DUTH does Probabilities of Relevance at the Legal Track *

Dim P. Papadopoulos **Vicky S. Kalogeiton** **Avi Arampatzis**

Department of Electrical and Computer Engineering,
Democritus University of Thrace,
Xanthi 67100, Greece.
{dimipapa4,vasikalo,avi}@ee.duth.gr

Abstract

We participated in the Learning Task of the TREC 2010 Legal Track, focusing solely on estimating probabilities of relevance. We submitted three automated runs based on the same tf.idf ranking, produced by the topic narratives and positive-only feedback of the training data in equal contributions. The runs differ in the way the probabilities of relevance are estimated: (1) `DUTHsdtA` employed the Truncated Normal-Exponential model to turn scores to probabilities. (2) `DUTHsdeA` did not assume any specific component score distributions but estimated those on the scores of training data via Kernel Density Estimation (KDE) methods. (3) `DUTHlrgA` used Logistic Regression with the co-efficients estimated on the scores of training data. We found that `DUTHsdeA` and `DUTHlrgA` are greatly affected by biases in the training set, since they assume that input score data are uniformly sampled. Also, KDE was found to be very sensitive to its parameters, influencing greatly the probability estimates. In these respects, `DUTHsdtA` was proven to be the most robust method.

1 Introduction

In this paper, we report our participation in the Learning Task of the TREC 2010 Legal Track. Task participants were given a "seed" (or training) set of documents from a larger collection that had previously been assessed by TREC as responsive or non-responsive (or—using traditional IR jargon—relevant or non-relevant) to a legal discovery request. Using this information, participants had (a) to rank the documents in the larger collection from most likely to least likely to be relevant, and (b) for each document, to estimate the likelihood of relevance as a probability. TREC assessed the quality of rankings, as well as the quality of probability estimates.

This year we focused solely on estimating probabilities of relevance. In Section 2, we talk about the task setup. The indexing, training, and retrieval methods are mentioned in Section 3. In Section 4, we describe the methods for estimating probabilities and their underlying assumptions. In Section 5, we provide a comparative evaluation of the methods in the context of the Learning Task. Conclusions are drawn in Section 6.

2 Task Setup

In this section, we briefly describe the datasets.

2.1 Test Collection

The Learning Task in the TREC 2010 Legal Track used the EDRM Enron v2 collection in either XML or PST formats. In the EDRM XML, a document is considered to be either an email message as a whole, or a particular attachment. Some documents may appear more than once in the original collection. We used the de-duplicated version of the text-only portion of XML dataset. This version has a total number of 685,592 documents.

2.2 Queries and Training Data

TREC provided to the participants a set of eight topics (200–207), their narrative parts only. Each topic was con-

*In: *Proceedings of the Nineteenth Text REtrieval Conference*, 2010.

sidered as an independent query. Also, as already mentioned, TREC provided a training set of documents that had previously been assessed as relevant or non-relevant. The numbers of relevant and non-relevant documents per topic are given in Table 1.

Table 1: Original/given seed set

Topic	Relevant	Non-relevant
200	230	620
201	168	523
202	1006	403
203	67	892
204	59	1132
205	333	1506
206	19	336
207	80	511

During the Track, conflicting relevance judgments were discovered for some documents which appeared both as relevant and non-relevant. TREC did not provide any guideline on how to deal with those, so we decided to take them as relevant. Thus, our numbers of training documents were effectively those in Table 2.

Table 2: Effective seed set

Topic	Relevant	Non-relevant
200	230	596
201	168	520
202	994	396
203	67	876
204	59	1122
205	333	1496
206	19	323
207	80	492

We note here that the given set of training data may be biased in unknown ways; this seems to have affected two of our three runs, as we will later see. Usually, such training sets are build by the taking the union of top-ranked documents retrieved by several systems (i.e. known as *pooling*), assessed in previous experiments; they are certainly not uniform samples of the whole collection.

3 Indexing and Retrieval

The EDRM Enron v2 collection was indexed with the Lemur Toolkit V4.11 and Indri V2.11, using the default settings that come with these versions, except that we enabled the Krovetz stemmer.[1] For the official runs, we used the tf.idf retrieval model.

In order to utilize the training set, we used the relevance feedback option in Lemur. All the feedback algorithms currently in Lemur assume that all entries in a judgment file are relevant documents, so we had to remove all the entries of judged non-relevant documents. We used the feedback model as well as the initial query (feedbackPosCoef=1), equally weighted. We added 64 terms to the initial query (feedbackTermCount=64).

4 Estimating Probabilities of Relevance

We investigated three methods for estimating the probabilities of relevance; the methods are based solely on document scores. The first method can be applied with or without training data, while the other two need training data. All methods are fully automated, i.e. they do not need any human intervention.

