Empirical Term Weighting and Expansion Frequency

Kyoji Umemura
Toyohashi University of Technology
Toyohashi Aichi 441-8580 Japan
umemura@tutics.tut.ac.jp

Kenneth W. Church
AT&T Labs-Research
180 Park Ave., Florham Park, NJ.
kwc@research.att.com

Abstract

We propose an empirical method for estimating term weights directly from relevance judgements, avoiding various standard but potentially troublesome assumptions. It is common to assume, for example, that weights vary with term frequency (tf) and inverse document frequency (idf) in a particular way, e.g., $tf \cdot idf$, but the fact that there are so many variants of this formula in the literature suggests that there remains considerable uncertainty about these assumptions. Our method is similar to the Berkeley regression method where labeled relevance judgements are fit as a linear combination of (transforms of) tf, idf, etc. Training methods not only improve performance, but also extend naturally to include additional factors such as burstiness and query expansion. The proposed histogram-based training method provides a simple way to model complicated interactions among factors such as tf, idf, burstiness and expansion frequency (a generalization of query expansion). The correct handling of expanded term is realized based on statistical information. Expansion frequency dramatically improves performance from a level comparable to BKJJBIDS, Berkeley's entry in the Japanese NACSIS NTCIR-1 evaluation for short queries, to the level of JCB1, the top system in the evaluation. JCB1 uses sophisticated (and proprietary) natural language processing techniques developed by Just System, a leader in the Japanese word-processing industry. We are encouraged that the proposed method, which is simple to understand and replicate, can reach this level of performance.

1 Introduction

An empirical method for estimating term weights directly from relevance judgements is proposed. The method is designed to make as few assumptions as possible. It is similar to Berkeley's use of regression (Cooper et al., 1994) (Chen et al., 1999) where labeled relevance judgements are fit as a linear combination of (transforms of) tf, idf, etc., but avoids potentially troublesome assumptions by introducing histogram methods. Terms are grouped into bins. Weights are computed based on the number of relevant and irrelevant documents associated with each bin. The resulting weights usually lie between 0 and idf, which is a surprise; standard formulas like $tf \cdot idf$ would assign values well outside this range.

- t: a term
- d: a document
- $tf(t,d)$: term freq = # of instances of t in d
- $df(t)$: doc freq = # of docs d with $tf(t,d) \geq 1$
- N: # of documents in collection
- $idf(t)$: inverse document freq: $-log_2 \frac{df(t)}{N}$
- $df(t, rel, tf_0)$: # of relevant documents d with $tf(t,d) = tf_0$
- $df(t, \overline{rel}, tf_0)$: # of irrelevant documents d with $tf(t,d) = tf_0$
- $ef(t)$: expansion frequency = # docs d in query expansion with $tf(t,d) \geq 1$
- $TF(t)$: standard notion of frequency in corpus-based NLP: $TF(t) = \sum_d tf(t,d)$
- $B(t)$: burstiness: $B(t) = 1$ iff $\frac{TF(t)}{df(t)}$ is large.

Table 1: Notation

The method extends naturally to include additional factors such as query expansion. Terms mentioned explicitly in the query receive much larger weights than terms brought in via query expansion. In addition, whether or not a term t is mentioned explicitly in the query, if t appears in documents brought in by query expansion ($ef(t) \geq 1$) then t will receive a much larger weight than it would have otherwise ($ef(t) = 0$). The interactions among these factors, however, are complicated and collection dependent. It is safer to use histogram methods than to impose unnecessary and potentially troublesome assumptions such as normality and independence.

Under the vector space model, the score for a document d and a query q is computed by summing a contribution for each term t over an appropriate set of terms, T. T is often limited to terms shared by both the document and the query (minus stop words), though not always (e.g, query expansion).

