Learning Models for Ranking Aggregates

Craig Macdonald and Iadh Ounis

School of Computing Science, University of Glasgow,
Glasgow G12 8QQ, UK
{craig.macdonald,iadh.ounis}@glasgow.ac.uk

Abstract. Aggregate ranking tasks are those where documents are not the final ranking outcome, but instead an intermediary component. For instance, in expert search, a ranking of candidate persons with relevant expertise to a query is generated after consideration of a document ranking. Many models exist for aggregate ranking tasks, however obtaining an effective and robust setting for different aggregate ranking tasks is difficult to achieve. In this work, we propose a novel learned approach to aggregate ranking, which combines different document ranking features as well as aggregate ranking approaches. We experiment with our proposed approach using two TREC test collections for expert and blog search. Our experimental results attest the effectiveness and robustness of a learned model for aggregate ranking across different settings.

1 Introduction

Identifying expert persons in an organisation, key bloggers for a topic on the blogosphere and finding related entities on the Web are all examples of aggregate ranking tasks, where objects - e.g. people - are represented by sets of documents that must be ranked in response to a query. Various models have been proposed for aggregate ranking [1–3]. However, while each model might perform well in a particular setting, it might not adapt well to another aggregate ranking task. For instance, the Model 2 approach [1] performs well for expert search, but less so for identifying key bloggers [4]. In this work, we investigate how to learn effective and robust aggregate ranking models using the learning to rank paradigm [5].

In learning to rank, features are normally defined on the objects being evaluated, i.e. document features which are then evaluated directly. However, in aggregate ranking tasks, the usefulness of document features is difficult to assess in a learning to rank framework, as relevance assessments are only defined on the aggregates. We propose a novel methodology for applying learning to rank to the ranking of document aggregates. In this methodology, features are defined in terms of three independent variables, namely the document ranking, the rank at which the document ranking is truncated, and the aggregate ranking strategy used to convert the document ranking into a ranking of aggregates. We evaluate the proposed methodology using standard TREC test collections for two aggregate ranking tasks, namely expert and blog search. Our experiments analyse the impact of each of the independent variables on the effectiveness of the learned models. The results show that effective and robust learned models for aggregate ranking can be obtained using our proposed methodology.

To the best of our knowledge, our work is the first framework showing how to apply learning to rank techniques to aggregate ranking. In particular, we conduct an in-depth study into learning models for two aggregate ranking task. The remainder of this paper is structured as follows: Section 2 reviews existing approaches for ranking aggregates, and discusses their limitations, before introducing the learning to rank paradigm for information retrieval; Section 3 defines our methodology for learning to rank in aggregate ranking tasks; Section 4 details our research questions and experimental setup, while the experimental results and analysis follow in Section 5; Finally, Section 6 compares our work to other recent learned approaches for aggregate ranking, and Section 7 provides concluding remarks.

2 Aggregate Ranking towards Learning to Rank

2.1 Aggregate Ranking

There are several well-known approaches for aggregate ranking spawned by research in the expert and blog search tasks. Typically, all approaches use profiles of documents to represent each *candidate object*. However, they differ in the way in which the profiles are ranked in response to a query. For instance, Balog et al. [1] proposed the Model 1 and Model 2 language modelling approaches for expert search. Similarly, Elsas et al. [3] proposed the "large document" and "small document" language models in the context of blog search, which correspond to Models 1 & 2, respectively. Macdonald & Ounis took a different approach to expert ranking - in their Voting Model [2], a document ranking provides votes to associated objects (e.g. experts) to be retrieved for the query. Twelve different voting techniques were proposed that define different ways in which the votes can be aggregated. In particular, the CombSUM voting technique is similar to Model 2, but agnostic to the particular document ranking technique applied. Different voting techniques have since been shown to be effective at finding key bloggers [6] and related entities from the Web [7].

While these various aggregate ranking approaches may be suitable for various tasks or datasets, an effective setting for one aggregate ranking task may not perform well on another task. For instance, while Balog et al. found Model 2 to be more effective than Model 1 for expert search [1], the opposite was found to be true for finding key bloggers [4]. In contrast, Elsas et al. [3] found the small document model (which corresponds to Model 2 of Balog et al.) to be more effective than the large document model (c.f. Model 1) for key blog identification. By proposing various different voting techniques, Macdonald & Ounis [2] acknowledged that different voting techniques may be suitable for different tasks or settings. For example, while the CombMAX voting technique can be effective for some expert search tasks [8], it was found to be less effective for finding key blogs [6]. Moreover, the voting techniques can consider any number of top-ranked documents from the underlying document ranking. However, the most effective rank cutoff can vary between tasks and collections [8].

