Combining Structural Information and the Use of Priors in Mixed Named-Page and Homepage Finding

Paul Ogilvie and Jamie Callan
Language Technologies Institute
School of Computer Science
Carnegie Mellon University
{pto, callan}@cs.cmu.edu

Abstract

This paper presents Carnegie Mellon University's experiments on the mixed named-page and homepage finding task of the TREC 12 Web Track. Our results were strong; we achieved the success using language models estimated from combining information from document text, in-link text, and information present in the structure of the documents. We also present experiments using expectations about posterior distributions to create class-based prior probabilities. We find that priors do provide a large gain for our official runs, but we do further experiments that show the priors do not always help. Some preliminary analysis shows that the prior probabilities are not providing the desired posterior distributions. In cases where applying the priors harm performance, the observed posterior distributions in the rankings are far off of the desired posterior distributions.

1. Introduction

Documents found on the Internet are rich in structure, and this provides many information sources useful for retrieval. In particular, structural information has been found useful for known-item searches such as homepage finding and named-page finding [Craswell 2001][Kraaij 2002][Ogilvie 2003][Zhang 2002]. A known-item retrieval system should attempt to leverage the structural information in a manner that is consistent with its model and provides an improvement in retrieval performance.

In this paper, we present experiments where we combine structural information using language modeling. We create several document representations for each document using the structural information present in HTML documents. From these document representations, we form language models. We combine the language models using a linear interpolation to form a new language model. This new language model is then used to estimate the probability that the document has generated the query.

It is also possible to leverage query independent information through the use of document priors. Document priors are probabilities or beliefs that the document is relevant to the query independent of any knowledge about the query. In previous homepage finding experiments, [Kraaij 2002] found a prior based on the type of the URL to be a very effective source of information. The URL types ROOT, SUBROOT, FILE, and PATH form four distinct document classes. We hypothesize that these classes will also be useful for a mixed homepage/named-page finding task, and present experiments using these priors. The prior probabilities are estimated from training data which gives us a desired posterior distribution. This desired posterior distribution defines ratios of the document classes what we would like to observe in the rankings. Using Bayes' rule, we can estimate the prior probabilities from training data and corpus statistics. A detailed derivation is provided Section 5.

In the next section, we describe the basic generative language model where we are using information from only one document representation. Section 3 describes combining information from several language models. Section 4 briefly discusses system specifics for the experiments. Section 5 provides a detailed derivation of the priors and posterior distribution we used for the mixed homepage/named-page finding track of the TREC 12 Web Track, and Section 6 describes our official runs and other experiments. Section 7 provides a discussion of the URL priors and their effectiveness in producing the desired posterior distributions over the rankings. We state conclusions in Section 8.

2. Generative Language Models

A unigram language model defines a multinomial probability distribution over all words in the vocabulary of the corpus. These probabilities are interpreted as word generation probabilities, and documents are ranked by their probability of generating the query. This generation probability is computed by taking the product over all query terms of the probability of the query term given the language model [Zhai 2001]:

$$P(Q|\theta_D) = \prod_{i=1}^{|Q|} P(q_i|\theta_D) \quad (1)$$

where q_i is the i^{th} query term of query Q, $|Q|$ is the length of Q, and θ_D is a language model estimated from document D. In the case where we desire a ranking using only one of the document representations, we directly take

$$P(w|\theta_D) = P(w|\theta_{D(i)}) \quad (2)$$

where i indicates a specific document representation.

The language models for an individual document representation can be estimated by smoothing a maximum likelihood estimator (MLE) with a collection-wide document representation model:

$$P(w|\theta_{D(i)}) = \lambda_i P_{MLE}(w|D_i) + (1-\lambda_i) P_{MLE}(w|C_i) \quad (3)$$

where $C(i)$ is the aggregate over all document representations $D(i)$. The MLE for a document representation is:

$$P_{MLE}(w|D_i) = \frac{count(w; D_i)}{|D_i|} \quad (4)$$

The MLE distribution for the $C(i)$ is estimated similarly. It is common to set the linear interpolation parameters λ_1 and λ_2 in Equation 3 using guidance from Dirichlet prior smoothing:

$$\lambda_i = \frac{|D_i|}{|D_i| + \mu_i} \quad (5)$$

where μ_i is a parameter often set emprically or by using cross-validation [Zhai 2002].

