The Log-Log Term Frequency Distribution

Jason D. M. Rennie
jrennie@gmail.com

July 14, 2005

Abstract

Though commonly used, the unigram is widely known as being a poor model of term frequency; it assumes that term occurrences are independent, whereas many words, especially topic-oriented ones, tend to occur in bursts. Herein, we propose a model of term frequency that treats words independently, but allows for much higher variance in frequency values than does the unigram. Although it has valuable properties, and may be useful as a teaching tool, we are not able to find any applications that make a compelling case for its use.

1 The Unigram Model

The unigram is a simple, commonly used, model of text. It assumes that each word occurrence is independent of all other word occurrences. There is one parameter per word, θ_i, $\sum_i \theta_i = 1$, which corresponds to that word's rate of occurrence. For a document of length l, the chance that a unigram with parameters $\{\theta_i\}$ yields term frequencies $\{x_i\}$ is

$$P_{\text{uni}}\left(\{x_i\} \middle| \sum_i x_i = l; \{\theta_i\}\right) = \frac{l!}{\prod_i x_i!} \prod_i \theta_i^{x_i}. \tag{1}$$

The normalization constant, $\frac{l!}{\prod_i x_i!}$ corresponds to the number of ways there are to arrange the words so as to achieve the given frequency values.

An issue with the unigram is that it assigns low weight to the possibility that a somewhat rare word may occur many times. It generally does a good job of modeling common, "English" words such as "the," "a" and "is." But, it tends to greatly underweight large frequency values for topic-oriented words. For applications like classification, this leads to poor class-conditional estimates and thus poor classification accuracy.

1.1 Binomial

We have mentioned that the unigram poorly models topic-oriented words. But, the unigram allows for dependence between words. This dependence is mild, especially for the case that is usual with English text—a large vocabulary, and word rates almost strictly below 10%. Hence, for text, an excellent approximation to the unigram is a product of binomial distributions, one binomial for each word. The product of binomials uses the same parameters as the unigram (and they have the same interpretation). For a document of length l, the chance that a product of binomials with parameters $\{\theta_i\}$ yields term frequencies $\{x_i\}$ is

$$L_{\text{bin}}(\{x_i\}|l) = \prod_i \frac{l!}{x_i!(l-x_i)!}\theta_i^{x_i}(1-\theta_i)^{l-x_i} \quad \text{for } x_i \in \{0,1,\ldots,l\}. \tag{2}$$

Figure 1 shows the combined histogram plot of frequency values of all words over all documents, in addition to the maximum likelihood fit of a binomial model of the data. In all cases, the binomial model underestimates the chance of events somewhat far from the mean. It is too "light-tailed." These graphs are somewhat deceiving, since part of the heavy-tailed-ness is due to document length variation. However, it appears that we may find benefit in a model that assigns probability mass further from the mean or mode—a so-called "heavy-tailed" distribution. We introduce such a model in the next section.

2 The Log-Log Model

Ignoring the normalization constant (which does not strongly affect the shape of the distribution), the Unigram/Binomial has an exponential dependence between frequency and probability density—probability falls off quickly as we move away from the mode. Here, we introduce a higher-variance model that assigns greater probability away from the mode. Specifically, we consider a distribution with a polynomial relationship between frequency values and probability mass. In its simplest form, the distribution we consider is

$$P(x) \propto (x+b)^a, \tag{3}$$

where x is the frequency value, and b and a are parameters of the distribution. We call b the "bias" and a the "axponent." This makes sense as a distribution if $b > 0$ (since $x \in \{0,1,2,\ldots\}$). To begin, we allow the distribution to assign mass to all values $x \geq 0$. Hence, we must have $a < -1$ in order that P has a finite normalizer. Later we discuss conditioning the distribution on document (as is done with the Unigram/Binomial) so that the normalization constant is finite for any value of a. We call this the "log-log" distribution, because, unlike the Unigram, which shows a linear relation between frequency and log-probability, our new distribution shows a logarithmic relation between frequency and log-probability.

