting, and Kneser-Ney (KN) smoothing (Kneser and Ney, 1995; Chen and Goodman, 1999) has consistently proven to be among the best-performing and most widely used methods. However, KN smoothing assumes integer counts, whereas in many NLP tasks, training instances appear with possibly fractional weights. Such cases have been noted for language modeling (Goodman, 2001; Goodman, 2004), domain adaptation (Tam and Schultz, 2008), grapheme-to-phoneme conversion (Bisani and Ney, 2008), and phrase-based translation (Andrés-Ferrer, 2010; Wuebker et al., 2012).

For example, in Expectation-Maximization (Dempster et al., 1977), the Expectation (E) step computes the posterior distribution over possible completions of the data, and the Maximization (M) step reestimates the model parameters as if that distribution had actually been observed. In most cases, the M step is identical to estimating the model from complete data, except that counts of observations from the E step are fractional. It is common to apply add-one smoothing to the M step, but we cannot apply KN smoothing.

Another example is instance weighting. If we assign a weight to each training instance to indicate how important it is (say, its relevance to a particular domain), and the counts are not integral, then we again cannot train the model using KN smoothing.

In this paper, we propose a generalization of KN smoothing (called *expected KN* smoothing) that operates on fractional counts, or, more precisely, on *distributions* over counts. We rederive all the steps of KN smoothing to operate on count distributions instead of integral counts. We demonstrate how to apply expected KN to two tasks where KN smoothing was not applicable before. One is language model domain adaptation, and the other is word alignment using the IBM models (Brown et al., 1993). In both tasks, expected KN smoothing improves performance significantly.

2 Smoothing on integral counts

Before presenting our method, we review KN smoothing on integer counts as applied to language models, although, as we will demonstrate in Section 7, KN smoothing is applicable to other tasks as well.

2.1 Maximum likelihood estimation

Let $\mathbf{u}w$ stand for an n-gram, where $\mathbf{u}$ stands for the $(n-1)$ context words and w, the predicted word. Let $c(\mathbf{u}w)$ be the number of occurrences of $\mathbf{u}w$. We use a bullet ($\bullet$) to indicate summation over words, that is, $c(\mathbf{u}\bullet) = \sum_w c(\mathbf{u}w)$. Under maximum-likelihood estimation (MLE), we max-

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 765–774,
Baltimore, Maryland, USA, June 23-25 2014.

imize

$$L = \sum_{\mathbf{u}w} c(\mathbf{u}w) \log p(w \mid \mathbf{u}),$$

obtaining the solution

$$p_{\text{mle}}(w \mid \mathbf{u}) = \frac{c(\mathbf{u}w)}{c(\mathbf{u}\bullet)}. \tag{1}$$

2.2 Absolute discounting

Absolute discounting (Ney et al., 1994) – on which KN smoothing is based – tries to generalize better to unseen data by subtracting a *discount* from each seen n-gram's count and distributing the subtracted discounts to unseen n-grams. For now, we assume that the discount is a constant D, so that the smoothed counts are

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

where $n_{1+}(\mathbf{u}\bullet) = |\{w \mid c(\mathbf{u}w) > 0\}|$ is the number of word types observed after context $\mathbf{u}$, and $q_{\mathbf{u}}(w)$ specifies how to distribute the subtracted discounts among unseen n-gram types. Maximizing the likelihood of the smoothed counts $\tilde{c}$, we get

$$p(w \mid \mathbf{u}) = \begin{cases} \dfrac{c(\mathbf{u}w) - D}{c(\mathbf{u}\bullet)} & \text{if } c(\mathbf{u}w) > 0 \\ \dfrac{n_{1+}(\mathbf{u}\bullet) D q_{\mathbf{u}}(w)}{c(\mathbf{u}\bullet)} & \text{otherwise.} \end{cases} \tag{2}$$

How to choose D and $q_{\mathbf{u}}(w)$ are described in the next two sections.

2.3 Estimating D by leaving-one-out

The discount D can be chosen by various means; in absolute discounting, it is chosen by the method of *leaving one out*. Given N training instances, we form the probability of each instance under the MLE using the other $(N-1)$ instances as training data; then we maximize the log-likelihood of all those instances. The probability of an n-gram token $\mathbf{u}w$ using the other tokens as training data is

$$p_{\text{loo}}(w \mid \mathbf{u}) = \begin{cases} \dfrac{c(\mathbf{u}w) - 1 - D}{c(\mathbf{u}\bullet) - 1} & c(\mathbf{u}w) > 1 \\ \dfrac{(n_{1+}(\mathbf{u}\bullet) - 1) D q_{\mathbf{u}}(w)}{c(\mathbf{u}\bullet) - 1} & c(\mathbf{u}w) = 1. \end{cases}$$