3. Combining Information Using Language Models

When we wish to combine information formed from a variety of document representations, we take a linear interpolation of the unigram language models estimated from the individual representations [Ogilvie 2003]. This new language model for a document should be designed so that it closely models what we would expect a user to write as a query when requesting the document. This is different from doing a linear combination of scores from different systems as we directly estimate the probability of a word given the differing language models. This by replacing Equation 2 with:

$$P(w|\theta_D) = \sum_{i=1}^{k} \varphi_i P(w|\theta_{D(i)}) \quad (6)$$

where k is the number of document representations for a document and φ_i is the weight placed on the i^{th} document representation. The φ values must be positive and sum to one.

When we wish to incorporate a prior probability in the ranking, we restate the problem as one of estimating the probability of the document given the query and applying Bayes' rule:

$$P(D|Q) = \frac{P(Q|D)P(D)}{P(Q)} \quad (7)$$

The $P(Q)$ constant can be ignored in ranking, and the $P(\theta_D)$ component is the prior. We estimate $P(Q|D)$ using the generative probability and estimate the priors using the URL class of the web page, giving:

$$P(D|Q) \propto P(Q|\theta_D) P(NP \vee HP | type(D)) \quad (8)$$

where $P(NP \text{ or } HP|type(D))$ is the prior probability of the page being a named page or homepage given the URL type of the document and $P(Q|\theta_D)$ is the generative probability defined in Equation 1.

4. System Specifics

We use the Lemur toolkit [Lemur] for document indexing and retrieval. For document tokenization we used Inquery's stopword list and the Porter stemmer. The URLs were tokenized on punctuation (., /) and were not stemmed. A shorter stopword list was used for URLs ("http", "www", "com", "gov", "html", etc.). Each document had as many as seven document representations, as shown in Table 1. For every representation except the URL, we formed language models using Dirichlet prior smoothing. The Dirichlet prior parameter was chosen to be close to twice the average length of the representation. This is not an optimal parameter setting, but may not have a large

effect on results. See [Zhai 2001][Zhai 2002] for more information. The probability of a word given the document's URL was computed treating the URL and word as a character sequence, then computing a character-based trigram generative probability. The numerator and denominator probabilities in the trigram expansion were estimated using a linear interpolation with the collection model (all URLs in the corpus). The final document scores were computed as the generative probability of the query given the document, taking the linear interpolation over the document representations.

Representation	Description
Alt	Image alternate text
Font	Changed font sizes and headings
Full	Full document text
Link	In-link text
Meta	Meta tags (keyword, description)
Title	Document title
URL	Character trigram on URL

Table 1: Document representations

5. Parameter Estimation

The weights for the document representations (the φ parameters used in Equation 2) were estimated by averaging the MRR (Mean Reciprocal Rank) scores of the individual document representations on the TREC 10 Homepage Finding task and the TREC 11 Named-Page Finding task. Table 2 shows the MRR of the individual representations on the previous known-item tasks and the resulting scaled weights. Table 2 also shows the φ we estimated from the TREC 10 and TREC 11 data. We estimated the φ values by scaling the TREC 10 and TREC 11 columns of Table 2 to sum to one then averaged the two scaled columns.

Representation	TREC 10 Homepage (MRR)	TREC 11 Named-Page (MRR)	φ
Alt	0.186	0.194	0.102
Font	0.155	0.191	0.093
Full	0.300	0.469	0.204
Link	0.515	0.455	0.263
Meta	0.115	0.144	0.069
Title	0.332	0.406	0.198
URL	0.132	0.131	0.071

Table 2: Estimating representation weight from TREC 10 and TREC 11 data

Table 3 shows the actual performance of the document representations on the TREC 12 test data. We can see that the performance of the document representations for named page finding in TREC 11 is similar to their performance in TREC 12. This is not surprising; the TREC 11 and TREC 12 named-page topics were selected in a similar manner and both use the .GOV corpus. However, the performance of the individual document representations TREC 10 homepage finding task is not as predictive for the TREC 12 homepage topics. In particular, the full document text is much less useful for the TREC 12 topics than for the TREC 10 topics. This could be a result of variance or small sample sizes, but we believe it is more likely a result of different corpus characteristics.