We introduce a full form of the log-log model, then proceed to provide graphs displaying the fit of the log-log model to empirical data. To begin, we use one axponent parameter per word and a single bias parameter as a global scaling factor. Later, we discuss variations on the model. Given axponent parameters

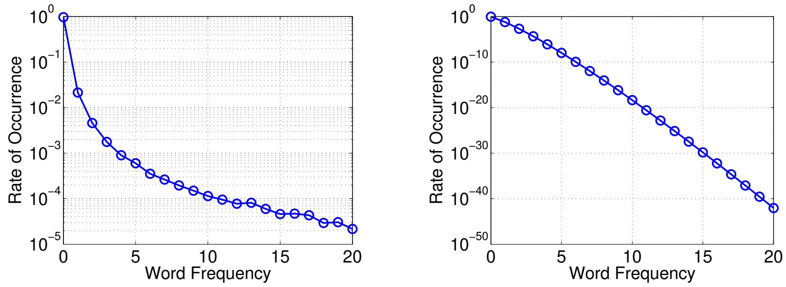

Figure 1: Shown are plots of (left) the empirical frequency distribution for all words, and (right) the corresponding maximum likelihood binomial model. The model highly underestimates large frequency values.

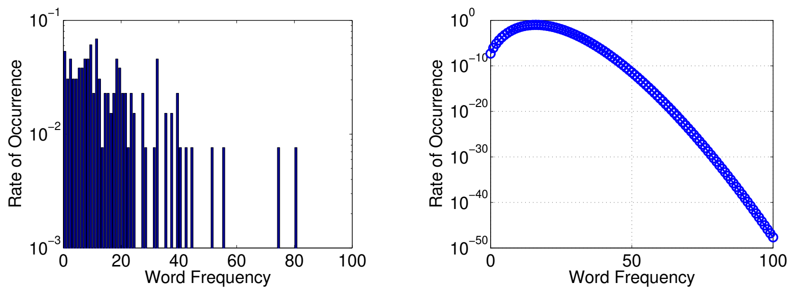

Figure 2: Shown are plots of (left) the empirical frequency distribution for the word "the," and (right) the corresponding maximum likelihood binomial model. As is to be expected of an exponential model, the model is too peaked, underestimating frequency rates not extremely close to the mean (which is between 16 and 17 occurrences for an average length document).

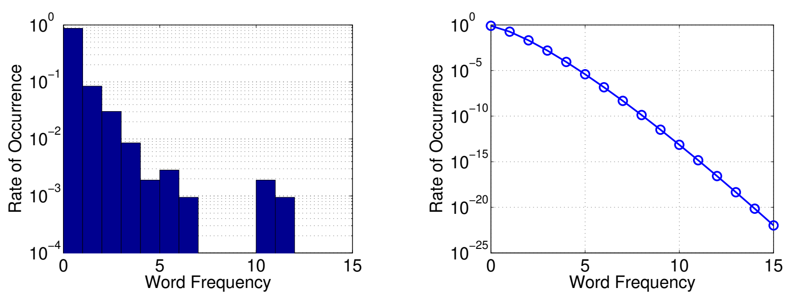

Figure 3: Shown are plots of (left) the empirical frequency distribution for eight words that occur exactly 30 times in the data, and (right) the corresponding maximum likelihood binomial model. The model greatly underestimates the chance of observing high frequency values.

$\{a_i\}$, and bias parameter b, the log-log model assigns the following probability to the set of word frequencies $\{x_{ij}\}$,

$$P_{\log}(\{x_{ij}\}) = \prod_{i,j} \frac{(x_{ij}+b)^{a_i}}{\sum_{x=0}^{\infty}(x+b)^{a_i}}. \tag{4}$$

The normalization constant, $Z(a,b) = \sum_{x=0}^{\infty}(x+b)^a$, is not generally feasible to compute exactly. Note that $Z(-a,1) = \zeta(a)$ is known as the Riemann Zeta Function. We can compute an excellent approximation to Z using partial sums. Let $S_n = \sum_{x=0}^{n}(x+b)^a$. We choose some large value, v, and construct a polynomial k^{th} degree fit of $f(x) = S_{1/x}$ for $x \in \{\frac{1}{v}, \frac{1}{v-1}, \ldots, \frac{1}{v-k}$, and use the value of the fitted function at $x = 0$ as our approximation for Z. P may not be a distribution if $a \geq -1$ or $b \leq 0$. Additionally, the expectation of P is not finite if $a \geq -2$.