We want to find the D that maximizes the leaving-one-out log-likelihood

$$\begin{aligned} L_{\text{loo}} &= \sum_{\mathbf{u}w} c(\mathbf{u}w) \log p_{\text{loo}}(w \mid \mathbf{u}) \\ &= \sum_{\mathbf{u}w \mid c(\mathbf{u}w) > 1} c(\mathbf{u}w) \log \frac{c(\mathbf{u}w) - 1 - D}{c(\mathbf{u}\bullet) - 1} \\ &\quad + \sum_{\mathbf{u}w \mid c(\mathbf{u}w) = 1} \log \frac{(n_{1+}(\mathbf{u}\bullet) - 1) D q_{\mathbf{u}}(w)}{c(\mathbf{u}\bullet) - 1} \\ &= \sum_{r>1} r n_r \log(r - 1 - D) + n_1 \log D + C, \end{aligned} \tag{3}$$

where $n_r = |\{\mathbf{u}w \mid c(\mathbf{u}w) = r\}|$ is the number of n-gram types appearing r times, and C is a constant not depending on D. Setting the partial derivative with respect to D to zero, we have

$$\begin{aligned} \frac{\partial L_{\text{loo}}}{\partial D} &= -\sum_{r>1} \frac{r n_r}{r - 1 - D} + \frac{n_1}{D} \\ \frac{n_1}{D} &= \sum_{r>1} \frac{r n_r}{r - 1 - D} \geq \frac{2 n_2}{1 - D}. \end{aligned}$$

Solving for D, we have

$$D \leq \frac{n_1}{n_1 + 2 n_2}. \tag{4}$$

Theoretically, we can use iterative methods to optimize D. But in practice, setting D to this upper bound is effective and simple (Ney et al., 1994; Chen and Goodman, 1999).

2.4 Estimating the lower-order distribution

Finally, $q_{\mathbf{u}}(w)$ is defined to be proportional to an $(n-1)$-gram model $p'(w \mid \mathbf{u}')$, where $\mathbf{u}'$ is the $(n-2)$-gram suffix of $\mathbf{u}$. That is,

$$q_{\mathbf{u}}(w) = \gamma(\mathbf{u}) p'(w \mid \mathbf{u}'),$$

where $\gamma(\mathbf{u})$ is an auxiliary function chosen to make the distribution $p(w \mid \mathbf{u})$ in (2) sum to one.

Absolute discounting chooses $p'(w \mid \mathbf{u}')$ to be the maximum-likelihood unigram distribution; under KN smoothing (Kneser and Ney, 1995), it is chosen to make p in (2) satisfy the following constraint for all $(n-1)$-grams $\mathbf{u}'w$:

$$p_{\text{mle}}(\mathbf{u}'w) = \sum_{v} p(w \mid v\mathbf{u}') p_{\text{mle}}(v\mathbf{u}'). \tag{5}$$

Substituting in the definition of p_{mle} from (1) and p from (2) and canceling terms, we get

$$\begin{aligned} c(\mathbf{u}'w) &= \sum_{v \mid c(v\mathbf{u}'w) > 0} (c(v\mathbf{u}'w) - D) \\ &\quad + \sum_{v \mid c(v\mathbf{u}'w) = 0} n_{1+}(v\mathbf{u}'\bullet) D \gamma(v\mathbf{u}') p'(w \mid \mathbf{u}'). \end{aligned}$$

Solving for $p'(w \mid \mathbf{u}')$, we have

$$p'(w \mid \mathbf{u}') = \frac{\sum_{v|c(v\mathbf{u}'w)>0} 1}{\sum_{v|c(v\mathbf{u}'w)=0} n_{1+}(v\mathbf{u}'\bullet)\gamma(v\mathbf{u}')}.$$

Kneser and Ney assume the denominator is constant in w and renormalize to get an approximation

$$p'(w \mid \mathbf{u}') \approx \frac{n_{1+}(\bullet\mathbf{u}'w)}{n_{1+}(\bullet\mathbf{u}'\bullet)}, \tag{6}$$

where

$$n_{1+}(\bullet\mathbf{u}'w) = |\{v \mid c(v\mathbf{u}'w) > 0\}|$$
$$n_{1+}(\bullet\mathbf{u}'\bullet) = \sum_w n_{1+}(\bullet\mathbf{u}'w).$$

3 Count distributions

The computation of D and p' above made use of n_r and n_{r+}, which presupposes integer counts. But in many applications, the counts are not integral, but fractional. How do we apply KN smoothing in such cases? In this section, we introduce *count distributions* as a way of circumventing this problem.

3.1 Definition

In the E step of EM, we compute a probability distribution (according to the current model) over all possible completions of the observed data, and the expected counts of all types, which may be fractional. However, note that in each completion of the data, the counts are integral. Although it does not make sense to compute n_r or n_{r+} on fractional counts, it does make sense to compute them on possible completions.

In other situations where fractional counts arise, we can still think of the counts as expectations under some distribution over possible "realizations" of the data. For example, if we assign a weight between zero and one to every instance in a corpus, we can interpret each instance's weight as the probability of that instance occurring or not, yielding a distribution over possible subsets of the data.

Let X be a random variable ranging over possible realizations of the data, and let $c_X(\mathbf{u}w)$ be the count of $\mathbf{u}w$ in realization X. The expectation $E[c_X(\mathbf{u}w)]$ is the familiar fractional expected count of $\mathbf{u}w$, but we can also compute the probabilities $p(c_X(\mathbf{u}w) = r)$ for any r. From now on, for brevity, we drop the subscript X and understand $c(\mathbf{u}w)$ to be a random variable depending on X. The $n_r(\mathbf{u}\bullet)$ and $n_{r+}(\mathbf{u}\bullet)$ and related quantities also become random variables depending on X.