Representation	TREC 12 Homepage (MRR)	TREC 12 Named-Page (MRR)	TREC 12 Mixed (MRR)
Alt	0.167	0.171	0.169
Font	0.107	0.233	0.170
Full	0.125	0.394	0.260
Link	0.487	0.467	0.477
Meta	0.160	0.083	0.121
Title	0.284	0.416	0.350
URL	0.079	0.122	0.100

Table 3: Performance of individual representations on TREC 12 and hypothetical φ values estimated from test data

The priors were estimated from TREC 10 data and TREC 11 data. We made the assumption that the URLs of homepages in the .GOV corpus would have similar characteristics to those in the WT10G corpus. In addition to

using the test topics for the TREC 10 Homepage Finding task, we also used the 80 training topics provided that year. We leveraged the knowledge that there would be an equal number of homepage topics and named-page topics in the test set by scaling our estimates on the posterior P(*type*|NP or HP) to the same number of topics.

There are simpler methods to devise equivalent numbers for ranking purposes, but in the interests of describing what we did with the numbers we used, we present our actual numbers and computations. These numbers are presented in Table 3, and we provide a justification for the method used in Table 3 below. In these derivations NP denotes a named page, HP denotes a homepage, and *t* is a page type (ROOT, SUBROOT, FILE, or PATH). The method of derivation for priors is similar to the approach used in [Kraaij 2002].

Posterior:

$$\mathrm{P}(t|\mathrm{NP} \vee \mathrm{HP}) = \mathrm{P}(t|\mathrm{NP})\mathrm{P}(\mathrm{NP}) + \mathrm{P}(t|\mathrm{HP})\mathrm{P}(\mathrm{HP}) \tag{9}$$

since NP and HP are disjoint.

$$\approx \frac{c(t,\mathrm{NP})}{2c(\mathrm{NP})} + \frac{c(t,\mathrm{HP})}{2c(\mathrm{HP})} \tag{10}$$

where c(*t*,NP) denotes the count of named pages of type *t* in the training data and c(NP) denotes the number of named page queries in the training data. In this step we approximated the values with training data. Leveraging the fact that we know that homepages and named-pages are equally likely in the test data, we assumed P(NP) = P(HP) = 0.5. What we're substituting in for P(HP) and P(NP) is actually P(correct document is a HP) and P(correct document is a NP). This is fine here, but will have some implications to the correctness of their use for the priors. With a little rewriting, we get our formula for the estimation of the posterior distribution:

$$= \frac{c(t,\mathrm{NP}) + \frac{c(t,\mathrm{HP})c(\mathrm{NP})}{c(\mathrm{HP})}}{2c(\mathrm{NP})} \tag{11}$$

Prior:

$$\mathrm{P}(\mathrm{NP} \vee \mathrm{HP}|t) = \frac{\mathrm{P}(t|\mathrm{NP} \vee \mathrm{HP})\mathrm{P}(\mathrm{NP} \vee \mathrm{HP})}{\mathrm{P}(t)} \tag{12}$$

by Bayes rule. Note that now we are interpreting P(NP or HP|*t*) as P(this document is a NP or HP|*t*) and not P(correct document is a NP or HP|*t*) as we do not know that the document is correct. We believe the document may be correct, but we do not know. This means that substituting in the P(*t*|NP or HP) we estimated above is not the value we should be using here, but we will assume that it is close to the true value we desire. Doing so gives:

$$\approx \frac{\frac{c(t,\mathrm{NP}) + \frac{c(t,\mathrm{HP})c(\mathrm{NP})}{c(\mathrm{HP})}}{2c(\mathrm{NP})}\mathrm{P}(\mathrm{NP} \vee \mathrm{HP})}{\mathrm{P}(t)} \tag{13}$$

We estimate P(*t*) from the training data and discard the constants P(NP or HP) and 2*c*(NP) as our prior is multiplicative and discarding a multiplicative constant will not affect the rankings:

$$\propto \frac{c(t,\mathrm{NP}) + \frac{c(t,\mathrm{HP})c(\mathrm{NP})}{c(\mathrm{HP})}}{c(t)/|collection|} \tag{14}$$

where *c*(*t*) is the number of documents of type *t* in the collection and |*collection*| is the number of documents in the collection. We can also ignore the constant size of the collection:

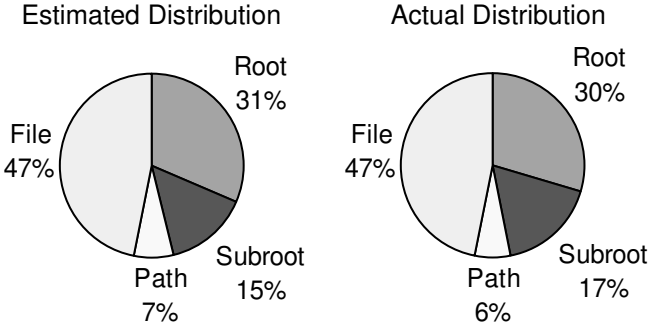
$$\propto \frac{c(t,\mathrm{NP}) + \frac{c(t,\mathrm{HP})c(\mathrm{NP})}{c(\mathrm{HP})}}{c(t)} \tag{15}$$

This gives us the formula we used to estimate the priors.