Due to these difficulties in finding consistently effective and robust settings for aggregate ranking approaches, in this work, we propose instead to learn an aggregate ranking model, which is robust and effective across different tasks and settings. In particular, the voting techniques of the Voting Model provide various different aggregate ranking strategies that are effective for different tasks or collections. We hypothesise that it is possible to learn an appropriate and effective combination of voting technique strategies using learning to rank techniques. In the remainder of this section, we provide background on learning to rank techniques, and the challenges incurred in their application to aggregate ranking. This is followed in Section 3 by our proposed methodology for learning an effective aggregate ranking model.

2.2 Learning to Rank

Learning to rank describes the application of machine learning techniques to select weights for different document features in an information retrieval (IR) system [5]. For instance, learning to rank techniques are often applied by Web search engines, to combine various document weighting models and other query-independent features [9]. The various learning to rank approaches in the literature fall into one of three categories, namely pointwise (learn relevance independently of other documents), pairwise (optimise the number of pairs of documents correctly ranked) and listwise (optimise an information retrieval evaluation measure that considers the entire ranking list). In this work, we consider two listwise approaches that directly evaluate with respect to the target IR evaluation measure, instead of the evaluation approximations that are used by pointwise or pairwise approaches. Moreover, listwise techniques have been shown to learn more effective models [5]. To examine the impact of different learning to rank techniques on the effectiveness of the learned models, we deploy two listwise techniques, namely Metzler's Automatic Feature Selection algorithm (AFS) [10], and AdaRank [11]. Both AFS and AdaRank take a greedy approach to feature selection, by iteratively selecting the feature that most improves retrieval performance in combination with the previously selected features. Features that are not beneficial to the retrieval performance on the training set will not be selected. However, while AFS finds the optimal weight for each feature, AdaRank calculates feature weights based on their boosted performance on the training queries [11]. In practice, this makes AdaRank considerably faster than AFS. The general steps to learn a ranking *model* are as follows [5]:
1. Generate a sample of training documents using an initial retrieval approach.
2. Extract all features for all of the documents in the sample. A feature is a numerical indicator thought to be of use in a learned model.
3. Learn a ranking model through the application of a learning to rank approach.
4. Apply the learned model on a sample of test documents with the same features.

It is of note that the strategy used to create the sample of documents to re-rank impacts on the effectiveness of the learned model. In [5], Liu states that for the LETOR learning to rank datasets, the top 1000 documents are sampled using BM25 on the content alone. However, Liu notes that while this method of producing the sample is sufficient, it may not be the best [5]. Indeed, if the sample has insufficient recall, then the scope for the learning to rank approach to generate a quality ranking will be hindered. In Section 4, we describe how an appropriate sample of aggregate objects is created for our learning approach.

A salient point of learning to rank is that the features are defined on the objects being evaluated, i.e. document features which are then evaluated directly. However, in aggregate ranking tasks, the usefulness of document features is difficult to assess in a learning to rank framework, since such features are not directly defined on the aggregates, and relevance assessments are only defined on

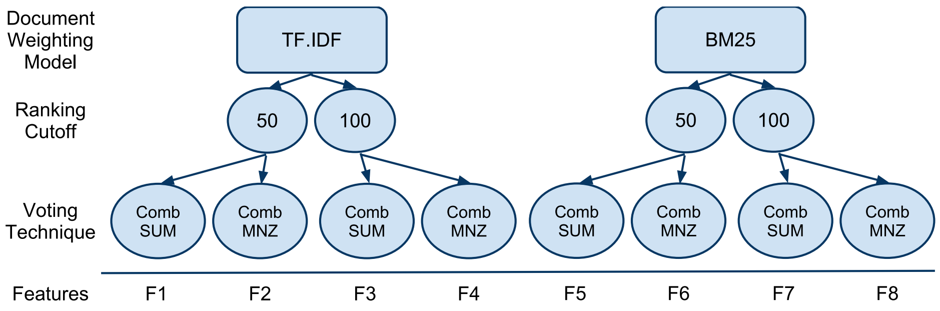

Fig. 1. Eight features obtained from two document ranking models (TF.IDF & BM25), two rank cutoffs (50 and 100) and two voting techniques (CombSUM & CombMNZ).

the aggregates. To tackle this problem, in the next section, we propose a novel methodology for applying learning to rank for ranking document aggregates.