4.1 The SD Method with Theoretical Distributions

Under the assumption of binary relevance, classic attempts model score distributions (SDs)—on a per-request basis—as a mixture of two SDs: one for relevant $P(s|1)$ and the other for non-relevant document scores $P(s|0)$. Given the two component SDs and their mix weight G, the probability of relevance of a document given its score s can be calculated straightforwardly [4, 7]:

$$P(1|s) = \frac{G\,P(s|1)}{G\,P(s|1) + (1-G)\,P(s|0)} \qquad (1)$$

Various combinations of *theoretical* distributions have been proposed for modeling $P(s|1)$ and $P(s|0)$ since the early years of IR; for a recent review of the proposed combinations, we refer the reader to [3]. We settled for employing the latest, improved normal-exponential model which uses truncated versions of the component densities trying to deal with some of the shortcomings of the original model [2]. The last two mentioned studies suggest that normal component for relevant fits best to (a) vector

[1] http://www.lemurproject.org

space or geometrical retrieval models, (b) scoring functions in the form of linear combination of document term weights, and (c) long queries. The exponential fits best on the top-end of the non-relevant scores.

Throughout the rest of the paper, we refer to this method as SDT, i.e. Score-Distributional with Theoretical distributions, irrespective of the component choices; our officially-submitted run with normal-exponential is tagged as `DUTHsdtA`. The underlying assumption of SDT is not only that a chosen pair of theoretical distributions provides a good fit to observed SDs of the retrieval model at hand, but also that the components indeed represent binary relevance and not some arbitrary effect. Furthermore, the non-relevant component we chose applies to the top-end of scores, leaving us with no $P(1|s)$ estimates for lower scores. However, the method provides an estimate of the number of relevant documents below the truncation, and using this we simply set $P(1|s)$ uniformly to that estimate divided by the number of documents below the truncation. In summary, there are 4 parameters to estimate: the mean and variance of the normal, the mean of the exponential, and the mix weight G. This can be done with or without training data.

We recovered the parameters of the component distributions and the mix weight with Expectation Maximization (EM) [8] without using the training data. Specifically, we applied the method as described in [2], using the Technical Truncation model, with the following differences: (1) We truncated the rankings at the top 2% of the corpus (i.e. 13,712 documents) in order to achieve better exponential fits; not having an estimate of the average query generality, this was an arbitrary choice. (2) We did a minimum of 8 and a maximum of 64 EM runs, with a cap at 32 iterations per run; thus we used lower values which exchange accuracy for speed. (3) In calculating the χ^2 of the fits, score data were binned into a minimum of 8 to a maximum of 32 bins. (4) We rejected fits with an expected relevant score lower than the expected non-relevant score in the truncated rankings; this is reasoned in [1] as a condition for improving parameter estimation.

An important disadvantage of the currently used parameter estimation method is that it is not using the available training data. Although EM can be modified to do that, it is currently unclear to us how to do it with data of unknown biases.

4.2 The SD Method with Empirical Distributions

We experimented with a new score-distributional method which needs training data. The advantage of this method is that it does not assume any specific theoretical distributions, and it is capable of approximating unknown distributions. The components are deduced, one at a time, with Kernel Density Estimation (KDE) methods [9] from the corresponding scores of the training documents. The mix weight may be estimated from or without training data.

Throughout this paper, we refer to this method as SDE, i.e. Score-Distributional with Empirical distributions; our officially-submitted run is `DUTHsdeA`. A disadvantage of the method is that KDE needs some score data per component, which in general may not be available. Also, depending on the shapes of the estimated densities and the mix weight, SDE may not result to a monotonic transformation of scores to probabilities. This implies that rankings are sub-optimal and that they can be improved by simply re-ranking them in a descending order of the estimated probabilities. Rather than reversing the rankings, randomizing the 'offending' score ranges could also be effective. However, we did something rougher: we forced a monotonic decline of the probability of relevance in rankings by setting it to the minimum value previously seen as we go down a ranking.

Kernel density estimation (KDE) is a non-parametric way of estimating the probability density function of a random variable; it is a fundamental data-smoothing problem, where inferences about the population are made based on a finite data sample. We used a Gaussian kernel and a bandwidth of $\sigma/5$, where σ is the standard deviation of a 2% uniform score sample of a query's results (i.e. approximately 13,712 scores—we down-sampled for speed). The bandwidth is a free parameter which exhibits a strong influence on the resulting estimate.

Using EM, we only recovered the mix weight G of the two distributions on the whole score range. Again, for efficiency reasons, we run EM on a 2% uniform sample from the total distribution and not the whole collection. As theoretical component distributions in EM, we plugged in the KDEs of relevant and non-relevant. The function that provides the probability of relevance is again Equation 1.