For example, suppose that our data consists of the following bigrams, with their weights:

(a)	fat cat	0.3
(b)	fat cat	0.8
(c)	big dog	0.9

We can interpret this as a distribution over eight subsets (not all distinct), with probabilities:

$\emptyset$	$0.7 \cdot 0.2 \cdot 0.1 = 0.014$
{a}	$0.3 \cdot 0.2 \cdot 0.1 = 0.006$
{b}	$0.7 \cdot 0.8 \cdot 0.1 = 0.056$
{a, b}	$0.3 \cdot 0.8 \cdot 0.1 = 0.024$
{c}	$0.7 \cdot 0.2 \cdot 0.9 = 0.126$
{a, c}	$0.3 \cdot 0.2 \cdot 0.9 = 0.054$
{b, c}	$0.7 \cdot 0.8 \cdot 0.9 = 0.504$
{a, b, c}	$0.3 \cdot 0.8 \cdot 0.9 = 0.216$

Then the count distributions and the $E[n_r]$ are:

	$r = 1$	$r = 2$	$r > 0$
$p(c(\text{fat cat}) = r)$	0.62	0.24	0.86
$p(c(\text{big dog}) = r)$	0.9	0	0.9
$E[n_r]$	1.52	0.24	

3.2 Efficient computation

How to compute these probabilities and expectations depends in general on the structure of the model. If we assume that all occurrences of $\mathbf{u}w$ are independent (although in fact they are not always), the computation is very easy. If there are k occurrences of $\mathbf{u}w$, each occurring with probability p_i, the count $c(\mathbf{u}w)$ is distributed according to the *Poisson-binomial distribution* (Hong, 2013). The expected count $E[c(\mathbf{u}w)]$ is just $\sum_i p_i$, and the distribution of $c(\mathbf{u}w)$ can be computed as follows:

$$p(c(\mathbf{u}w) = r) = s(k, r)$$

where $s(k, r)$ is defined by the recurrence

$$s(k, r) = \begin{cases} s(k-1, r)(1 - p_k) & \\ \quad + s(k-1, r-1)p_k & \text{if } 0 \le r \le k \\ 1 & \text{if } k = r = 0 \\ 0 & \text{otherwise.} \end{cases}$$

We can also compute

$$p(c(\mathbf{u}w) \ge r) = \max\left\{s(m, r), 1 - \sum_{r'<r} s(m, r')\right\},$$

the floor operation being needed to protect against rounding errors, and we can compute

$$E[n_r(\mathbf{u}\bullet)] = \sum_w p(c(\mathbf{u}w) = r)$$
$$E[n_{r+}(\mathbf{u}\bullet)] = \sum_w p(c(\mathbf{u}w) \ge r).$$

Since, as we shall see, we only need to compute these quantities up to a small value of r (2 or 4), this takes time linear in k.

4 Smoothing on count distributions

We are now ready to describe how to apply KN smoothing to count distributions. Below, we recapitulate the derivation of KN smoothing presented in Section 2, using the expected log-likelihood in place of the log-likelihood and applying KN smoothing to each possible realization of the data.

4.1 Maximum likelihood estimation

The MLE objective function is the expected log-likelihood,

$$\begin{aligned} E[L] &= E\left[\sum_{\mathbf{u}w} c(\mathbf{u}w) \log p(w \mid \mathbf{u})\right] \\ &= \sum_{\mathbf{u}w} E[c(\mathbf{u}w)] \log p(w \mid \mathbf{u}) \end{aligned}$$

whose maximum is

$$p_{\text{mle}}(w \mid \mathbf{u}) = \frac{E[c(\mathbf{u}w)]}{E[c(\mathbf{u}\bullet)]}. \tag{7}$$

4.2 Absolute discounting

If we apply absolute discounting to every realization of the data, the expected smoothed counts are

$$\begin{aligned} E[\tilde{c}(\mathbf{u}w)] &= \sum_{r>0} p(c(\mathbf{u}w) = r)(r - D) \\ &\quad + p(c(\mathbf{u}w) = 0) E[n_{1+}(\mathbf{u}\bullet)] D q_{\mathbf{u}}(w) \\ &= E[c(\mathbf{u}w)] - p(c(\mathbf{u}w) > 0) D \\ &\quad + p(c(\mathbf{u}w) = 0) E[n_{1+}(\mathbf{u}\bullet)] D q_{\mathbf{u}}(w) \end{aligned} \tag{8}$$

where, to be precise, the expectation $E[n_{1+}(\mathbf{u}\bullet)]$ should be conditioned on $c(\mathbf{u}w) = 0$; in practice, it seems safe to ignore this. The MLE is then

$$p(w \mid \mathbf{u}) = \frac{E[\tilde{c}(\mathbf{u}w)]}{E[\tilde{c}(\mathbf{u}\bullet)]}. \tag{9}$$

4.3 Estimating D by leaving-one-out

It would not be clear how to perform leaving-one-out estimation on fractional counts, but here we have a distribution over realizations of the data, each with integral counts, and we can perform leaving-one-out estimation on each of these. In other words, our goal is to find the D that maximizes the *expected* leaving-one-out log-likelihood, which is just the expected value of (3):

$$\begin{aligned} E[L_{\text{loo}}] &= E\Big[n_1 \log D + \sum_{r>1} r n_r \log(r - 1 - D) + C\Big] \\ &= E[n_1] \log D \\ &\quad + \sum_{r>1} r E[n_r] \log(r - 1 - D) + C, \end{aligned}$$

where C is a constant not depending on D. We have made the assumption that the n_r are independent.

