Empirical Term Weighting

Kyoji UMEMURA Yoshiyuki TAKEDA Michiko TANAKA Lin FENG Eiko YAMAMOTO
Toyohashi University of Technology
Toyohashi Aichi 441-8580 Japan
ulab@ss.ics.tut.ac.jp

Abstract

Our system used 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 a kind of 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 factor, that is burstiness. The proposed histogram-based training method provides a simple way to model complicated interactions among factors such as tf and idf.

1 Introduction

An empirical method for estimating term weights directly from relevance judgements is proposed by [7]. The method is designed to make as few assumptions as possible. It is a kind of regression [4] [1] 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.

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).

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

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, and (2) fit-B, which introduces burstiness. We are also interested in the interpretations of the parameters.

In this report, we have borrowed the description from [7], since our system uses the method described in the paper. The following description is essentially same as [7] except the description of query expansion.

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.

We have chosen bigram (2 characters) for term. Though properly segmented words will be another obvious choice, we are interested in bigram system.

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 $tf \cdot idf$

- 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 $tf(t,d) = tf_0$
- $TF(t)$: 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

$\overline{idf}$	$tf=0$	$tf=1$	$tf=2$	$tf=3$	$tf \geq 4$
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.

	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 $tf(t,d) \geq 1$
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.

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 [4] [1] 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.

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

ber 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.

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 $b(4+)$).

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. 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 [4] used multiple regression techniques. Although our performance is similar with the same number of parameters. 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.

```
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 pir-
rel = (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.**

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?

	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.

3 Burstiness

Table 6 is like tables 4 but the binning rule not only uses idf, but also burstiness (B). Burstiness [3][5][2] 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 is about. Since "Kennedy" and "except" have similar idf values, they would normally receive similar term weights, which doesn't seem right. Kwok [6] 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 Summary

Our system uses 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, and burstiness. 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).

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. It is interesting to compare histogram methods with multiple regression that tries to account for all of the interactions at once in a single multiple regression.

Even though selecting the effective parameters is still a kind of art, this framework gives a systematic way to handle them. We have shown that this assumption re-invent standard practice, we are encouraged with this result.

Acknowdedgement

Authors thank Kenneth W. Church, AT&T, to provide the fundamental framework used in our system. This development is supported by Sumitomo Electric.

References

[1] A. Chen, F. C. Gey, K. Kishida, H. Jiang, and Q. Liang. Comparing multiple methods for japanese and japanese-english text retrieval. In *NTCIR Workshop 1*, pages 49–58, Tokyo Japan, Sep. 1999.

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

[3] K. W. Church and W. A. Gale. Poisson mixture. *Natural Language Engineering*, 1(2):163–190, 1995.

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

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

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

[7] K. Umemura and K. W. Church. Empirical term weighting and expansion frequency. In *ACL SIGDAT*, pages 117–123, 2000.

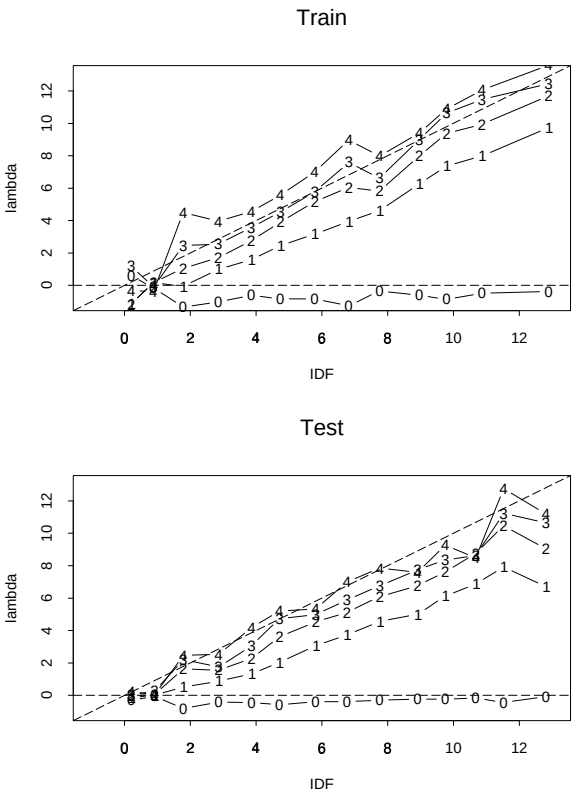

Figure 1. Empirical weights, $\hat{\lambda}$. Top panel shows values in previous table. Most points fall between the dashed lines (lower limit of $\hat{\lambda} = 0$ and upper limit of $\hat{\lambda} = idf$). The plotting character denotes tf. Note that the line with $tf = 4$ is above the line with $tf = 3$, 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.