Finally, it should be mentioned that the method assumes that the training data are a uniform sample of the

collection, so that the score densities estimated with KDE are representative.

4.3 Logistic Regression

Without recovering the component distributions, we mapped scores to probabilities directly by using the standard method of Logistic Regression [5] on the scores of the seed set. We refer to this method as LRG, and our officially-submitted run is `DUTHlrgA`.

This approach is non-parametric—that is, it is not dependent on any assumptions about or analysis of score distributions. However, it is not applicable if there are no training data, and the coefficients estimated depend heavily on the choice of the training sample [6].

Logistic regression has two coefficients, β_1 and β_2. In order to recover them we used once more the training data. We applied the Newton-Raphson algorithm to calculate maximum likelihood estimates of a simple logistic regression, using the training data as input. In the end, each score s was mapped to a probability of relevance according to:

$$P(1|s) = \frac{\exp(\beta_1 + \beta_2\, s)}{1 + \exp(\beta_1 + \beta_2\, s)} \qquad (2)$$

An obvious problem of the method is that the training data counts are not representative of the relative density of relevant to non-relevant in the collection. In order to compensate for this, trying to remove some of the bias, we assumed an extra 100,000 non-relevant documents scoring at 0 during the estimation of the co-efficients.

Logistic regression predicts the probability of occurrence of an event by fitting a logistic curve to the data. But, this curve is defined on the whole real axis. So, the tf.idf model used may not be appropriate, and maybe it would have been better to apply the method on OKAPI scores since OKAPI scores fall on the whole real axis and not only to the positive one.

5 Experiments

First, in Table 3, we investigate the quality of our ranking. Note that we used the same ranking for all three submitted runs. The hypothetical F1 is a measure of the quality of the ranking, independent of the submitted probability estimates. It is the F1 that would have been achieved, had the best cutoff been chosen for the ranking.

We are above median only in 205, which is also very close to the best result from all participating systems. We are at the median in 200 and 203. We are below the median in the rest 5 topics, but nowhere the worst. A hypothetical system with median performance in all topics would have achieved an average hypothetical F1 of 21.86; the quality of our ranking is close to this. All in all, we consider our ranking quality to be median, i.e. a good representative of all participating rankings.

In Table 4, we investigate the quality of the R estimates for all three methods. While, in principle, R can be estimated in a way that is independent of the quality of the underlying ranking, some estimation methods may be influenced by the ranking quality. For example, sdt is known to perform better on good quality queries or results [3].

For lrg, half the topics are above median, and the other half are below; 203 has the best R estimate of all participating runs. Its average accuracy is above this of the hypothetical median system, thus, we can say that lrg performs better than median. The sde method fails with all topics performing below the median, and 3 of them have the worst submitted accuracy in estimated R. The sdt method shows a great variance in its effectiveness: while it works great for some topics (best estimates in 204 and 206), it performs below the median in others; 207 is nearly the worst. Overall, the average R accuracy of sdt is above this of the hypothetical median system. It is worth mentioning that the best two R estimates are on the topics with the worst ranking quality compared to the other topics in our ranking.

In Table 5, we investigate the % accuracy of the F1 estimate. F1 is the score that would have been achieved if the estimates had been relied on selecting the cutoff. Using the same way of analysis as for the first table, we can deduce that our three methods present worse results that a median system for the accuracy of the F1 estimates. So, we have a kind of good R estimates for lrg and sdt, but they don't translate to good F1 estimates.

In Table 6, we investigate the % accuracy of our cutoffs. The average accuracy of all our methods is below the median. lrg and sdt are very close to the median but as far as the sde is concerned, three topics (203, 204, 205) have the worst submitted accuracy of the K estimates.

In summary, our ranking quality is close to this of a median system. The lrg and sdt estimate the number of relevant documents better than a median system, but the

Table 3: Ranking quality.

	200	201	202	203	204	205	206	207	avg.
hyp. F1	8.9	11.6	32.1	24.2	8.2	51.4	7.6	16.7	20.08
best	25.8	43.0	70.6	39.4	26.6	52.1	37.0	90.3	—
median	8.9	14.1	38.2	24.2	10.3	47.2	12.9	19.1	21.86
worst	1.8	1.5	4.5	3.2	5.3	18.0	3.4	6.7	—

Table 4: % accuracy of R estimates.

	200	201	202	203	204	205	206	207	avg.
lrg	14.5	39.4	27.3	91.2	40.6	29.3	5.2	78.3	40.73
sde	4.1	21.9	24.3	0.9	1.3	1.0	3.1	18.6	9.40
sdt	15.4	18.1	14.7	22.9	99.4	16.4	77.4	9.9	34.28
best	75.3	93.7	93.3	91.2	99.4	69.4	77.4	89.4	—
median	15.4	28.2	48.8	39.9	49.7	31.1	4.8	32.0	31.24
worst	0.9	0.8	2.0	0.9	1.3	1.0	0.3	7.2	—

Table 5: % accuracy of the $F1$ estimates.