By exactly the same reasoning as before, we obtain an upper bound for D:

$$D \leq \frac{E[n_1]}{E[n_1] + 2E[n_2]}. \tag{10}$$

In our example above, $D = \frac{1.52}{1.52+2\cdot 0.24} = 0.76$.

4.4 Estimating the lower-order distribution

We again require p' to satisfy the marginal constraint (5). Substituting in (7) and solving for p' as in Section 2.4, we obtain the solution

$$p'(w \mid \mathbf{u}') = \frac{E[n_{1+}(\bullet \mathbf{u}' w)]}{E[n_{1+}(\bullet \mathbf{u}' \bullet)]}. \tag{11}$$

For the example above, the estimates for the unigram model $p'(w)$ are

$$\begin{aligned} p'(\text{cat}) &= \tfrac{0.86}{0.86+0.9} \approx 0.489 \\ p'(\text{dog}) &= \tfrac{0.9}{0.86+0.9} \approx 0.511. \end{aligned}$$

4.5 Extensions

Chen and Goodman (1999) introduce three extensions to Kneser-Ney smoothing which are now standard. For our experiments, we used all three, for both integral counts and count distributions.

4.5.1 Interpolation

In interpolated KN smoothing, the subtracted discounts are redistributed not only among unseen events but also seen events. That is,

$$\tilde{c}(\mathbf{u}w) = \max\{0, c(\mathbf{u}w) - D\} + n_{1+}(\mathbf{u}\bullet) D p'(w \mid \mathbf{u}').$$

In this case, $\gamma(\mathbf{u})$ is always equal to one, so that $q_{\mathbf{u}}(w) = p'(w \mid \mathbf{u}')$. (Also note that (6) becomes an exact solution to the marginal constraint.) Theoretically, this requires us to derive a new estimate for D. However, as this is not trivial, nearly all implementations simply use the original estimate (4).

On count distributions, the smoothed counts become

$$\begin{aligned} E[\tilde{c}(\mathbf{u}w)] &= E[c(\mathbf{u}w)] - p(c(\mathbf{u}w) > 0) D \\ &\quad + E[n_{1+}(\mathbf{u}\bullet)] D p'(w \mid \mathbf{u}'). \end{aligned} \tag{12}$$

In our example, the smoothed counts are:

$\mathbf{u}w$	$E[\tilde{c}]$
fat cat	$1.1 - 0.86 \cdot 0.76 + 0.86 \cdot 0.76 \cdot 0.489 \approx 0.766$
fat dog	$0 \quad - 0 \cdot 0.76 \quad + 0.86 \cdot 0.76 \cdot 0.511 \approx 0.334$
big cat	$0 \quad - 0 \cdot 0.76 \quad + 0.9 \cdot 0.76 \cdot 0.489 \approx 0.334$
big dog	$0.9 - 0.9 \cdot 0.76 \quad + 0.9 \cdot 0.76 \cdot 0.511 \approx 0.566$

which give the smoothed probability estimates:

$$p(\text{cat} \mid \text{fat}) = \tfrac{0.766}{0.766+0.334} = 0.696$$
$$p(\text{dog} \mid \text{fat}) = \tfrac{0.334}{0.766+0.334} = 0.304$$
$$p(\text{dog} \mid \text{big}) = \tfrac{0.334}{0.334+0.556} = 0.371$$
$$p(\text{cat} \mid \text{big}) = \tfrac{0.556}{0.334+0.556} = 0.629.$$

4.5.2 Modified discounts

Modified KN smoothing uses a different discount D_r for each count $r < 3$, and a discount D_{3+} for counts $r \geq 3$. On count distributions, a similar argument to the above leads to the estimates:

$$\begin{aligned} D_1 &\leq 1 - 2Y\frac{E[n_2]}{E[n_1]} \\ D_2 &\leq 2 - 3Y\frac{E[n_3]}{E[n_2]} \\ D_{3+} &\approx 3 - 4Y\frac{E[n_4]}{E[n_3]} \\ Y &= \frac{E[n_1]}{E[n_1] + 2E[n_2]}. \end{aligned} \tag{13}$$

One side-effect of this change is that (6) is no longer the correct solution to the marginal constraint (Teh, 2006; Sundermeyer et al., 2011). Although this problem can be fixed, standard implementations simply use (6).

4.5.3 Recursive smoothing

In the original KN method, the lower-order model p' was estimated using (6); recursive KN smoothing applies KN smoothing to p'. To do this, we need to reconstruct counts whose MLE is (6). On integral counts, this is simple: we generate, for each n-gram type $v\mathbf{u}'w$, an $(n-1)$-gram token $\mathbf{u}'w$, for a total of $n_{1+}(\bullet\mathbf{u}'w)$ tokens. We then apply KN smoothing to these counts.