Next, we consider whether this log-log model is an improvement over the Unigram. First, we look at overall fit to three different data sets. Then, we look at what sorts of words each model fits best.

2.1 Data

Here we describe the data sets that we will use for evaluating our model(s).

The Restaurant data is a collection of 132 threads from a local-area restaurant discussion bulletin board containing 615 messages. We pre-process the Restaurant data in preparation for the task of named-entity extraction. Thus, many non-words (such as punctuation) are tokenized.

The WebKB data is a popular text classification data set. It is a collection of web pages from computer science departments. We use the recommended pre-processing, which removes headers and HTML, treats digits specially, and does not use a "stop list." We also bypass documents empty documents and documents with only a single word. We avoid the "other" category, which is a truly miscellaneous collection of pages[1]; we use the other universities for training since the four universities collection is extremely class-skewed and relatively small once the "other" category is removed.

The 20 Newsgroups data is also a popular text classification data set. It is a collection of newsgroup postings from 20 varied newsgroups. We use a collection sorted by date so that our training and test sets are separated in time. This collection also has duplicate postings removed. For processing, we remove headers, bypass empty posts and those with uuencoded blocks, and do not use a "stop list."

2.2 Overall Fit

Here we look at how the Log-Log model fits actual term frequency data and how it compares to the Unigram and Binomial. Table 1 gives information on

[1]When we tried including the "other" category, we found that the Log-Log model fit extremely well (compared to the Unigram). We found that this was due to the odd mixture of pages in the "other" category and not so much because it was modeling something valuable

	Restaurant	WebKB	20 News
Log-Log	2.8976	**3.6154**	3.7822
Unigram	**2.7340**	3.8283	**3.7348**
Binomial	2.7483	3.8516	3.7567

Table 1: Shown are average, per-word encoding length (in bits) of three data sets using three different term frequency models. The log-log model gives the smallest encoding (best fit) for both WebKB and 20 Newsgroups. The Unigram gives the smallest encoding for the Restaurant data. The Binomial consistently requires a slightly larger encoding that the Unigram.

fit. We trained each model on term frequency data for three different data sets, Restaurant, WebKB and 20 Newsgroups. For WebKB and 20 Newsgroups, the documents are organized into different classes[2]. To fit these data, we train the model once for each class of documents, calculate a total encoding length (or negative log-likelihood) for each class, sum the class-lengths, then divide by the total number of word occurrences in the data to achieve an average per-word encoding length.

The Log-Log model provides the better fit (smallest encoding length) for WebKB. Compared to Unigram, the Log-Log model provides the better fit for 5 of the 6 WebKB classes ("department" is the lone exception). The Unigram provies the better fit for both 20 Newsgroups and the Restaurant data; the Unigram provides the better fit for 14 of the 20 categories (5 newsgroups are better fit by Log-Log, the models fit talk.politics.mideast equally well). The fact that Unigram provides the best fit on the Restaurant data may be somewhat affected by the fact that our pre-processing includes punctuation as tokens.

Considering that our Log-Log model is so different from the Unigram, is document-length unaware, and has only one additional parameter, the Log-Log model seems to provide a reasonable overall fit. However, it is not compelling as a term frequency model. In the next section, we consider how well the model fits the frequency distributions of individual words.

2.3 Per-Word Fit

Here we present lists of words/tokens that had the largest difference in model fit between the Binomial[3] and Log-Log models. Whereas in the last section, we fit one set of parameters for each class (for WebKB and 20 News), here we fit one set of parameters for the entire data set. Thus, the model has no information about class labels. We present valuses which are difference in encoding length between the two models. We also calculate for WebKB and 20 News, for each word, the

[2]We train one set of parameters per class to avoid any advantage the Log-Log model might gain from mixing data from different topics.

[3]We use the Binomial in place of the Unigram because we cannot easily look at the fit for a specific word in the Unigram model.