$\overline{idf}$	$tf=0$	$tf=1$	$tf=2$	$tf=3$	$tf\geq 4$
12.89	-0.37	9.73	11.69	12.45	13.59
10.87	-0.49	8.00	9.95	11.47	12.06
9.79	-0.86	7.36	9.38	10.63	10.88
8.96	-0.60	6.26	7.99	8.99	9.41
7.75	-0.34	4.62	5.82	6.62	7.98
6.82	-1.26	3.94	6.05	7.59	8.98
5.78	-0.83	3.16	5.17	5.77	7.00
4.74	-0.84	2.46	3.91	4.54	5.58
3.85	-0.60	1.58	2.76	3.57	4.55
2.85	-1.02	1.00	1.72	2.55	3.96
1.78	-1.33	-0.06	1.05	2.46	4.50
0.88	-0.16	0.17	0.19	-0.10	-0.37

Table 2: Empirical estimates of $\hat{\lambda}$ as a function of tf and idf. Terms are assigned to bins based on idf. The column labeled $\overline{idf}$ is the mean idf for the terms in each bin. $\hat{\lambda}$ is estimated separately for each bin and each tf value, based on the labeled relevance judgements.

$$score_v(d,q) = \sum_{t \in T} tf(t,d) \cdot idf(t)$$

Under the probabilistic retrieval model, documents are scored by summing a similar contribution for each term t.

$$score_p(d,q) = \sum_{t \in T} log_2 \frac{P(t|rel)}{P(t|\overline{rel})}$$

In this work, we use λ to refer to term weights.

$$score(d,q) = \sum_{t \in T} \lambda(t,d,q)$$

This paper will start by showing how to estimate λ from relevance judgements. Three parameterizations will be considered: (1) fit-G, (2) fit-B, which introduces burstiness, and (3) fit-E, which introduces expansion frequency. The evaluation section shows that each model improves on the previous one. But in addition to performance, we are also interested in the interpretations of the parameters.

2 Supervised Training

The statistical task is to compute $\hat{\lambda}$, our best estimate of λ, based on a training set. This paper will use supervised methods where the training materials not only include a large number of documents but also a few queries labeled with relevance judgements.

To make the training task more manageable, it is common practice to map the space of all terms into a lower dimensional feature space. In other words, instead of estimating a different $\hat{\lambda}$ for each term in the vocabulary, we can model $\hat{\lambda}$ as a function of tf and idf and various other features of terms. In this way, all of the terms in a bin are assigned the weight, $\hat{\lambda}$. The common practice, for example, of assigning $tf \cdot idf$ weights can be interpreted as grouping all terms with the same idf into a bin and assigning them all the same weight, namely $tf \cdot idf$. Cooper and his colleagues at Berkeley (Cooper et al., 1994) (Chen et al., 1999) have been using regression methods to fit $\hat{\lambda}$ as a linear combination of idf, $log(tf)$ and various other features. This method is also grouping terms into bins based on their features and assigning similar weights to terms with similar features. In general, term weighting methods that are fit to data are more flexible than weighting methods that are not fit to data. We believe this additional flexibility improves precision and recall (table 8).

Instead of multiple regression, though, we choose a more empirical approach. Parametric as-