Analogously, on count distributions, for each n-gram type $v\mathbf{u}'w$, we generate an $(n-1)$-gram token $\mathbf{u}'w$ with probability $p(c(v\mathbf{u}'w) > 0)$. Since

$$E[c(\mathbf{u}'w)] = \sum_v p(c(v\mathbf{u}'w) > 0) = E[n_{1+}(\bullet\mathbf{u}'w)],$$

this has (11) as its MLE and therefore satisfies the marginal constraint. We then apply expected KN smoothing to these count distributions.

For the example above, the count distributions used for the unigram distribution would be:

	$r=0$	$r=1$
$p(c(\text{cat}) = r)$	0.14	0.86
$p(c(\text{dog}) = r)$	0.1	0.9

4.6 Summary

In summary, to perform expected KN smoothing (either the original version or Chen and Goodman's modified version), we perform the steps listed below:

	orig.	mod.
compute count distributions	§3.2	
estimate discount D	(10)	(13)
estimate lower-order model p'	(11)	§4.5.3
compute smoothed counts $\tilde{c}$	(8)	(12)
compute probabilities p	(9)	

The computational complexity of expected KN is almost identical to KN on integral counts. The main addition is computing and storing the count distributions. Using the dynamic program in Section 3.2, computing the distributions for each r is linear in the number of n-gram types, and we only need to compute the distributions up to $r = 2$ (or $r = 4$ for modified KN), and store them for $r = 0$ (or up to $r = 2$ for modified KN).

5 Related Work

Witten-Bell (WB) smoothing is somewhat easier than KN to adapt to fractional counts. The SRILM toolkit (Stolcke, 2002) implements a method which we call *fractional WB*:

$$p(w \mid \mathbf{u}) = \lambda(\mathbf{u})p_{\text{mle}}(w \mid \mathbf{u}) + (1 - \lambda(\mathbf{u}))p'(w \mid \mathbf{u}')$$
$$\lambda(\mathbf{u}) = \frac{E[c(\mathbf{u})]}{E[c(\mathbf{u})] + n_{1+}(\mathbf{u}\bullet)},$$

where $n_{1+}(\mathbf{u}\bullet)$ is the number of word types observed after context $\mathbf{u}$, computed by ignoring all weights. This method, although simple, inconsistently uses weights for counting tokens but not types. Moreover, as we will see below, it does not perform as well as expected KN.

The only previous adaptation of KN smoothing to fractional counts that we are aware of is that of Tam and Schultz (2008) and Bisani and Ney (2008), called *fractional KN*. This method subtracts D directly from the fractional counts, zeroing out counts that are smaller than D. The discount D must be set by minimizing an error metric on held-out data using a line search (Tam, p. c.) or Powell's method (Bisani and Ney, 2008), requiring repeated estimation and evaluation of the language model. By contrast, we choose D by leaving-one-out. Like KN on integral counts, our method has a closed-form approximation and requires neither held-out data nor trial and error.

6 Language model adaptation

N-gram language models are widely used in applications like machine translation and speech recognition to select fluent output sentences. Although they can easily be trained on large amounts of data, in order to perform well, they should be trained on data containing the right kind of language. For example, if we want to model spoken language, then we should train on spoken language data. If we train on newswire, then a spoken sentence might be regarded as ill-formed, because the distribution of sentences in these two domains are very different. In practice, we often have limited-size training data from a specific domain, and large amounts of data consisting of language from a variety of domains (we call this *general-domain* data). How can we utilize the large general-domain dataset to help us train a model on a specific domain?

Many methods (Lin et al., 1997; Gao et al., 2002; Klakow, 2000; Moore and Lewis, 2010; Axelrod et al., 2011) rank sentences in the general-domain data according to their similarity to the in-domain data and select only those with score higher than some threshold. Such methods are effective and widely used. However, sometimes it is hard to say whether a sentence is totally in-domain or out-of-domain; for example, quoted speech in a news report might be partly in-domain if the domain of interest is broadcast conversation. Here, we propose to assign each sentence a probability to indicate how likely it is to belong to the domain of interest, and train a language model using expected KN smoothing. We show that this approach yields models with much better perplexity than the original sentence-selection approach.

6.1 Method

One of the most widely used sentence-selection approaches is that of Moore and Lewis (2010). They first train two language models, p_{in} on a set of in-domain data, and p_{out} on a set of general-domain data. Then each sentence $\mathbf{w}$ is assigned a score

$$H(\mathbf{w}) = \frac{\log(p_{\text{in}}(\mathbf{w})) - \log(p_{\text{out}}(\mathbf{w}))}{|\mathbf{w}|}.$$

They set a threshold on the score to select a subset.