3 Learning to Rank Aggregates

In aggregate search tasks, the goal is to rank objects such as people, entities or blogs, where each aggregate object is defined as a set of documents. However, as mentioned above, a key complication of learning to rank aggregates is that many features (e.g. uni- and bi-gram weighting models, PageRank, to name a few) are defined at the level of documents, rather than at the level of aggregate objects. However, when using learning to rank, the features must be defined at the same level as the relevance assessments, i.e. at the aggregate level.

To take such document features into account, it could be intuitive to use a learned document ranking as input to an aggregate ranking strategy. However, two factors combine to make such an approach not viable. Firstly, in aggregate ranking tasks, the relevance assessments are defined at the aggregate level, therefore there is no easy way to learn an effective ranking. Secondly, even when there are document level relevance assessments for the same queries as the aggregate ranking task (e.g. TREC 2007 and 2008 Enterprise tracks, for the document search and expert search tasks [12, 13]), it has been shown that increasing the quality of the document ranking does not always result in increasing the effectiveness of the resulting ranking of candidate experts [14]. Moreover, Macdonald & Ounis [15] showed that when perfect document rankings (e.g. MAP 1.0) are applied, the resulting candidate rankings were further degraded. Such counter-intuitive results can be explained in that the document relevance assessments measure different properties of the document ranking than those desirable for an effective ranking of candidates [14], and suggest that the direct learning of document ranking features for use to rank aggregates is not a viable strategy.

Instead of trying to directly learn a document ranking, we propose to work with features directly defined at the level of the aggregate objects. Firstly, a sample of aggregate objects to be re-ranked is defined, using a single effective aggregate ranking strategy (c.f. BM25 used by LETOR [5]). Then, we propose that each possible aggregate ranking strategy is a feature, defined on the objects in the sample. In particular, in this work, as mentioned in Section 2.1, we apply different instantiations of the Voting Model. Indeed, the Voting Model defines many aggregate ranking strategies as voting techniques, each of which can use various document rankings with different rank cutoffs. Hence, each different voting technique that is applied to the same document ranking represents a different feature. Similarly, if the document ranking is changed, or truncated at a different rank cutoff, then a new feature is defined. Figure 1 shows an example feature set, where two document ranking models (TF.IDF and BM25), truncated at two rank cutoffs (50 and 100) are combined with two voting techniques from [2] (CombSUM and CombMNZ), to make a total of eight features. From Figure 1, it is clear that a single feature represents a path through the different levels of the tree, where the document ranking model, the cutoff and the voting technique are three independent variables of the feature. More formally, a single feature f for an object O for query Q is defined as:

$$f(O, Q, DM, \theta, VT) = VT(O, Tr(DM(Q), \theta)) \tag{1}$$

where $VT(O, Tr(DM(Q), \theta))$ is the score for object O according to a particular voting technique, operating on a ranking of documents returned by a ranking model DM for query Q. $Tr(DM(Q), \theta)$ truncates document ranking $DM(Q)$ to the top θ ranked documents. In this formulation of aggregate learning, there are three levels. Each level corresponds to one of the independent variables in Equation (1), namely DM, θ or VT, and the particular values for each variable define the exact feature generated. For example, in Figure 1, eight features are generated by $DM = \{DPH, BM25\}$, $\theta = \{50, 100\}$, and two voting techniques $VT = \{\text{CombSUM}, \text{CombMNZ}\}$.

Once many features have been extracted for the sampled candidates, a learning to rank technique can be applied. The outcome of the learning to rank technique is a weighted linear combination of features, such that a new aggregate ranking strategy is created, consisting of an *ensemble* of various voting techniques using different document rankings and cutoffs. Moreover, as with learning to rank applied on documents, the success of a learned approach depends on the number and usefulness of the features. As will be shown in the next section, we vary the three independent variables, to generate a large number of features for learning to rank aggregates. In this regard, the Voting Model is particularly suitable, as it defines many voting techniques for the independent variable VT. Moreover, it is agnostic to the choice of the document ranking DM and the ranking cutoff θ. In essence, this allows a combinatorial approach to feature generation by varying the instantiations of each independent variable.