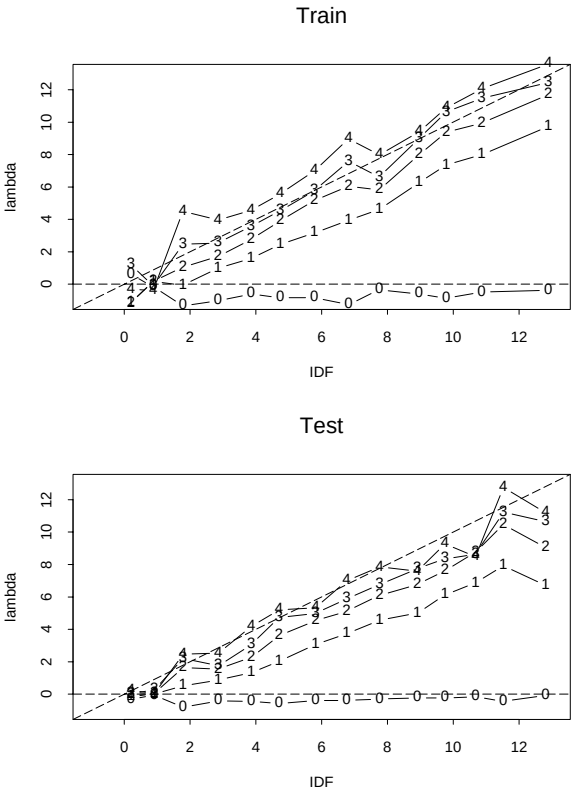

Figure 1: Empirical weights, $\hat{\lambda}$. Top panel shows values in previous table. Most points fall between the dashed lines (lower limit of $\hat{\lambda}=0$ and upper limit of $\hat{\lambda}=idf$). The plotting character denotes tf. Note that the line with $tf=4$ is above the line with $tf=3$, which is above the line with $tf=2$, and so on. The higher lines have larger intercepts and larger slopes than the lower lines. That is, when we fit $\hat{\lambda} \approx a(tf) + b(tf) \cdot idf$, with separate regression coefficients, $a(tf)$ and $b(tf)$, for each value of tf, we find that both $a(tf)$ and $b(tf)$ increase with tf.

	Description (function of term t)
1	$df(t, rel, 0) \equiv$ # rel docs d with $tf(t,d) = 0$
2	$df(t, rel, 1) \equiv$ # rel docs d with $tf(t,d) = 1$
3	$df(t, rel, 2) \equiv$ # rel docs d with $tf(t,d) = 2$
4	$df(t, rel, 3) \equiv$ # rel docs d with $tf(t,d) = 3$
5	$df(t, rel, 4+) \equiv$ # rel docs d with $tf(t,d) \geq 4$
6	$df(t, \overline{rel}, 0) \equiv$ # $\overline{rel}$ docs d with $tf(t,d) = 0$
7	$df(t, \overline{rel}, 1) \equiv$ # $\overline{rel}$ docs d with $tf(t,d) = 1$
8	$df(t, \overline{rel}, 2) \equiv$ # $\overline{rel}$ docs d with $tf(t,d) = 2$
9	$df(t, \overline{rel}, 3) \equiv$ # $\overline{rel}$ docs d with $tf(t,d) = 3$
10	$df(t, \overline{rel}, 4+) \equiv$ # $\overline{rel}$ docs d with $tf(t,d) \geq 4$
11	# rel docs d
12	# $\overline{rel}$ docs d
13	freq of term in corpus: $TF(t) = \sum_d tf(t,d)$
14	# docs d in collection $= N$
15	df = # docs d with $tf(t,d) \geq 1$
20	ef = # docs d in query exp. with $tf(t,d) \geq 1$
21	where: D (description), E (query expansion)
25	burstiness: B

Table 3: Training file schema: a record of 25 fields is computed for each term (ngram) in each query in training set.

sumptions, when appropriate, can be very powerful (better estimates from less training data), but errors resulting from inappropriate assumptions can outweigh the benefits. In this empirical investigation of term weighting we decided to use conservative non-parametric histogram methods to hedge against the risk of inappropriate parametric assumptions.

Terms are assigned to bins based on features such as idf, as illustrated in table 2. (Later we will also use B and/or ef in the binning process.) $\hat{\lambda}$ is computed separately for each bin, based on the use of terms in relevant and irrelevant documents, according to the labeled training material.

The estimation method starts with a training file which indicates, among other things, the number of relevant and irrelevant documents for each term t in each training query, q. That is, for each t and q, we are are given $df(t, rel, tf_0)$ and $df(t, \overline{rel}, tf_0)$, where $df(t, rel, tf_0)$ is the number of relevant documents d with $tf(t,d) = tf_0$, and $df(t, \overline{rel}, tf_0)$ is the number of irrelevant documents d with $tf(t,d) = tf_0$. The schema for the training file is described in table 3. From these training observations we wish to obtain a mapping from bins to λs that can be applied to unseen test material. We interpret λ as a log likelihood ratio:

$$\hat{\lambda}(bin, tf) = log_2 \frac{\hat{P}(bin, tf|rel)}{\hat{P}(bin, tf|\overline{rel})}$$

where the numerator can be approximated as:

$$\hat{P}(bin, tf|rel) \approx log_2 \frac{df(bin, rel, tf)}{\hat{N}_{rel}}$$

where $df(bin, rel, tf)$ is

$$df(bin, rel, tf) \equiv \frac{1}{|bin|} \sum_{t \in bin} df(t, rel, tf)$$

Similarly, the denominator can be approximated as:

$$\hat{P}(bin, tf|\overline{rel}) \approx log_2 \frac{df(bin, \overline{rel}, tf)}{\hat{N}_{\overline{rel}}}$$

where $df(bin, \overline{rel}, tf)$ is

$$df(bin, \overline{rel}, tf) \equiv \frac{1}{|bin|} \sum_{t \in bin} df(t, \overline{rel}, tf)$$