We adapt this approach as follows. After selection, for each sentence in the subset, we use a sigmoid function to map the scores into probabilities:

$$p(\mathbf{w} \text{ is in-domain}) = \frac{1}{1 + \exp(-H(\mathbf{w}))}.$$

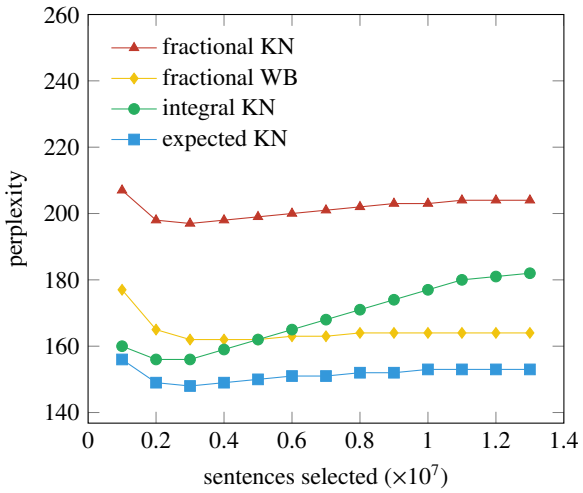

Figure 1: On the language model adaptation task, expected KN outperforms all other methods across all sizes of selected subsets. Integral KN is applied to unweighted instances, while fractional WB, fractional KN and expected KN are applied to weighted instances.

Then we use the weighted subset to train a language model with expected KN smoothing.

6.2 Experiments

Moore and Lewis (2010) test their method by partitioning the in-domain data into training data and test data, both of which are disjoint from the general-domain data. They use the in-domain training data to select a subset of the general-domain data, build a language model on the selected subset, and evaluate its perplexity on the in-domain test data. Here, we follow this experimental framework and compare Moore and Lewis's unweighted method to our weighted method.

For our experiments, we used all the English data allowed for the BOLT Phase 1 Chinese-English evaluation. We took 60k sentences (1.7M words) of web forum data as in-domain data, further subdividing it into 54k sentences (1.5M words) for training, 3k sentences (100k words) for testing, and 3k sentences (100k words) for future use. The remaining 12.7M sentences (268M words) we treated as general-domain data.

We trained trigram language models and compared expected KN smoothing against integral KN smoothing, fractional WB smoothing, and fractional KN smoothing, measuring perplexity across various subset sizes (Figure 1). For fractional KN, for each subset size, we optimized D to mini-

mize perplexity on the *test* set to give it the greatest possible advantage; nevertheless, it is clearly the worst performer. Expected KN consistently gives the best perplexity, and, at the optimal subset size, obtains better perplexity (148) than the other methods (156 for integral KN, 162 for fractional WB and 197 for fractional KN). Finally, we note that integral KN is very sensitive to the subset size, whereas expected KN and the other methods are more robust.

7 Word Alignment

In this section, we show how to apply expected KN to the IBM word alignment models (Brown et al., 1993). This illustrates both how to use expected KN inside EM and how to use it beyond language modeling. Of course, expected KN can be applied to other instances of EM besides word alignment.

7.1 Problem

Given a French sentence $\mathbf{f} = f_1 f_2 \cdots f_m$ and its English translation $\mathbf{e} = e_1 e_2 \cdots e_n$, an alignment $\mathbf{a}$ is a sequence $a_1, a_2, \ldots, a_m$, where a_i is the index of the English word which generates the French word f_i, or NULL. As is common, we assume that each French word can only be generated from one English word or from NULL (Brown et al., 1993; Och and Ney, 2003; Vogel et al., 1996).

The IBM models and related models define probability distributions $p(\mathbf{a}, \mathbf{f} \mid \mathbf{e}, \theta)$, which model how likely a French sentence $\mathbf{f}$ is to be generated from an English sentence $\mathbf{e}$ with word alignment $\mathbf{a}$. Different models parameterize this probability distribution in different ways. For example, Model 1 only models the lexical translation probabilities:

$$p(\mathbf{a}, \mathbf{f} \mid \mathbf{e}, \theta) \propto \prod_{j=1}^{m} p(f_j \mid e_{a_j}).$$

Models 2–5 and the HMM model introduce additional components to model word order and fertility. All, however, have the lexical translation model $p(f_j \mid e_i)$ in common. It also contains most of the model's parameters and is where overfitting occurs most. Thus, here we only apply KN smoothing to the lexical translation probabilities, leaving the other model components for future work.

7.2 Method

The $\mathbf{f}$ and $\mathbf{e}$ are observed, while $\mathbf{a}$ is a latent variable. Normally, in the E step, we collect expected counts $E[c(e, f)]$ for each e and f. Then, in the M step, we find the parameter values that maximize their likelihood. However, MLE is prone to overfitting, one symptom of which is the "garbage collection" phenomenon where a rare English word is wrongly aligned to many French words.

To reduce overfitting, we use expected KN smoothing during the M step. That is, during the E step, we calculate the distribution of $c(e, f)$ for each e and f, and during the M step, we train a language model on bigrams ef using expected KN smoothing (that is, with $\mathbf{u} = e$ and $w = f$). This gives a smoothed probability estimate for $p(f \mid e)$.

One question that arises is: what distribution to use as the lower-order distribution p'? Following common practice in language modeling, we use the unigram distribution $p(f)$ as the lower-order distribution. We could also use the uniform distribution over word types, or a distribution that assigns zero probability to all known word types. (The latter case is equivalent to a backoff language model, where, since all bigrams are known, the lower-order model is never used.) Below, we compare the performance of all three choices.