Finally, in this work, our experiments only use features based on query-dependent document ranking models. However, query-independent features could also be handled within our proposed methodology, by defining them on a sample of documents selected by a query-dependent document ranking model. e.g. using PageRank as a document ranking, but defined on documents ranked by BM25.

4 Experimental Setup

In the proposed methodology for learning aggregate models using many features, each feature is generated by different instantiations of the DM, θ and VT independent variables. In our experiments, we investigate the importance of each independent variable, by varying one while holding all other variables constant, to create *groups* of features. We then ascertain the importance of each of these feature groups, by addressing the following research questions:

Table 1. Tasks and test collections used.

	EX:07	EX:08	BD:07	BD:08
Corpus	CERC		Blogs06	
Number of Documents	370K		3M	
Number of Candidates	3.4K		100K	
Mean Profile Size	68.2		31.94	
Number of Topics	50	55	45	50

(1). Does using more than one voting technique benefit retrieval performance?
(2). Does using more than one document cutoff benefit retrieval performance?
(3). Does using more than one document ranking benefit retrieval performance?
(4). Finally, what are the most important features for each of the investigated retrieval tasks?

The remainder of this section defines the experimental setup to address these research questions. In particular, Section 4.1 details the selected test collections and the adopted training regime, while Section 4.2 details the generated features.

4.1 Tasks and Training

We address the above research questions using two aggregate ranking search tasks, namely expert search and blog distillation. In the expert search task, candidate experts within an enterprise organisation are aggregate objects that must be ranked in response to a query. In particular, the expertise of each candidate is represented by a profile of intranet documents containing their name or email address. We use the TREC Enterprise track 2007 and 2008 expert search task test collections [12, 13] - denoted EX:07 and EX:08, respectively. Both tasks are based on the CERC corpus of intranet documents. In the blog distillation task, key blog(ger)s that have a recurring interest in a query topic should be identified. In this task, each blog is an aggregate, consisting of all of the blog's postings. In particular, we use the TREC Blog track 2007 and 2008 blog distillation test collections [16, 17], denoted BD:07 and BD:08, respectively, both of which use the Blogs06 corpus. Statistics of the used test collections are presented in Table 1.

We deploy the two listwise learning to rank techniques described in Section 2.2, namely AFS and AdaRank. Moreover, applying a learning to rank technique requires enough training data to successfully learn the weights of the features, as well as sufficient test data to adequately evaluate the learned models. The selected test collections are the only aggregate ranking test collections currently available with more than 50 topics, sufficient for both training and testing. In contrast, the LETOR datasets for evaluating learning to rank approaches [5] do not address aggregate ranking tasks. While the TREC 2009 Blog track faceted blog distillation task and the TREC 2009 Entity track related entity finding task are also aggregate ranking tasks, they do not have enough topics (39 and 20, respectively) for successfully applying learning to rank techniques.

In our experiments, we apply an appropriate training regime whereby results are reported on different topics from the training topics, but within the same corpus. Hence, a clear separation between training and testing topics is enforced. For instance, we train on EX:07 topics and test on EX:08, and vice versa. Note

Table 2. Applied instantiations of each independent variable. A total of 540 aggregate level features are generated.

Variable	#	Description
DM	6	DPH document weighting model [18] on either content or anchor text, with and without query term proximity [19]. Collection enrichment using Wikipedia as per [20], applied on either content or anchor text.
θ	9	Ranking cutoffs: $\theta = \{50, 100, 250, 500, 1000, 2000, 3000, 4000, 5000\}$.
VT	10	CombSUM, CombMNZ, CombMAX, expCombSUM, expCombMNZ from the Voting Model, as per [2,6]. CombSUMNorm1D, CombMNZNorm1D, CombMAXNorm1D, expCombSUMNorm1D, expCombMNZNorm1D adds profile length normalisation [8].

that this training regime makes our results perfectly comparable to the participating systems of the EX:08 and BD:08 TREC tasks only. However, for the EX:07 and BD:07 topics, we are using training data that was not available to the TREC participants of that year. Mean Average Precision (MAP) is used as both the training measure during learning, and the measure reported in the results.