$\hat{N}_{rel}$ is an estimate of the total number of relevant documents. Since some queries have more relevant documents than others, $\hat{N}_{rel}$ is computed by averaging:

$$\hat{N}_{rel} \equiv \frac{1}{|bin|} \sum_{t \in bin} N_{rel}$$

To ensure that $\hat{N}_{rel} + \hat{N}_{\overline{rel}} = N$, where N is the number of documents in the collection, we define $\hat{N}_{\overline{rel}} \equiv N - \hat{N}_{rel}$

This estimation procedure is implemented with the simple `awk` program in figure 2. The `awk` program reads each line of the training file, which contains a line for each term in each training query. As described in table 3, each training line contains 25 fields. The first five fields contain $df(t, rel, tf)$ for five values of tf, and the next five fields contain $df(t, \overline{rel}, tf)$ for the same five values of tf. The next two fields contain N_{rel} and $N_{\overline{rel}}$. As the `awk` program reads each of these lines from the training file, it assigns each term in each training query to a bin (based on $\lceil log_2(df) \rceil$, except when $df < 100$), and maintains running sums of the first dozen fields which are used for computing $df(bin, rel, tf)$, $df(bin, \overline{rel}, tf)$, $\hat{N}_{rel}$ and $\hat{N}_{\overline{rel}}$ for five values of tf. Finally, after reading all the training material, the program outputs the table of $\hat{\lambda}$s shown in table 2. The table contains a column for each of the five tf values and a row for each of the dozen idf bins. Later, we will consider more interesting binning rules that make use of additional statistics such as burstiness and query expansion.

2.1 Interpolating Between Bins

Recall that the task is to apply the $\hat{\lambda}$s to new unseen test data. One could simply use the $\hat{\lambda}$s in table 2 as is. That is, when we see a new term in the test material, we find the closest bin in table 2 and report the corresponding $\hat{\lambda}$ value. But since the idf of a term in the test set could easily fall between two bins, it seems preferable to find the two closest bins and interpolate between them.

```
awk 'function log2(x) {
            return log(x)/log(2)}
 $21 ~ /^D/ { N = $14; df=$15;
  # binning rule
  if(df < 100) {bin = 0}
   else {bin=int(log2(df))};
  docfreq[bin] += df;
  Nbin[bin]++;
  # average df(t,rel,tf), df(t,irrel,tf)
  for(i=1;i<=12;i++) n[i,bin]+=$i }
END {for(bin in Nbin) {
 nbin = Nbin[bin]
 Nrel = n[11,bin]/nbin
 Nirrel = N-Nrel
 idf = -log2((docfreq[bin]/nbin)/N)
 printf("%6.2f ", idf)
 for(i=1;i<=5;i++) {
  if(Nrel==0) prel = 0
   else prel = (n[i,bin]/nbin)/Nrel
  if(Nirrel == 0) pirrel = 0
   else pirrel = (n[i+5,bin]/nbin)/Nirrel
  if(prel <= 0 || pirrel <= 0) {
    printf "%6s ", "NA" }
   else {
    printf "%6.2f ", log2(prel/pirrel)} }
  print ""}}'
```

Figure 2: `awk` program for computing $\hat{\lambda}$s.

We use linear regression to interpolate along the *idf* dimension, as illustrated in table 4. Table 4 is a smoothed version of table 2 where $\hat{\lambda} \approx a + b \cdot idf$. There are five pairs of coefficients, a and b, one for each value of tf.

Note that interpolation is generally not necessary on the tf dimension because tf is highly quantized. As long as $tf < 4$, which it usually is, the closest bin is an exact match. Even when $tf \geq 4$, there is very little room for adjustments if we accept the upper limit of $\hat{\lambda} < idf$.

Although we interpolate along the *idf* dimension, interpolation is not all that important along that dimension either. Figure 1 shows that the differences between the test data and the training data dominate the issues that interpolation is attempting to deal with. The main advantage of regression is computational convenience; it is easier to compute $a + b \cdot idf$ than to perform a binary search to find the closest bin.