	200	201	202	203	204	205	206	207	avg.
lrg	8.2	13.1	23.6	42.4	15.3	36.6	4.7	15.9	19.98
sde	36.6	74.3	33.6	5.9	2.0	1.8	6.3	20.8	22.66
sdt	12.0	7.2	27.7	32.4	6.0	29.6	11.0	15.5	17.68
best	98.8	97.4	91.4	93.9	79.5	95.5	34.6	94.9	—
median	12.5	19.8	23.6	29.8	15.3	59.0	11.3	38.1	26.18
worst	1.5	1.3	2.5	2.4	2.0	1.8	0.4	2.9	—

Table 6: % accuracy of the K estimates.

	200	201	202	203	204	205	206	207	avg.
lrg	18.8	17.8	12.0	53.4	5.7	27.5	1.1	30.8	20.89
sde	11.6	22.0	39.3	0.3	0.0	0.5	0.9	21.1	11.97
sdt	26.2	12.3	39.3	28.2	10.6	15.0	5.2	20.9	19.71
best	90.4	87.6	99.8	78.1	89.5	91.2	26.8	99.3	—
median	11.6	12.3	32.6	37.8	15.6	42.4	4.7	25.7	22.84
worst	0.3	0.7	0.6	0.3	0.0	0.5	0.1	1.3	—

sde lags far behind. In the accuracy of F1 estimates, all methods show a comparable performance which is worse than a median system. As far as the accuracy of K estimates is concerned, the best of our methods is lrg, with sdt close behind and both around a median system, but sde fails again. Overall, we seem to have a problem with the sde, while both sdt and lrg are competitive; nevertheless, in lrg we arbitrarily assumed an extra 100,000 non-relevant documents scoring at zero, while sdt employs no arbitrary choices.

6 Conclusions

First, we are not satisfied with the quality of our ranking, although it seems to be around this of a median system. We were targeting on using only the feedback model in training and discard the initial narrative query, but we misinterpreted Lemur's parameters. The ranking quality affects the quality of estimates in two out of our three methods, namely, the sdt and sde, with the latter being affected the most. However, we remind the reader that the analysis of results in this paper was based on comparisons to a hypothetical (non-existent) median system. According to the official preliminary analysis (Legal Track's overview paper in TREC Notebook), our ranking was the 3rd best in recall at 30%, and the 2nd best in Area Under the receiver operating characteristic Curve (AUC). AUC is a measure of the quality of the ranking, independent of the submitted probability estimates.

Second, sde and lrg must be greatly affected by biases in the training set; sdt does not use training data. Both former methods assume training data uniformly sampled from the collection, which is clearly not the case here. Our heuristic to remove some of the bias in lrg by assuming 100,000 non-relevant documents scoring at zero does seem to have helped however. Nevertheless, this was an arbitrary number.

Third, the KDE methods used in sde was found to be very sensitive to the choice of bandwidth, influencing greatly the resulting probability estimates. We will report on these and other issues more extensively in further work.

References

[1] Avi Arampatzis and Jaap Kamps. Where to stop reading a ranked list? In *Proceedings TREC 2008*. NIST, 2008.

[2] Avi Arampatzis, Jaap Kamps, and Stephen Robertson. Where to stop reading a ranked list? Threshold optimization using truncated score distributions. In *Proceedings SIGIR'09*, pages 524–531. ACM Press, 2009.

[3] Avi Arampatzis and Stephen Robertson. Modeling score distributions in information retrieval. *Information Retrieval*, 14:26–46, 2011.

[4] Avi Arampatzis and André van Hameren. The score-distributional threshold optimization for adaptive binary classification tasks. In *Proceedings SIGIR'01*, pages 285–293. ACM Press, 2001.

[5] D. R. Cox. *The Analysis of Binary Data*. Chapman & Hall, London, 1970.

[6] Norbert Fuhr, Ulrich Pfeifer, Christoph Bremkamp, Michael Pollmann, and Chris Buckley. Probabilistic learning approaches for indexing and retrieval with the trec-2 collection. In *Proceedings TREC 1993*. NIST, 1993.

[7] R. Manmatha, Toni M. Rath, and Fangfang Feng. Modeling score distributions for combining the outputs of search engines. In *Proceedings SIGIR'01*, pages 267–275. ACM Press, 2001.

[8] Brian D. Ripley and N. L. Hjort. *Pattern Recognition and Neural Networks*. Cambridge University Press, New York, NY, USA, 1995.

[9] Larry Wasserman. *All of Statistics: A Concise Course in Statistical Inference (Springer Texts in Statistics)*. Springer, September 2004.