4.2 Feature Generation

For both the expert search and blog distillation tasks, we use various instances for the variables DM, θ and VT to generate different features. In general, to permit an impartial cross-comparison of selected features across expert search and blog distillation tasks as per research question (4), we adopt a uniform setting between both tasks, in that only techniques that are applicable to both tasks are applied. Table 2 details the instantiations of the DM, θ and VT independent variables. In particular, the DPH document weighting model [18] is applied, with and without proximity [19] or collection enrichment [20], on either document content or the corresponding anchor text of the incoming hyperlinks. Nine different document ranking cutoffs are applied, along with ten different voting techniques from the Voting Model [2]. Five of these voting techniques apply profile length normalisation, which often increases effectiveness by preventing aggregate objects with many associated documents from being over emphasised in the final ranking [8]. The product of all of the above possible instantiations of variables DM, θ and VT is a total of $6 \times 9 \times 10 = 540$ features that can be considered by the two learning to rank techniques used in our experiments.

Lastly, we consider the generation of the sample of objects to re-rank. As discussed in Section 2.2, for the LETOR datasets, Liu suggests that selecting the top 1000 ranked BM25 documents produces a sample of sufficient recall [5]. Similarly, in this work, our sample consists of the top 200 aggregate objects ranked by the CombSUM voting technique [2], using a DPH document ranking cutoff at rank 1000. In particular, DPH represents an effective parameter-free document weighting model [18], while CombSUM was shown to be effective for both expert search and blog distillation [2,8], and is similar to the Model 2 language modelling approach [1]. By using a sample with depth 200, we have a good recall from which to re-rank objects, as the testing evaluation is limited to rank 100 as per the TREC setting of the expert search and blog distillation tasks.

Table 3. MAP performances of various feature sets, trained using two learning to rank techniques. Significant differences from C 1000 CombSUM are denoted with $\gg$ ($p < 0.01$), $>$ ($0.01 \leq p \leq 0.05$) and $=$ ($p > 0.05$). $* * *$ denotes when all features from Table 2 are applied. The best learned model for each task is emphasised.

	DM	θ	VT	# Features	EX:07	EX:08	BD:07	BD:08
				AFS				
1	C	1000	CombSUM	1	0.2651	0.2532	0.2468	0.1909
2	C	1000	$*$	10	**0.4184**$\gg$	**0.4128**$\gg$	0.2870$\gg$	0.2493$\gg$
3	C	$*$	CombSUM	9	0.3978$\gg$	0.3180$>$	0.2514$=$	0.2067$=$
4	C	$*$	$*$	90	0.4134$\gg$	0.4043$\gg$	0.3148$\gg$	0.2422$\gg$
5	$*$	1000	CombSUM	6	0.2776$=$	0.2578$=$	0.2705$\gg$	0.2022$>$
6	$*$	1000	$*$	60	0.4153$\gg$	0.4103$\gg$	0.3264$\gg$	0.2547$\gg$
7	$*$	$*$	CombSUM	54	0.4076$\gg$	0.3133$>$	0.2653$=$	0.2170$\gg$
8	$*$	$*$	$*$	540	0.4107$\gg$	0.4041$\gg$	**0.3480**$\gg$	**0.2710**$\gg$
				AdaRank				
9	C	1000	CombSUM	1	0.2651	0.2532	0.2468	0.1909
10	C	1000	$*$	10	0.4141$\gg$	**0.4211**$\gg$	0.2925$\gg$	0.2338$\gg$
11	C	$*$	CombSUM	9	0.3802$\gg$	0.3670$\gg$	0.2439$=$	0.1893$=$
12	C	$*$	$*$	90	**0.4199**$\gg$	0.3857$\gg$	0.3189$\gg$	**0.2493**$\gg$
13	$*$	1000	CombSUM	6	0.2896$>$	0.2581$=$	0.2698$\gg$	0.2044$\gg$
14	$*$	1000	$*$	60	0.4010$\gg$	0.4035$\gg$	**0.3204**$\gg$	0.2327$\gg$
15	$*$	$*$	CombSUM	54	0.3823$\gg$	0.3688$\gg$	0.2700$>$	0.2075$\gg$
16	$*$	$*$	$*$	540	0.3973$\gg$	0.3791$\gg$	0.2817$=$	0.2284$\gg$