Previous work (Cooper et al., 1994) used multiple regression techniques. Although our performance is similar (until we include query expansion) we believe that it is safer and easier to treat each value of tf as a separate regression for reasons discussed in table 5. In so doing, we are basically restricting the regression analysis to such an extent that it is unlikely to do much harm (or much good). Imposing the limits of $0 \leq \hat{\lambda} \leq idf$ also serves the purpose of preventing the regression from wandering too far astray.

tf	a	b
0	-0.95	0.05
1	-0.98	0.69
2	-0.15	0.78
3	0.53	0.81
4+	1.32	0.77

Table 4: Regression coefficients for method fit-G. This table approximates the data in table 1 with $\hat{\lambda} \approx a(tf) + b(tf) \cdot idf$. Note that both the intercepts, $a(tf)$, and the slopes, $b(tf)$, increase with tf (with a minor exception for $b(4+)$).

tf	$a(tf)$	$b(tf)$	$a_2 + c_2 \cdot log(1+tf)$	b_2
0	-0.95	0.05	-4.1	0.66
1	-0.98	0.69	-1.4	0.66
2	-0.15	0.78	0.18	0.66
3	0.53	0.81	1.3	0.66
4	1.32	0.77	2.2	0.66
5	1.32	0.77	2.9	0.66

Table 5: A comparison of the regression coefficients for method fit-G with comparable coefficients from the multiple regression: $\hat{\lambda} = a_2 + b_2 \cdot idf + c_2 \cdot log(1+tf)$ where $a_2 = -4.1$, $b_2 = 0.66$ and $c_2 = 3.9$. The differences in the two fits are particularly large when $tf = 0$; note that $b(0)$ is negligible (0.05) and b_2 is quite large (0.66). Reducing the number of parameters from 10 to 3 in this way increases the sum of square errors, which may or may not result in a large degradation in precision and recall. Why take the chance?

3 Burstiness

Table 6 is like tables 4 but the binning rule not only uses *idf*, but also burstiness (B). Burstiness (Church and Gale, 1995)(Katz, 1996)(Church, 2000) is intended to account for the fact that some very good keywords such as "Kennedy" tend to be mentioned quite a few times in a document or not at all, whereas less good keywords such as "except" tend to be mentioned about the same number of times no matter what the document

	B=0		B=1	
tf	a	b	a	b
0	-0.05	-0.00	-0.61	0.02
1	-1.23	0.63	-0.80	0.79
2	-0.76	0.71	-0.05	0.79
3	0.00	0.69	0.23	0.82
4+	0.68	0.71	0.75	0.83

Table 6: Regression coefficients for method fit-B. Note that the slopes and intercepts are larger when $B = 1$ than when $B = 0$ (except when $tf = 0$). Even though $\hat{\lambda}$ usually lies between 0 and idf, we restrict $\hat{\lambda}$ to $0 \leq \hat{\lambda} \leq idf$, just to make sure.

		where=D		where=E	
tf	ef	a	b	a	b
1	0	-1.57	0.37		
2	0	-3.41	0.82		
3	0	-1.30	0.11		
4+	0	0.40	0.06		
1	1	-1.84	0.87	-2.64	0.68
2	1	-2.12	1.10	-2.70	0.71
3	1	-0.66	0.95	-2.98	0.74
4+	1	0.84	0.98	-3.35	0.78
1	2	-1.87	0.92	-3.00	0.86
2	2	-1.77	1.12	-2.78	0.85
3	2	-1.72	1.10	-3.07	0.93
4+	2	-3.06	1.71*	-3.25	0.79
1	3	-2.52	0.95	-2.71	0.91
2	3	-1.81	1.02	-2.28	0.88
3	3	0.45	0.85	-2.63	0.97
4+	3	0.38	1.22	-3.66	1.14

Table 7: Many of the regression coefficients for method fit-E. (The coefficients marked with an asterisk are worrisome because the bins are too small and/or the slopes fall well outside the normal range of 0 to 1.) The slopes rarely exceeded .8 is previous models (fit-G and fit-B), whereas fit-E has more slopes closer to 1. The larger slopes are associated with robust conditions, e.g., terms appearing in the query ($where = D$), the document ($tf > 1$) and the expansion ($ef \geq 1$). If a term appears in several documents brought in by query expansion ($ef \geq 2$), then the slope can be large even if the term is not explicitly mentioned in the query ($where = E$). The interactions among tf, idf, ef and $where$ are complicated and not easily captured with a straightforward multiple regression.