Token	Difference
.	341.5
the	278.9
and	237.6
a	237.0
,	221.0
i	204.7
of	191.4
to	170.3
in	140.4
but	137.1

Token	Difference
sichuan	135.3
fish	85.1
speed	65.9
buffet	60.2
...	56.8
lobster	49.8
tacos	46.2
sour	42.7
greek	41.4
sauce	40.8

Table 2: Shown are (left) the 10 tokens with the best fit to the Binomial model, and (right) the 10 tokens with the best fit to the Log-Log model. Differences are in terms of bits of total encoding length on the Restaurant data.

mutual information (MI) between the label and the word[4]. We observe that the Binomial tends to provide better fit for uninformative or "English" words, such as "the," "a" and "and." We find that the Log-Log model provides the best fit for topic-oriented words. Also, many words with high mutual information with the class label are fit well by the Log-Log model. These observations suggest that this sort of difference in model fit may be a good way to find words that are good for classification (i.e. feature selection).

Table 2 gives the tokens in the Restaurant data with the largest differences in model fit. The tokens with (comparatively) the best Binomial fit are punctuation and "stop list" words (words that provide structure for the English language, but relate little to content). The tokens with the best Log-Log fit include words that appear in restaurant names ("sichuan," "fish," "speed," "buffet") and words that refer to a type of food or describe food ("lobster," "tacos," "sour," "greek," "sauce"). These are all topic-oriented words that are highly informative about the topic of a discussion. The lone exception, the elipsis, is a stylistic convention that is used by a subset of authors on the board. It would be a valuable marker for attempting to identify authorship.

Table 3 gives the words in the WebKB data with the largest differences in model fit. Again we see "English" words providing the best fit for the Unigram model. There appears to be some correlation with the classification label, but it is unlikely that any of it would generalize to unseen data. In the top 10 best fit words for the Log-Log model, we see three words/tokens with relatively large MI values. "Time" is not the word time, but rather any sequence of characters that specify a time of day (e.g. 5:29, or 12:00). As one might expect, these are extremely common on course pages (a time is found on 58% of course pages; the next highest rate is 18% for department pages). The word "my" is very common on student, staff and faculty pages, and less common on other pages.

[4]When calculating mutual information, we only use word presence/absence information and ignore frequency information.

Token	Diff	MI
of	4730.0	0.0180
and	4085.0	0.0254
for	3155.8	0.0157
in	3154.9	0.0062
to	2503.9	0.0119
on	2334.9	0.0060
with	1290.6	0.0114
an	1230.9	0.0125
is	1228.0	0.0204
from	1153.9	0.0061

Token	Diff	MI
OneDigit	7938.8	0.0703
TwoDigit	4228.1	0.0298
Time	3729.3	0.1325
nbsp	3135.7	0.0003
my	2851.8	0.2214
Digits	2764.1	0.0238
homework	2479.7	0.1511
parallel	2167.7	0.0235
eecs	2091.1	0.0068
postscript	2053.1	0.0078

Table 3: Shown are (left) the 10 tokens with the best fit to the Binomial model, and (right) the 10 tokens with the best fit to the Log-Log model. Differences are in terms of bits of total encoding length for the WebKB data. Each table also include a column giving mutual information (MI) with the classification label.

The word "homework" is almost exclusively found on course pages. Digits are commonly used to refer to courses, and so are more common on course pages than other pages. Again, we see that words that are better fit by the Log-Log distribution are related to topics in the data.

Table 4 gives the words in the 20 Newsgroups data with the largest differences in model fit. Again we see "English" words providing the best fit for the Unigram model. Although the mutual information values are not trivial, these words all appear regularly across all classes and would be of minimal use for discrimination purposes. In contrast, the words that best fit the Log-Log modeltend to be highly indicative of the topic. The word "he" is rare in technical newsgroups (such as comp.os.ms-windows.misc and sci.electronics), but very common in religious (where it is used to refer to God), political and sports newsgroups (where it refers to politicians and ballplayers). The word "god" is very common in religion-related newsgroups (such as alt.atheism and soc.religion.christian), and rare in other newsgroups; "file" is common in computer-related newsgroups; "scsi" refers to a type of hard drive (and the underlying protocol) and is common in the hardware and for sale newsgroups; "we" is found in most newsgroups, but is especially common in the religion-related newsgroups. Again, we see that words that are better fit by the Log-Log distribution are related to topics in the data.