5 Experimental Results

In this section, we experiment to address each of the research questions described in Section 4 in turn. In particular, in Sections 5.1, 5.2, and 5.3, we investigate the impact of each specific feature group on the learning process, i.e. by varying VT, θ, and DM one at a time, while holding the other two variables constant. Each section analyses the results of Table 3. In this table, the independent variables DM, θ and VT are varied in turn - each can take a single instantiation, namely DPH on content (denoted C), 1000 and CombSUM, respectively, or all of the listed variable instantiations in Table 2 (denoted *). For example, C 1000 CombSUM (rows 1 & 9) denotes our baseline approach that created the sample of aggregate objects to re-rank (c.f. BM25 used by Liu for LETOR [5]). We test for significant differences from the baseline using the Wilcoxon Signed Rank test, denoted by $\gg$ ($p < 0.01$), $>$ ($0.01 \leq p \leq 0.05$) and $=$ ($p > 0.05$). Finally, Section 5.4 analyses the most important features for each task, while additional discussion follows in Section 5.5.

5.1 Feature Group: Voting Techniques

Firstly, we examine the impact of the research question (1), namely whether applying more than a single voting technique increases the effectiveness of the learned aggregate ranking model. To analyse the influence of adding additional voting techniques on the effectiveness of the learned model, we compare C 1000 CombSUM (rows 1 & 9) with C 1000 * (rows 2 & 10) in Table 3. We observe up to 70% relative improvements over the baseline. Indeed, these improvements are statistically significant for both learning to rank techniques, and for all topic sets. We conclude that building a ranking model with multiple voting techniques can massively benefit retrieval performance. Moreover, if we examine other settings, e.g. where multiple cutoffs or multiple ranking feature groups have already been applied, we see further improvements (see each setting in rows 3 vs 4, 5 vs 6, 7 vs 8, 11 vs 12, 13 vs 14, or 15 vs 16). Overall, these positive results show that applying multiple voting techniques and learning a suitable combination results in an effective model that robustly generalises to other topic sets on the same corpus.

5.2 Feature Group: Ranking Cutoffs

Next, we investigate research question (2), addressing whether adding more document ranking cutoffs increases retrieval effectiveness. Comparing C 1000 CombSUM (rows 1 & 9) with C * CombSUM (rows 3 & 11), we observe significant increases for the expert search task. However, on the blog distillation task, only AFS can identify models that improve over the baseline sample. This suggests that having multiple cutoffs are very useful for expert search, where there are highly on-topic documents retrieved early in the system ranking that bring valuable expertise evidence. On the other hand, for blog distillation, adding additional cutoffs brings less benefit, suggesting that for this task, the top ranked documents are, in general, less useful. We conclude that the multiple cutoffs feature group is useful for the expert search task only.

We also examine the impact of the multiple cutoffs feature group when the multiple document rankings or multiple voting techniques feature groups have already been applied. In these cases, we note that, in general, adding the multiple cutoffs feature group is beneficial for improving the effectiveness of the multiple document ranking feature group alone (rows 5 vs 7 and 13 vs 15). However when multiple voting techniques have been applied, multiple document ranking cutoffs have little or no positive impact on retrieval performance (rows 2 vs 4, 6 vs 8, 10 vs 12 and 14 vs 16). This suggests that the sources of evidence from multiple voting techniques and document ranking cutoffs are correlated. However, in a learning to rank setting, this is not a disadvantage, as the learning process will only select one of two similar features.

5.3 Feature Group: Document Rankings

We now examine the impact on effectiveness of using features based on multiple document rankings, as per the research question (3). Comparing with C 1000 CombSUM (rows 1 vs 5 and 9 vs 13), we note improvements for all settings. However, these are only statistically significant in 5 out of 8 settings. Overall, we conclude that while adding the multiple document rankings feature group does positively impact retrieval effectiveness, it does not have as large an effect as the multiple cutoffs or multiple voting techniques feature groups (e.g. rows 1 vs 5, compared to 1 vs 2, and 1 vs 3).

Nevertheless, the effectiveness of multiple document rankings can be improved by further adding the multiple cutoffs or multiple voting techniques feature groups. In fact, the best settings for the BD:07 and BD:08 tasks can be found when applying the 540 features of * * * (see row 8 for AFS - we compare AFS and

Table 4. Strongest four features in C * *, and the number of features selected by AFS (out of 90).