is about. Since "Kennedy" and "except" have similar idf values, they would normally receive similar term weights, which doesn't seem right. Kwok (1996) suggested average term frequency, $avtf = TF(t)/df(t)$, be used as a tie-breaker for cases like this, where $TF(t) = \sum_d tf(t,d)$ is the standard notion of frequency in the corpus-based NLP. Table 6 shows how Kwok's suggestion can be reformulated in our empirical framework. The table shows the slopes and intercepts for ten regressions, one for each combination of tf and B ($B = 1$ iff $avtf$ is large. That is, $B = 1$ iff $TF(t)/df(t) > 1.83 - 0.048 \cdot idf$).

4 Query Expansion

We applied query expansion (Buckley et al., 1995) to generate an expanded part of the query. The original query is referred to as the description (D) and the new part is referred to as the expansion (E). (Queries also contain a narrative (N) part that is not used in the experiments below so that our results could be compared to previously published results.)

The expansion is formed by applying a baseline query engine (fit-B model) to the description part of the query. Terms that appear in the top $k = 10$ retrieved documents are assigned to the E portion of the query ($where(t) = E$), unless they were previously assigned to some other portion of the query (e.g., $where(t) = D$). All terms, t, no matter where they appear in the query, also receive an expansion frequency ef, an integer from 0 to $k = 10$ indicating how many of the top k documents contain t.

The fit-E model is: $\hat{\lambda} = a(tf, where, ef) + b(tf, where, ef) \cdot idf$, where the regression coefficients, a and b, not only depend on tf as in fit-G, but also depend on where the term appears in the query and expansion frequency ef. We consider 5 values of tf, 2 values of $where$ (D and E) and 6 values of ef (0, 1, 2, 3, 4 or more). 32 of these 60 pairs of coefficients are shown in table 7. As before, most of the slopes are between 0 and 1. $\hat{\lambda}$ is usually between 0 and idf, but we restrict $\hat{\lambda}$ to $0 \leq \hat{\lambda} \leq idf$, just to make sure.

In tables 4-7, the slopes usually lie between 0 and 1. In the previous models, fit-B and fit-G, the largest slopes were about 0.8, whereas in fit-E, the slope can be much closer to 1. The larger slopes are associated with very robust conditions, e.g., terms mentioned explicitly in all three areas of interest: (1) the query ($where = D$), (2) the document ($tf \geq 1$) and (3) the expansion ($ef \geq 1$). Under such robust conditions, we would expect to find very little *shrinking* (downweighting to compensate for uncertainty).

On the other hand, when the term is not mentioned in one of these areas, there can be quite a bit of shrinking. Table 7 shows that the slopes are generally much smaller when the term is not in the query ($where = E$) or when the term is not in the expansion ($ef = 0$). However, there are some exceptions. The bottom right corner of table 7 contains some large slopes even though these terms are not mentioned explicitly in the query ($where = E$). The mitigating factor in this case is the large ef. If a term is mentioned in several documents in the expansion ($ef \geq 2$), then it is not as essential that it be mentioned explicitly in the query.

With this model, as with fit-G and fit-B, $\hat{\lambda}$ tends to increase monotonically with tf and idf, though there are some interesting exceptions. When the term appears in the query ($where = D$) but not in the expansion ($ef = 0$), the slopes are quite small (e.g., $b(3, D, 0) = 0.11$), and the slopes actually decrease as tf increases ($b(2, D, 0) = 0.83 > b(3, D, 0) = 0.11$). We normally expect to see slopes of .7 or more when $tf \geq 3$, but in this case ($b(3, D, 0) = 0.11$), there is a considerable shrinking because we very much expected to see the term in the expansion and we didn't.