Class	NP	HP	HP at 170	NP + HP at 170	Posterior	.GOV	Prior
ROOT	5	216	102.0	107.0	0.315	6768	0.0158
SUBROOT	14	75	35.4	49.4	0.145	120923	0.000409
PATH	10	29	13.7	23.7	0.070	65980	0.000359
FILE	141	40	18.9	159.9	0.470	1054082	0.000152
Total	170	360	170.0	340.0	1	1247753	
Computation	From data	From data	HP * 170/360	NP + HP at 170	(NP + HP at 170) / 340	From data	(NP + HP at 170) / .GOV

Table 4: Estimation of prior probabilities and posterior distribution expectations

We note that this method of parameter estimation for the posterior distribution expectation was very accurate given the training data. Figures 1 and 2 show pie charts corresponding to the distribution we estimated and the actual distribution observed in the relevance judgments. We believe it is reasonable to achieve such good estimation of this distribution in practice, as it is a relatively low cost activity to provide assessments for known-item tasks.



Figures 1 and 2: Estimated posterior distribution of page types for correct documents (left) and actual distribution of correct documents (right).

6. Experiments

In this section we describe official and unofficial runs. We submitted four official runs LmrEq, LmrEst, LmrEqUrl, and LmrEstUrl. "Lmr" denotes the Lemur system, "Eq" indicates the φ values were set to be equal to each other, "Est" indicates that the φ were those presented in Table 2, and "Url" signifies that the URL priors presented in Table 4 were applied to the scores. Table 5 summarizes these runs and their performance. LmrFlat and LmrFlatUrl are unofficial runs that do not use document structure or the text of the URL.

Run	Official	Structure	φ	URL Prior	Not found	Found by 10	MRR
LmrEq	YES	YES	equal	NO	8.3 %	83.3 %	0.652
LmrEst	YES	YES	estimated	NO	7.7 %	83.3 %	0.640
LmrFlat	NO	NO	-	NO	11.0 %	78.7 %	0.612
LmrEqUrl	YES	YES	equal	YES	5.3 %	88.0 %	0.713
LmrEstUrl	YES	YES	estimated	YES	4.7 %	89.3 %	0.727
LmrFlatUrl	NO	NO	-	YES	9.3 %	50.7 %	0.315

Table 5: Summary of runs and their performance

Our best performance was achieved when we used the estimated φ in combination with the URL priors. However, when we did not use the URL priors, the estimated φ parameters had worse performance than equally weighted φ values. This raises questions about both the use of URL priors and the method of training the φ values. We recognize that our approach to training the φ values is not optimal or always effective. What we found more interesting was that the URL priors did not help the different runs uniformly, despite the fact that they had similar initial performance. Applying the URL priors to the LmrFlat run severely degraded performance, so this leads us to have questions about the use of URL priors.

7. Discussion of URL Priors

When we apply the priors to the scores returned in a ranking, we do so with the hopes that the reranked lists will match our expectations of the posterior distributions. The fact that the URL priors do help in some cases suggests that we may indeed be observing this behavior when the priors are helping.

To test this hypothesis, we plotted the cumulative distribution of results across all queries. We estimated the probability of a class given a run by counting all documents of a given class up to a rank threshold across all queries and dividing by the number of documents returned by that rank. Figures 4-6 (last page of the paper) show these estimated distributions. The lines without the points are the desired posterior distribution we hoped to achieve by applying the prior probabilities. For LmrEqUrl and LmrEstUrl, we see that the FILE, PATH, and SUBROOT classes are favored more than desired, but the ROOT class is returned less often than we would like. However, we know that performance is increased by applying the priors to the LmrEq and LmrEst runs, and we can see in Figures 4 and 5 that the URL prior brought the results closer to the desired distribution of documents in the ranking.