EX:07	EX:08	BD:07	BD:08
C 50 CombMNZ-Norm1D	C 2000 expCombMNZ-Norm1D	C 250 expCombMNZ-Norm1D	C 5000 CombSUM-Norm1D
C 50 expCombMNZ	C 100 expCombSUM	C 100 expCombMNZ-Norm1D	C 4000 expCombSUM
C 2000 CombSUM-Norm1D	C 100 expCombMNZ-Norm1D	C 100 CombSUM	C 1000 CombMAX
C 1000 expCombMNZ	C 4000 expCombSUM	C 1000 CombMAX	C 500 CombMAX
(12)	(19)	(19)	(14)

AdaRank performances in Section 5.5 below). We conclude that some aspects of the extra document rankings (collection enrichment, proximity or anchor text) do bring some useful additional evidence, particularly for blog distillation.

5.4 Task Analysis

We now address research question (4), by analysing the most important features identified when training for each task. Table 4 reports the top 4 features by weight as identified by AFS in the C * * feature set of 90 features, as well as the total number of features selected (out of 90). From this table[1], we observe that 12-19 features were typically chosen - this suggests the similarity of many of the features derived from the C document ranking (i.e. DPH). Of the first ranked chosen features, all apply voting techniques with profile length normalisation, suggesting that this is an important attribute of effective voting techniques. However, it is of note that the chosen voting techniques and cutoffs vary for different corpora and tasks. This attests the usefulness of our approach to learn effective models for different aggregate ranking corpora and tasks, since the various effective features for each task can be automatically selected and weighted.

5.5 Discussion

Having examined each of the feature groups in turn, we now analyse the combination of all feature groups. Looking across all feature groups, we note that once multiple voting techniques have been applied, applying multiple cutoffs or multiple document rankings has in general no marked benefit in retrieval performance across all search tasks. However, the multiple voting techniques feature group is robust across all tasks and learning to rank techniques (i.e. 27 significant increases out of 28 across all even numbered rows in Table 3). Indeed, for blog distillation, * * * learned by AFS exhibits the highest performance, suggesting that by allowing the learner to choose from all 540 features, an effective and robust model can be learned. It is also of note that we performed additional experiments using 5-fold cross validation across all $\sim$100 topics for each task (These results are omitted for reasons of brevity). While the obtained retrieval performances were naturally improved by more training, promisingly, all of the experimental conclusions were unchanged.

Comparing the learning to rank techniques, we note that higher quality results are generally found by AdaRank on expert search, however, on blog distillation, AFS is more successful. This contrasts with the results obtained on the

[1] For space reasons, we only report the best features from C * *. However, all of the conclusions are equally applicable to the most important features in * * *.

training topics, where AFS always identifies a model that is significantly better than that by AdaRank. We leave a study on the attributes of features sets that make each learning to rank technique amenable to different tasks to future work.

Finally, we compare our results to the TREC best runs for each task. In particular, for both EX:08 and BD:08, we note that our best results are comparable to the top ranked group. For the expert search task, the highest performing TREC run deployed models encompassing the proximity of candidate name occurrences to the query terms. However, as this source of evidence is specific to the expert search task, we chose not to deploy it to facilitate the inter-task cross-comparison of research question (4). Nevertheless, even without this expert search-specific evidence, our learned approach exhibits comparable results.

Overall, the experimental results allow us to conclude that our methodology for learning aggregate ranking models is effective and robust across two aggregate search tasks and four topic sets. Using a thorough analysis, we identified that the most effective features originated from the VT independent variable. However, both DM and θ also bring valuable additional sources of evidence that can successfully be integrated into the learned model.

6 Related Work

Learned approaches for aggregate ranking tasks such as expert search have seen very little published work. In this section, we review the very recent existing literature, comparing and contrasting with our own approach. In particular, in [21], Cummins et al. used genetic programming to learn a formula for the weights of document associations within the candidate profiles. This is orthogonal to our approach, where features and weights are defined and learned at the aggregate level. While the genetic programming approach appears promising, our results on the expert search tasks are 15-19% higher than the best results obtained by their learned approach, showing the superiority of our proposed approach.

Fang et al. [22] recently introduced a discriminative approach for expert search. In this approach, the importance of candidate features and association features are automatically learned from the training data. However, the mathematical machinery to the discriminative approach is complex, requiring partial derivatives to be empirically evaluated via nonlinear optimisations (the authors in [22] used a BFGS Quasi-Newton optimisation). In contrast, by defining all features at the aggregate level, our approach can easily use existing learning to rank approaches without complex derivations, while also being agnostic to various aggregation strategies (e.g. voting techniques), rather than the two used in [22].