As we have seen, the interactions among tf, idf, ef and $where$ are complicated and probably de-

filter	trained on	sys.	11	R
NA	?	JCB1	.360	.351
2+, E_1	tf,where,ef	fit-E	.354	.363
2	B,tf	fit-B	.283	.293
2, K	tf + ...	BKJJBIDS	.272	.282
2, K	B,tf	fit-B	.264	.282
2, K	tf	fit-G	.257	.267
2, K	none	$log(1+tf) \cdot idf$	.249	.262
2, K	none	$tf \cdot idf$	.112	.138

Table 8: Training helps: methods above the line use training (with the possible exception of JCB1); methods below the line do not.

pend on many factors such as language, collection, typical query patterns and so on. To cope with such complications, we believe that it is safer to use histogram methods than to try to account for all of these interactions at once in a single multiple regression. The next section will show that fit-E has very encouraging performance.

5 Experiments

Two measures of performance are reported: (1) 11 point average precision and (2) R, precision after retrieving N_{rel} documents, where N_{rel} is the number of relevant documents. We used the "short query" condition of the NACSIS NTCIR-1 Test Collection (Kando et al., 1999) which consists of about 300,000 documents in Japanese, plus about 30 queries with labeled relevance judgement for training and 53 queries with relevance judgements for testing. The result of "short query" is shown in page 25 of(Kando et al., 1999), which shows that "short query" is hard for statistical methods.

Two previously published systems are included in the tables below: JCB1 and BKJJBIDS. JCB1, submitted by Just System, a company with a commercially successful product for Japanese word-processing, produced the best results using sophisticated (and proprietary) natural language processing techniques.(Fujita, 1999) BKJJBIDS used Berkeley's logistic regression methods (with about half a dozen variables) to fit term weights to the labeled training material.

Table 8 shows that training often helps. The methods above the line (with the possible exception of JCB1) use training; the methods below the line do not. Fit-E has very respectable performance, nearly up to the level of JCB1, not bad for a purely statistical method.

The performance of fit-B is close to that of BKJJBIDS. For comparison sake, fit-B is shown both with and without the K filter. The K filter restricts terms to sequences of Katakana and Kanji characters. BKJJBIDS uses a similar heuristic to eliminate Japanese function words. Although the K filter does not change performance very much, the use of this filter changes the relative order of fit-B and BKJJBIDS. These results suggest that the K filter is slightly unhelpful.

- 2: restrict terms to bigrams explicitly mentioned in query ($where = D$)
- 2+: restrict terms to bigrams, but include $where = E$ as well as $where = D$
- W: restrict terms to words, as identified by Chasen (Matsumoto et al., 1997)
- K: restrict terms to sequences of Katakana and/or Kanji characters
- B: restrict terms to bursty ($B = 1$) terms
- E_k: require terms to appear in more than k docs brought in by query expansion ($ef(t) > k$).

Table 9: Filters: results vary somewhat depending on these choices, though not too much, which is fortunate, since since we don't understand stop lists very well.

filter	trained on	sys.	11	R
2+, E_1	tf,where,ef	fit-E	.354	.363
2+, E_2	tf,where,ef	fit-E	.350	.359
2+, E_4	tf,where,ef	fit-E	.333	.341
2+	tf,where,ef	fit-E	.332	.366
NA	NA	JCB1	.360	.351

Table 10: The best filters (E_k) improve the performance of the best method (fit-E) to nearly the level of JCB1.

A number of filters have been considered (table 9). Results vary somewhat depending on these choices, though not too much, which is fortunate, since since we don't understand stop lists very well. To the extent that there is a pattern, we suspect that words are slightly better than bigrams, and that the E filter is slightly better than the B filter which is slightly better than the K filter. Table 10 shows that the best filters (E_k) improve the performance of the best method (fit-E) to nearly the level of JCB1.

filter	sys.	UL	LL	11	R
2	fit-B	+	+	.283	.293
2	fit-B	+	−	.280	.296
2	fit-B	−	+	.280	.296
2	fit-B	−	−	.275	.288
2	fit-G	+	+	.266	.279
2	fit-G	−	+	.251	.268
2	fit-G	+	−	.248	.259
2	fit-G	−	−	.232	.249

Table 11: Limits do no harm: two limits are slightly better than one, and one is slightly better than none. (UL = upper limit of $\hat{\lambda} \leq idf$; LL = lower limit of $0 \leq \hat{\lambda}$)

The final experiment (table 11) shows that restricting $\hat{\lambda}$ to $0 \leq \hat{\lambda} \leq idf$ improves performance slightly. The combination of both the upper limit and the lower limit is slightly better than just one limit which is better than none. We view limits as a robustness device. Hopefully, they won't have to do much but every once in a while they prevent the system from wandering far astray.