7.3 Alignment experiments

We modified GIZA++ (Och and Ney, 2003) to perform expected KN smoothing as described above. Smoothing is enabled or disabled with a command-line switch, making direct comparisons simple. Our implementation is publicly available as open-source software.[1]

We carried out experiments on two language pairs: Arabic to English and Czech to English. For Arabic-English, we used 5.4+4.3 million words of parallel text from the NIST 2009 constrained task,[2] and 346 word-aligned sentence pairs (LDC2006E86) for evaluation. For Czech-English, we used all 2.0+2.2 million words of training data from the WMT 2009 shared task, and 515 word-aligned sentence pairs (Bojar and Prokopová, 2006) for evaluation.

For all methods, we used five iterations of IBM Models 1, 2, and HMM, followed by three iterations of IBM Models 3 and 4. We applied expected KN smoothing to all iterations of all models. We aligned in both the foreign-to-English

[1] `https://github.com/hznlp/giza-kn`

[2] All data was used except for: United Nations proceedings (LDC2004E13), ISI Automatically Extracted Parallel Text (LDC2007E08), and Ummah newswire text (LDC2004T18).

Smoothing	p'	Alignment F1		B	
		Ara-Eng	Cze-Eng	Ara-Eng	Cze-Eng
none (baseline)	–	66.5	67.2	37.0	16.6
variational Bayes	uniform	65.7	65.5	36.5	16.6
fractional WB	unigram	60.1	63.7	–	–
	uniform	60.8	66.5	37.8	16.9
	zero	60.8	65.2	–	–
fractional KN	unigram	67.7	70.2	37.2	16.5
expected KN	unigram	**69.7**	**71.9**	**38.2**	**17.0**
	uniform	69.4	71.3	–	–
	zero	69.2	**71.9**	–	–

Table 1: Expected KN (interpolating with the unigram distribution) consistently outperforms all other methods. For variational Bayes, we followed Riley and Gildea (2012) in setting α to zero (so that the choice of p' is irrelevant). For fractional KN, we chose D to maximize F1 (see Figure 2).

and English-to-foreign directions and then used the *grow-diag-final* method to symmetrize them (Koehn et al., 2003), and evaluated the alignments using F-measure against gold word alignments.

As shown in Table 1, for KN smoothing, interpolation with the unigram distribution performs the best, while for WB smoothing, interestingly, interpolation with the uniform distribution performs the best. The difference can be explained by the way the two smoothing methods estimate p'. Consider again a training example with a word e that occurs nowhere else in the training data. In WB smoothing, $p'(f)$ is the empirical unigram distribution. If $\mathbf{f}$ contains a word that is much more frequent than the correct translation of e, then smoothing may actually encourage the model to wrongly align e with the frequent word. This is much less of a problem in KN smoothing, where p' is estimated from bigram types rather than bigram tokens.

We also compared with variational Bayes (Riley and Gildea, 2012) and fractional KN. Overall, expected KN performs the best. Variational Bayes is not consistent across different language pairs. While fractional KN does beat the baseline for both language pairs, the value of D, which we optimized D to maximize F1, is not consistent across language pairs: as shown in Figure 2, on Arabic-English, a smaller D is better, while for Czech-English, a larger D is better. By contrast, expected KN uses a closed-form expression for D that outperforms the best performance of fractional KN.

Table 2 shows that, if we apply expected KN smoothing to only selected stages of training, adding smoothing always brings an improvement,

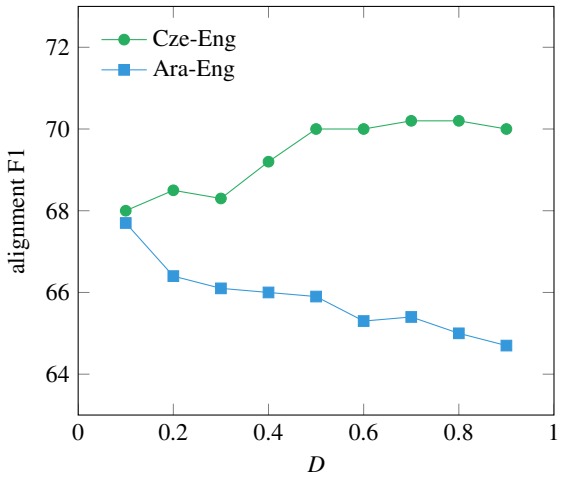

Figure 2: Alignment F1 vs. D of fractional KN smoothing for word alignment.

Smoothed models					Alignment F1	
1	2	H	3	4	Ara-Eng	Cze-Eng
○	○	○	○	○	66.5	67.2
●	○	○	○	○	67.3	67.9
○	●	○	○	○	68.0	68.7
○	○	●	○	○	68.6	70.0
○	○	○	●	○	66.9	68.4
○	○	○	○	●	67.0	68.6
●	●	●	●	●	**69.7**	**71.9**

Table 2: Smoothing more stages of training makes alignment accuracy go up. For each row, we smoothed all iterations of the models indicated. Key: H = HMM model; ● = smoothing enabled; ○ = smoothing disabled.

with the best setting being to smooth all stages. This shows that expected KN smoothing is consistently effective. It is also interesting to note that smoothing is less helpful for the fertility-based Models 3 and 4. Whether this is because modeling fertility makes them less susceptible to "garbage collection," or the way they approximate the E step makes them less amenable to smoothing, or another reason, would require further investigation.