7 Conclusions

In this work, we proposed a novel yet natural approach to learn rankings for aggregate search tasks such as expert search and blog distillation. In this approach, each feature is defined at the aggregate object level (e.g. persons or blogs), and generated by instantiations of three independent variables, namely the document ranking weighting model, its rank cutoff, and the voting technique. From these features, we showed that effective and robust ensemble models can be learned using existing learning to rank techniques. Moreover, our experimental results showed that the inclusion of multiple voting techniques - each with different ways of ranking aggregates - results in the most marked and significant increases in retrieval performance. In the future, we will continue to develop novel and effective features within our learning methodology for aggregate ranking.

References

1. Balog, K., Azzopardi, L., de Rijke, M.: Formal models for expert finding in enterprise corpora. In: Proceedings of SIGIR 2006, 43–50.
2. Macdonald, C., Ounis, I.: Voting for candidates: Adapting data fusion techniques for an expert search task. In: Proceedings of CIKM 2006, 387–396.
3. Elsas, J.L., Arguello, J., Callan, J., Carbonell, J.G.: Retrieval and feedback models for blog feed search. In: Proceedings of SIGIR 2008, 347–354.
4. Balog, K., de Rijke, M., Weerkamp, W.: Bloggers as experts: feed distillation using expert retrieval models. In: Proceedings of SIGIR 2008, 753–754.
5. Liu, T.Y.: Learning to rank for information retrieval. Foundations and Trends in Information Retrieval **3**(3) (2009) 225–331
6. Macdonald, C., Ounis, I.: Key blog distillation: ranking aggregates. In: Proceedings of CIKM 2008, 1043–1052.
7. Santos, R.L.T., Macdonald, C., Ounis, I.: Voting for related entities. In: Proceedings of RIAO 2010.
8. Macdonald, C.: The Voting Model for People Search. PhD thesis, Univ. of Glasgow (2009)
9. Pederson, J.: The Machine Learned Ranking Story. http://jopedersen.com/Presentations/The_MLR_Story.pdf (2008)
10. Metzler, D.A.: Automatic feature selection in the Markov random field model for information retrieval. In: Proceedings of CIKM 2007, 253–262.
11. Xu, J., Li, H.: AdaRank: a boosting algorithm for information retrieval. In: Proceedings of SIGIR 2007, 391–398
12. Bailey, P., Craswell, N., de Vries, A.P., Soboroff, I.: Overview of the TREC-2007 Enterprise track. In: Proceedings of TREC 2007.
13. Balog, K., Soboroff, I., Thomas, P., Bailey, P., Craswell, N., de Vries, A.P.: Overview of the TREC 2008 Enterprise track. In: Proceedings of TREC 2008.
14. Macdonald, C., Ounis, I.: The influence of the document ranking in expert search. In: Proceedings of CIKM 2009, 1983–1986.
15. Macdonald, C., Ounis, I.: On perfect document rankings for expert search. In: Proceedings of SIGIR 2009, 740–741.
16. Macdonald, C., Ounis, I., Soboroff, I.: Overview of the TREC-2007 Blog track. In: Proceedings of TREC 2007.
17. Ounis, I., Macdonald, C., Soboroff, I.: Overview of the TREC-2008 Blog track. In: Proceedings of TREC 2008.
18. Amati, G., Ambrosi, E., Bianchi, M., Gaibisso, C., Gambosi, G.: FUB, IASI-CNR and Univ. of Tor Vergata at TREC 2007 Blog track. In: Proceedings of TREC 2007.
19. Peng, J., Macdonald, C., He, B., Plachouras, V., Ounis, I.: Incorporating term dependency in the DFR framework. In: Proceedings of SIGIR 2007, 843–844.
20. Peng, J., He, B., Ounis, I.: Predicting the usefulness of collection enrichment for enterprise search. In: Proceedings of ICTIR 2009, 366-370.
21. Cummins, R., Lalmas, M., O'Riordan, C.: Learned aggregation functions for expert search. In: Proceedings of ECAI 2010, 535–540.
22. Fang, Y., Si, L., Mathur, A.P.: Discriminative models of integrating document evidence and document-candidate associations for expert search. In: Proceedings of SIGIR 2010, 683–690.