6 Conclusions

This paper introduced an empirical histogram-based supervised learning method for estimating term weights, $\hat{\lambda}$. Terms are assigned to bins based on features such as inverse document frequency, burstiness and expansion frequency. A different $\hat{\lambda}$ is estimated for each bin and each tf by counting the number of relevant and irrelevant documents associated with the bin and tf value. Regression techniques are used to interpolate between bins, but care is taken so that the regression cannot do too much harm (or too much good). Three variations were considered: fit-G, fit-B and fit-E. The performance of query expansion (fit-E) is particularly encouraging. Using simple purely statistical methods, fit-E is nearly comparable to JCB1, a sophisticated natural language processing system developed by Just System, a leader in the Japanese word processing industry.

In addition to performance, we are also interested in the interpretation of the weights. Empirical weights tend to lie between 0 and idf. We find these limits to be a surprise given that standard term weighting formulas such as $tf \cdot idf$ generally do not conform to these limits. In addition, we find that $\hat{\lambda}$ generally grows linearly with idf, and that the slope is between 0 and 1. We interpret the slope as a statistical shrink. The larger slopes are associated with very robust conditions, e.g., terms mentioned explicitly in all three areas of interest: (1) the query ($where = D$), (2) the document ($tf \geq 1$) and (3) the expansion ($ef \geq 1$). There is generally more shrinking for terms brought in by query expansion ($where = E$), but if a term is mentioned in several documents in the expansion ($ef \geq 2$), then it is not as essential that the term be mentioned explicitly in the query. The interactions among tf, idf, $where$, B, ef, etc., are complicated, and therefore, we have found it safer and easier to use histogram methods than to try to account for all of the interactions at once in a single multiple regression.

Acknowdedgement

Authors thank Prof. Mitchell P. Marcus of University of Pennsylvania for the valuable discussion about noise reduction in context of information retrieval. This reseach is supported by Sumitomo Electric.

References

Chris Buckley, Gerard Salton, James Allan, and Amit Singhal. 1995. Automatic query expansion using smart: Trec 3. In *The Third Text REtrieval Conference(TREC-3)*, pages 69–80.

Aitao Chen, Fredric C. Gey, Kazuaki Kishida, Hailing Jiang, and Qun Liang. 1999. Comparing multiple methods for japanese and japanese-english text retrieval. In *NTCIR Workshop 1*, pages 49–58, Tokyo Japan, Sep.

Kenneth W. Church and William A. Gale. 1995. Poisson mixture. *Natural Language Engineering*, 1(2):163–190.

Kenneth W. Church. 2000. Empirical estimates of adaptation: The chance of two noriegas is closer to $p/2$ than p^2. In *Coling-2000*, pages 180–186.

William S. Cooper, Aitao Chen, and Fredric C. Gey. 1994. Full text retrieval based on probabilistic equation with coefficients fitted by logistic regressions. In *The Second Text REtrieval Conference(TREC-2)*, pages 57–66.

Sumio Fujita. 1999. Notes on phrasal indexing: Jscb evaluation experiments at ntcir ad hoc". In *NTCIR Workshop 1*, pages 101–108, http://www.rd.nacsis.ac.jp/ ~ntcadm/, Sep.

Noriko Kando, Kazuko Kuriyama, Toshihiko Nozue, Koji Eguchi, and Hiroyuki Katoand Souichiro Hidaka. 1999. Overview of ir tasks at the first ntcir workshop. In *NTCIR Workshop 1*, pages 11–44, http://www.rd.nacsis.ac.jp/ ~ntcadm/, Sep.

Slava M. Katz. 1996. Distribution of content words and phrases in text and language modelling. *Natural Language Engineering*, 2(1):15–59.

K. L. Kwok. 1996. A new method of weighting query terms for ad-hoc retrieval. In *SIGIR96*, pages 187–195, Zurich, Switzerland.

Yuji Matsumoto, Akira Kitauchi, Tatsuo Yamashita, Yoshitaka Hirano, Osamu Imaichi, and Tomoaki Imamura. 1997. Japanese morphological analysis system chasen manual. Technical Report NAIST-IS-TR97007, NAIST, Nara, Japan, Feb.