7.4 Translation experiments

Finally, we ran MT experiments to see whether the improved alignments also lead to improved translations. We used the same training data as before. For the Arabic-English tasks, we used the NIST 2008 test set as development data and the NIST 2009 test set as test data; for the Czech-English tasks, we used the WMT 2008 test set as development data and the WMT 2009 test set as test data.

We used the Moses toolkit (Koehn et al., 2007) to build MT systems using various alignments (for expected KN, we used the one interpolated with the unigram distribution, and for fractional WB, we used the one interpolated with the uniform distribution). We used a trigram language model trained on Gigaword (AFP, AP Worldstream, CNA, and Xinhua portions), and minimum error-rate training (Och, 2003) to tune the feature weights.

Table 1 shows that, although the relationship between alignment F1 and B is not very consistent, expected KN smoothing achieves the best B among all these methods and is significantly better than the baseline ($p < 0.01$).

8 Conclusion

For a long time, and as noted by many authors, the usage of KN smoothing has been limited by its restriction to integer counts. In this paper, we addressed this issue by treating fractional counts as distributions over integer counts and generalizing KN smoothing to operate on these distributions. This generalization makes KN smoothing, widely considered to be the best-performing smoothing method, applicable to many new areas. We have demonstrated the effectiveness of our method in two such areas and showed significant improvements in both.

Acknowledgements

We thank Qing Dou, Ashish Vaswani, Wilson Yik-Cheung Tam, and the anonymous reviewers for their input to this work. This research was supported in part by DOI IBC grant D12AP00225.

References

Jesús Andrés-Ferrer. 2010. *Statistical approaches for natural language modelling and monotone statistical machine translation*. Ph.D. thesis, Universidad Politécnica de Valencia.

Amittai Axelrod, Xiaodong He, and Jianfeng Gao. 2011. Domain adaptation via pseudo in-domain data selection. In *Proc. EMNLP*, pages 355–362.

Maximilian Bisani and Hermann Ney. 2008. Joint-sequence models for grapheme-to-phoneme conversion. *Speech Communication*, 50(5):434–451.

Ondřej Bojar and Magdalena Prokopová. 2006. Czech-English word alignment. In *Proc. LREC*, pages 1236–1239.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19:263–311.

Stanley F. Chen and Joshua Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech and Language*, 13:359–394.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1–38.

Jianfeng Gao, Joshua Goodman, Mingjing Li, and Kai-Fu Lee. 2002. Toward a unified approach to statistical language modeling for Chinese. *ACM Transactions on Asian Language Information*, 1:3–33.

Joshua T. Goodman. 2001. A bit of progress in language modeling: Extended version. Technical Report MSR-TR-2001-72, Microsoft Research.

Joshua Goodman. 2004. Exponential priors for maximum entropy models. In *Proc. HLT-NAACL*, pages 305–312.

Yili Hong. 2013. On computing the distribution function for the Poisson binomial distribution. *Computational Statistics and Data Analysis*, 59:41–51.

Dietrich Klakow. 2000. Selecting articles from the language model training corpus. In *Proc. ICASSP*, pages 1695–1698.

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

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proc. HLT-NAACL*, pages 127–133.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. ACL, Companion Volume*, pages 177–180.

Sung-Chien Lin, Chi-Lung Tsai, Lee-Feng Chien, Keh-Jiann Chen, and Lin-Shan Lee. 1997. Chinese language model adaptation based on document classification and multiple domain-specific language models. In *Proc. Eurospeech*, pages 1463–1466.

Robert Moore and William Lewis. 2010. Intelligent selection of language model training data. In *Proc. ACL*, pages 220–224.

Hermann Ney, Ute Essen, and Reinhard Kneser. 1994. On structuring probabilistic dependencies in stochastic language modelling. *Computer Speech and Language*, 8:1–38, 1.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. ACL*, pages 160–167.

Darcey Riley and Daniel Gildea. 2012. Improving the IBM alignment models using variational Bayes. In *Proc. ACL (Volume 2: Short Papers)*, pages 306–310.

Andreas Stolcke. 2002. SRILM – an extensible language modeling toolkit. In *Proc. International Conference on Spoken Language Processing*, volume 2, pages 901–904.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2011. On the estimation of discount parameters for language model smoothing. In *Proc. Interspeech*, pages 1433–1436.

Yik-Cheung Tam and Tanja Schultz. 2008. Correlated bigram LSA for unsupervised language model adaptation. In *Proc. NIPS*, pages 1633–1640.

Yee Whye Teh. 2006. A hierarchical Bayesian language model based on Pitman-Yor processes. In *Proc. COLING-ACL*, pages 985–992.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proc. COLING*, pages 836–841.

Joern Wuebker, Mei-Yuh Hwang, and Chris Quirk. 2012. Leave-one-out phrase model training for large-scale deployment. In *Proc. WMT*, pages 460–467.