Although the Log-Log model does not clearly provide a better overall fit to textual data, it does provide a better fit for informative, or topic-centric words. By comparing to Binomial fit, we can identify words that are associated with topics that are discussed in the text. Thus, we might be able to use the Log-Log model as part of a feature selection algorithm. Or, we might be able to use it for unsupervised, or semi-supervised discrimination methods as it appears to be able to identify informative words without class label information.

Token	Diff	MI
to	16907.6	0.0190
in	15773.8	0.0258
the	14948.6	0.0222
and	14568.6	0.0143
of	11890.8	0.0394
for	9333.3	0.0048
is	9184.9	0.0255
it	7899.1	0.0240
this	7616.9	0.0218
that	7563.9	0.0549

Token	Diff	MI
he	6768.1	0.0772
db	5421.0	0.0019
god	4999.5	0.1099
file	4367.5	0.0374
scsi	4106.9	0.0206
we	3216.8	0.0609
key	3204.5	0.0617
space	3199.9	0.0553
drive	3161.3	0.0352
windows	3086.9	0.0958

Table 4: Shown are (left) the 10 tokens with the best fit to the Binomial model, and (right) the 10 tokens with the best fit to the Log-Log model. Differences are in terms of bits of total encoding length for the 20 Newsgroups data. Each table also include a column giving mutual information (MI) with the classification label.

3 Variations on the Log-Log Model

Recall that we introduced the Log-Log model so that the probability mass associated with a certain term frequency value is proportional to a polynomial function of the term frequency,

$$P(x) \propto (x+b)^a. \tag{5}$$

We chose to make a (what we call the "axponent") the per-word parameter and to use a single b (the "bias") to scale the distribution. In this section, we discuss three extensions. The first, an almost trivial extension, is to make the model length-conditional and only normalize over frequency values less than or equal to the document length. For the second extension, we make the model truly document-length aware and consider the problem of modeling frequency rates rather than frequency counts. Finally, we discuss the alternate parameterization where the bias is the per-word parameter and the axponent is the single scaling parameter.

3.1 The Length-Conditional Normalizer

An obvious improvement on the Log-Log model as we have discussed it so far is simply to limit the normalization to the length of the document. This eliminates the possibility that the distribution is improper if $a \leq 1$, but makes the normalization constant conditional on the document. Let $l_j = \sum_i x_{ij}$ be the length of document j. Then, recalling Equation 4, the new normalization constant is $Z(a, b, l) = \sum_{x=0}^{l} (x+b)^a$ and the new distribution is $P(\{x_{ij}\}) = \prod_{i,j} (x_{ij}+b)^{a_i} / Z(a_i, b, l_j)$. We find that this trick improves model fit/decreases encoding length somewhat, but it also makes the gradient/objective code more computationally intensive.

3.2 The Log-Log Frequency Rate Model

Whereas normalizing only over feasible frequency values saves some probability mass, it does little to make the model better account for documents of different length. Even with a length-conditional normalizer, the model still assigns very similar probability masses to small frequency values whether the document is 100 words long or 1000. Here, we consider a different perspective, that of modeling frequency rates rather than frequency counts. Now, the x in our model is a rate value $\in [0,1]$ instead of a count $\in \{0,1,2,\dots,l\}$ (where l is the length of the document). But, although rate values are not integer, they are discrete; possible values are $\{0, 1/l, 2/l, \dots, 1\}$. We could follow our earlier development, simply using $P(x/l) \propto (x/l+b)^a$ as our unnormalized distribution. However, by instead using area under the curve to assign probability mass, we can make our normalization constant very easy to manage. Let x be a term frequency value; let l be the length of the document; let a and b be parameters of the distribution. Then, we assign probability mass to x proportional to the integral from x/l to $(x+1)/$ of the function $f(r) = (r+b)^a$. Our normalization constant is simply the integral from 0 to $(l+1)/l$. Using the same notation and parameterization as before, our Log-Log rate model is