On the other hand, LmrFlatUrl does not match our expected posterior distribution well (Figure 6). Applying the URL prior to LmrFlat produces very undesirable behavior in the rankings. It is not surprising that the mean reciprocal rank of LmrFlatUrl is much worse than that of LmrFlat. The cause of this behavior is not apparent from the current analysis, but the analysis does suggest a way to assess whether the priors producing a desirable effect on the posterior distributions. This can be done without relevance assessments, so it may be useful during the training and tuning phase of a retrieval experiment.

8. Conclusions

This paper described our TREC mixed homepage/named-page runs. We feel that our performance on this task was very strong. Our basic approach was to use language models estimated using structural information present in the documents to estimate the query generation probability. We found a URL-based prior that others found successful for the TREC 10 homepage finding task was also effective here. We described how we estimated the priors, and provided data analysis as to the effectiveness of the priors.

However, we also demonstrate that the prior probabilities are not producing the desired expected posterior distributions. We provide some analysis suggesting that when the priors harm performance, they produce undesirable effects to the posterior distribution. This analysis is simple to do and may be useful for others when tuning their systems.

For future work, we would like to gain a better understanding of the reason why the priors are not producing desired posterior distributions. One hypothesis is that the distribution of documents in the ranking does not match the distribution of documents in the collection. This may cause any biases present in the original ranking to be present in the reranked results lists. Another cause may be that the scores behave differently for the different classes. In this case, a simple flat prior may not fix the problem. We feel that if we can come up with a solution that produces the desired posterior distribution while preserving as much information in the scores as possible, we may be able to improve on our already strong results.

9. Acknowledgements

This work was sponsored in full by the Advanced Research and Development Activity in Information Technology (ARDA) under its Statistical Language Modeling for Information Retrieval Research Program. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors, and do not necessarily reflect those of the sponsor.

10. References

[Lemur] The Lemur Toolkit. http://www.cs.cmu.edu/~lemur

[Craswell 2001] N. Craswell, D. Hawking, and S. Robertson. Effective site finding using link anchor information. In *Proceedings of the Twenty-Fourth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'01)*, 2001.

[Kraaij 2002] W. Kraaij, T. Westerveld, and D. Hiemstra. The importance of prior probabilities for entry page search. In *Proceedings of the Twenty-Fifth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'02)*, 2002.

[Ogilvie 2003] P. Ogilvie and J. Callan. Combining document representations for known-item search. In *Proceedings of the Twenty-Sixth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'03),* 2003.

[Zhai 2001] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the Twenty-Fourth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'01)*, 2001.

[Zhang 2003] M. Zhang, R. Song, C. Lin, S. Ma, Z. Jiang, Y. Jin, Y. Liu, and L. Zhao. THU TREC2002 Web Track Experiments. In *Proceedings of the Eleventh Text Retrieval Conference, TREC 2002*, 2003.

[Zhai 2002] C. Zhai and J. Lafferty. Two-stage language models for information retrieval. In *Proceedings of the Twenty-Fifth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'02)*, 2002.

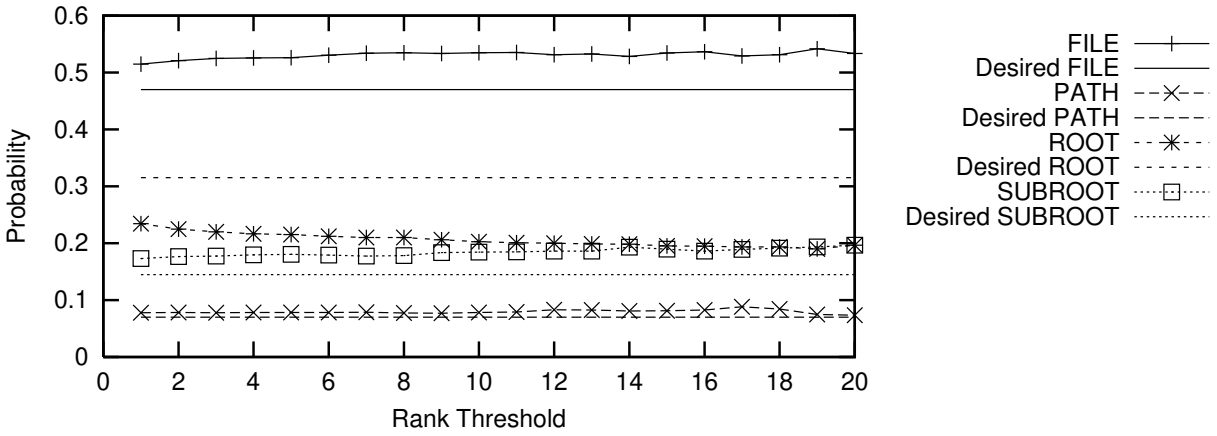


Figure 4: Posterior distribution for **LmrEqUrl**. Applying the URL prior to LmrEq gives result lists are more towards biased towards the FILE and SUBROOT classes than desired, and less biased toward the ROOT class then desired.