$$P\left(\frac{x_{ij}}{l_j}\right) = \frac{1}{Z(a_i, b, l_j)} \int_{\frac{x_{ij}}{l_j}}^{\frac{x_{ij}+1}{l_j}} (r+b)^{a_i}\, dr, \tag{6}$$

where $Z(a_i, b, l_j) = \int_0^{(l_j+1)/l_j} (r+b)^{a_i}\, dr$. While retaining the desired "heavy-tail," this model also scales nicely with length—the part of the curve used to determine probability mass depends on the rate, not the raw frequency value. However, in limited experimentation, our Log-Log rate model does not seem to produce substantially better fit than our basic Log-Log model.

3.3 An Alternate Parameterization

Finally, we discuss an alternate parameterization for the Log-Log model. We have so far assumed the use of one axponent per-word and a single bias parameter which acts as a scaling factor. Here, we reverse the roles, using one bias parameter per-word and a single axponent parameter. Using the set-up from Equation 4, our per-word bias model is

$$P(x) = \frac{(x+b_i)^a}{\sum_{x=0}^{\infty}(x+b_i)^a}. \tag{7}$$

We note that this change of parameters is also easily achieved for the rate model. A disadvantage of this parameterization is that it is highly non-convex—each bias parameter essentially adds an additional non-convexity. Whereas the original parameterization can be solved quickly via approximate second-order methods (such as pre-conditioned Conjugate Gradients), such methods do not seem to provide any advantage over first-order methods (such as plain Conjugate

Gradients). However, the bias-per-word parameterization yields somewhat improved fit in a test on the Restaurant data (though still behind the Unigram). We were not able to run tests on the other data sets—optimization is very slow.

4 On Model Fit and Classification Accuracy

We have established that the Log-Log model may be useful for identifying informative, or topic-related terms. In this section, we explore whether this improved fit on informative words translates into an improved ability to perform discrimination tasks.

4.1 Classification

We conducted classification experiments on the WebKB data, training one set of parameters for each class, then for test data, assigning the label of the corresponding model that provides the best fit. What we find is that the Log-Log model performs extremely poorly. For each model, we smooth by assuming that each model contains an additional single document with each word occurring once. This is the standard smoothing technique for the Unigram and seems to be a reasonable choice for the Log-Log model. The Unigram is effective, misclassifying only 22.1% of the test documents when we use the entire vocabulary. Performance is somewhat worse, 24.5% error, when we select 1000 features with the highest mutual information on the training set. We also tested feature selection using the difference in Binomial encoding length and Log-Log encoding length (as suggested in an earlier section). We find that this method of selecting features is not as effective as mutual information, achieving 27.3% error when we select the 1000 features with the largest difference. However, unlike mutual information, this methods uses no label information. For comparison, selected the bottom 1000 features according to this differences and found 39.2% error—the best-fit Log-Log words are better for classification than the best-fit Binomial words. Note that always selecting the most frequent class in the test set ("student") achieves 48.9% error. When we try to use the Log-Log model for classification, we find that performance is exceedingly poor, as it nearly always selects the "department" class. We note that the bias parameter, b, selected during optimization for the "department" model was the largest of all the class models; we posit that classification may be highly sensitive to this bias value. However, when we fix $b = 1$ for all class models, we again find that performance is dismal—this time the Log-Log classifier always selects the "student" model, achieving 48.9% error. We note that the "department" model selected the lowest minimum axponent value; we posit that this may be the reason that it always selects the "department" class.

Though maximum-likelihood training of parameters for the Log-Log model does not achieve effective classification, we find good performance by training the parameters of the model discriminatively. The discriminative version of the Unigram model is a linear classifier where the features are the word counts.

Regularization Parameter	Error	
	Unigram	Log-Log
10^4	20.0%	19.9%
10^3	16.1%	13.9%
10^2	17.0%	**13.4%**
10^1	18.7%	15.4%
10^0	19.1%	16.5%

Table 5: Shown are classification errors on the WebKB data set using a linear, discriminative classifier. The left column gives the value of the regularization parameter used for the classifier. The middle column gives error when frequency values are used as features. The right column gives error when frequency values are first transformed via $f(x) = \log(x+1)$.