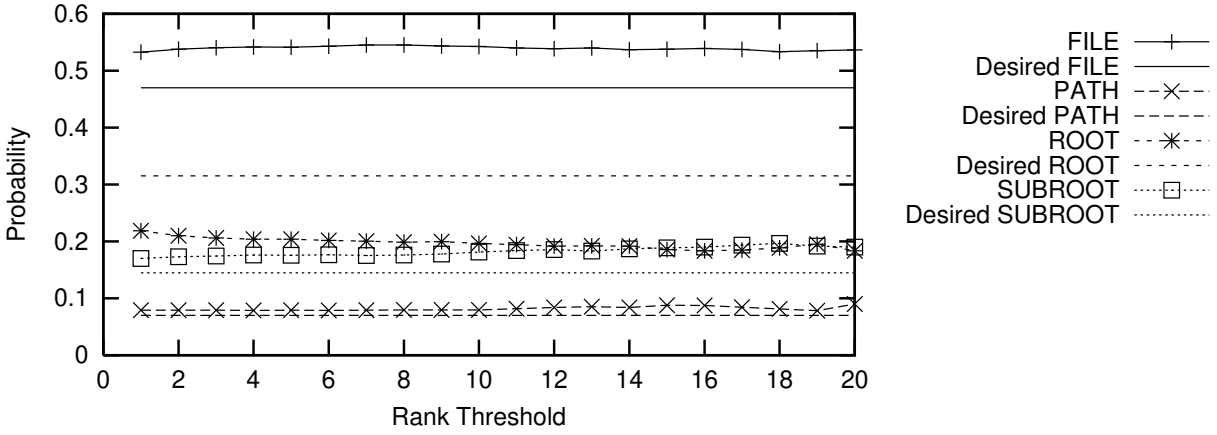


Figure 5: Posterior distribution for **LmrEstUrl**. The trends here are similar to those in Figure 4 for LmrEqUrl.

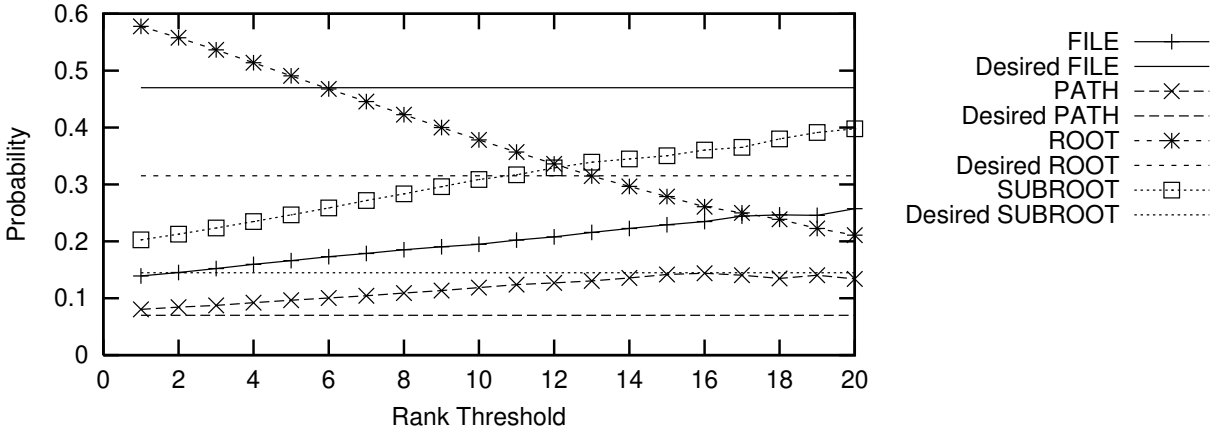


Figure 6: Posterior distribution for **LmrFlatUrl**. Applying the URL priors to LmrFlat results in a heavy bias towards ROOT pages at early ranks which decays rapidly. There is a trend towards an increasing bias toward the SUBROOT class. The FILE class has a much stronger bias against it then desired, and the PATH class has a stronger bias towards it then desired. The bias against the FILE class may account for the poor performance.