Regularization Parameter	Error	
	Unigram	Log-Log
10^4	34.1%	33.4%
10^3	28.1%	28.3%
10^2	27.4%	**25.7%**
10^1	28.6%	26.1%
10^0	29.3%	27.4%

Table 6: Shown are classification errors on the 20 Newsgroups data set using a linear, discriminative classifier. The left column gives the value of the regularization parameter used for the classifier. The middle column gives error when frequency values are used as features. The right column gives error when frequency values are first transformed via $f(x) = \log(x+1)$.

Maximum-likelihood training of the Unigram model selects weights equal to the logarithm of the empirical frequency rate. However, these are not generally the best settings for classification. Similarly for the Log-Log model, those parameters trained via maximum-likelihood (or to maximize fit of the data), are not necessarily good for classification. The discriminative version of the Log-Log model scores documents according to $a_i \log(x_{ij} + b)$, summed over words in the vocabulary. For simplicity, we set $b = 1$ and have the classifier learn the $\{a_i\}$. This is equivalent to simply transforming the term frequency values via $f(x) = \log(x+1)$ and learning linear weights.

We conducted similar experiments on the 20 Newsgroups data set, comparing discriminative version of the Unigram and Log-Log models. Again, we used a simple translation of the Log-Log model—training linear weights on data that had been transformed via $f(x) = \log(x+1)$. As with WebKB, we found better results using the transformed data. Table 6 gives the results. Only for one regularization parameter did the Unigram-based model outperform. And, the Log-Log-based model achived by far the lowest error.

When we trained the Unigram and Log-Log models to maximize the like-

	F1 breakeven
Baseline	55.04%
Log-Log	56.27%
IDF*Log-Log	**58.09**%

Table 7: Shown are F1-breakeven values for three different sets of features. "Baseline" includes only traditional NEE features. "Log-Log" adds the Log-Log score. "IDF*Log-Log" adds a feature which is the product of the IDF score and the Log-Log score. Larger values are better.

lihood (or fit) of the data, we found that the Unigram achieved reasonable classification performance, but the Log-Log model performed miserably. However, when we used a discriminative objective that minimized a bound on the classification error, we found that both models achieved good rates of error, and the Log-Log model outperformed the Unigram, with lower error for all values of the regularization parameter.

4.2 Named Entity Detection

We also consider the use of the Log-Log model in a task of named entity detection. A first step in extracting information from a text collection is the identification of named entities. For the restaurant discussion board data we have collected, it is valuable to be able to identify restaurant names in order to extract further information about restaurants (such as reviews, dishes, the name of the chef, location, etc.). As our Log-Log model provides good fit to topic-oriented terms, and restaurants are the main item of discussion in this collection, we posit that fit of the Log-Log model (in comparison to the Ungram) may be a useful feature in identifying restaurant names.

We use the same setup as in [1], using the difference between Log-Log model fit and Unigram fit as a feature available to the classifier; we call this difference in model fit the Log-Log score. We also consider a feature which is the product between the inverse document frequency (IDF) score and the Log-Log score[5]. Table 7 gives the results. The Log-Log score appears to be valuable for helping to extract restaurant names, as F1-breakeven is larger when the feature is included. However, a nonparametric test does not indicate that the improvement is significant. We see further improvement when the product between IDF and the Log-Log score is included. However, the improvement is somewhat less than what we saw when we used a product of IDF and the Mixture score as a feature for named entity extraction. While the Log-Log model provides good fit for topic-oriented words, it does not seem to be any more useful for identifying informative words than a mixture of Unigrams.

[5]For the "IDF*Log-Log" experiment, we also include features for the IDF score, the Log-Log score, the square of the IDF score and the square of the Log-Log score.

5 Conclusion

We have introduced the Log-Log model as a "heavy-tailed" alternative term frequency model to the Unigram. We found that it provided overall that was no better than that of the Unigram. The Log-Log was very effective at modeling topic-oriented, or informative words. But, in classification and named-entity extraction experiments, we were not able to make a compelling case for its use.

References

[1] J. D. M. Rennie and T. Jaakkola. Using term informativeness for named entity detection. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2005.