ECAI 2010
H. Coelho et al. (Eds.)
IOS Press, 2010
© 2010 The authors and IOS Press. All rights reserved.
doi:10.3233/978-1-60750-606-5-535

Learning Aggregation Functions for Expert Search

Ronan Cummins[1] and **Mounia Lalmas**[2] and **Colm O'Riordan**[3]

Abstract. Machine learning techniques are increasingly being applied to problems in the domain of information retrieval and text mining. In this paper we present an application of evolutionary computation to the area of expert search. Expert search in the context of enterprise information systems deals with the problem of finding and ranking candidate experts given an information need (query). A difficult problem in the area of expert search is finding relevant information given an information need and associating that information with a potential expert.

We attempt to improve the effectiveness of a benchmark expert search approach by adopting a learning model (genetic programming) that learns how to aggregate the documents/information associated with each expert. In particular, we perform an analysis of the aggregation of document information and show that different numbers of documents should be aggregated for different queries in order to achieve optimal performance.

We then attempt to learn a function that optimises the effectiveness of an expert search system by aggregating different numbers of documents for different queries. Furthermore, we also present experiments for an approach that aims to learn the best way to aggregate documents for individual experts. We find that substantial improvements in performance can be achieved, over standard analytical benchmarks, by the latter of these approaches.

1 Introduction

In large modern organisations, employees have often accumulated unique expertise in specific topic areas. People are a vital source of information in a topic of expertise, as they can explain aspects of a topic of interest that may otherwise be unavailable. Automatically identifying experts in a certain area, given a specific topic, is therefore a useful goal in modern information systems. Thus, the aim of an expert search system is to find and rank experts in a large corpus of semi-structured or unstructured documents given a user query.

There are several reasons why one may wish to identify people with relevant expertise in a topic rather than documented sources of information. These reasons include: a lack of access to documented forms of information, a lack of individual ability to formulate a solution to the problem being addressed, a need for explanation or reinterpretation of a certain topic and a human need for social interaction [15].

The best performing approach to the expert search task is the two-stage document centric model. In this approach, all the documents in a corpus are scored and ranked using a state of the art ranking function (e.g. $BM25$). Then, the top N document scores associated with a candidate expert are aggregated in order to rank the experts. This simplistic approach has been shown to be more effective than other approaches for the expert search task for a wide variety of retrieval functions and parameter settings.

In this paper, we adopt a genetic programming approach to learn functions that can best aggregate the on-topic information in documents. The number of documents to aggregate and the weighting for these documents is very important to the overall performance of the model. We show that the optimal number of documents to aggregate differs for individual queries. The contributions of this paper are three-fold:

- We show that the number of documents to aggregate is an important factor in the effectiveness of an expert search system (section 3.2).
- We investigate a number of features to determine if they are useful in predicting the optimal number of documents to aggregate for the expert search task (sections 3.3 and 3.4).
- We present results from two approaches using genetic programming that aim to learn how to combine features for use in effective document aggregation functions for the expert search task (section 5).

The remainder of the paper is organised as follows: Section 2 outlines some background and related research in the theory and practice of expert search. Section 3 presents a preliminary analysis of the problem that further motivates our subsequent experiments. Section 4 outlines the framework in which we attempt to learn expert search aggregation functions. Section 5 presents the experiments and results. Finally, Section 6 comprises our conclusions and future work.

2 Related Research

The expert search task of the enterprise track of TREC [3, 14] has been run since 2005 and provides a corpus, topics and associated relevant experts to enable researchers to develop techniques in advancing the area of expert search. The evaluation metrics used in this task are similar to those used in the standard IR (information retrieval) document retrieval task except that the metric measures relevant candidate experts rather than relevant documents.

One of the first steps in an expert search system is to identify candidate (or potential) experts in the corpus. This is typically done by extracting the email address and/or names in the '*mailto:*' tags within a document. In large semi-structured document collections it is necessary to gather email addresses and other features (e.g. first name and surname) that may uniquely identify a possible expert candidate. This is not a trivial problem and is called *'named entity recognition'* in the domain of information extraction but is beyond the scope of this paper.

Associating candidates to the information in the corpus has also been studied and although proximity has been shown to be a use-

[1] University of Glasgow, Scotland, email: ronanc@dcs.gla.ac.uk
[2] University of Glasgow, Scotland, email: mounia@acm.org
[3] National University of Ireland, Galway, Ireland

ful heuristic [13], candidates are usually deemed to be associated with documents to some degree if a candidate identifier appears in the document. Various models of expert search have been formally studied [2] and it is concluded that the document-centric approach outperforms other approaches to expert search for most parameters settings and performance metrics. The document-centric approach to expert search first ranks the documents in the collection with respect to the topic using a standard term-weighting scheme (e.g. $BM25$). Then, it aggregates the *score* of the documents that are associated with a candidate to produce a final ranking of candidates. Recent research [11, 12] has modelled this approach as a voting problem and researched various strategies of aggregating the strengths of votes of documents for specific candidates. Many fusion techniques have been experimented with that deal with the aggregation of document scores in order to promote a candidate.

Using notation similar to that previously used [12], $R(Q)$ denotes a ranked list of documents returned with respect to a query Q and $D(x_i)$ denotes the set of documents associated with a candidate expert x_i. In this approach, scores from a single ranked list of documents are aggregated to score a candidate expert. The following fusion (or aggregation) technique combines the scores of documents associated with a candidate when matched against a specific topic:

$$combSUM(Q, x_i, N) = \sum_{d_j \in R(Q) \cap D(x_i)}^{N} (S(Q, d_j)) \qquad (1)$$

where x_i is candidate expert i, d_j is document j, Q is a query (topic), $D(x_i)$ is the set of documents associated with x_i, $R(Q)$ is the ranking of documents (i.e. an ordered set) when given query Q and $S(Q, d_j)$ is the score of document d_j given query Q. Thus, $combSUM$ is a summation of the top N document scores associated with the candidate x_i.This two-stage approach has consistently outperformed other approaches (e.g. Model I [2]) on TREC data over the years. It is trivial to show that by changing the number of aggregated documents, one can easily change the ranking of candidate experts. We can also surmise that if N is large, it will tend to promote experts who are associated with a larger number of documents, as all documents with a non-zero score will be aggregated to increase the score of the candidate. This may not be desirable in certain cases.

Traditional genetic algorithms (GAs) have been adopted by some to learn term-weights that are useful for retrieval for a specific collection [6]. A genetic programming (GP) approach to evolving term-weighting schemes in IR has previously been adopted in a number of works [5, 4]. These GP approaches are a useful form of off-line learning as they can often produce formulae that outperform human-designed solutions. Furthermore, they produce a symbolic representation of the solution that is useful for future analysis. However, no work to date has attempted to learn functions for the expert search problem using evolutionary computation.

3 Preliminary Analysis

In this section, we perform a preliminary analysis to show that different aggregation methods are optimal for different queries. We also perform an analysis of query-based and expert-based features in order to determine if certain features are correlated with an optimal aggregation strategy.

3.1 Data & Experimental Setup

The data used in this work are the TREC collections from 2005 to 2008. TREC 2005 and 2006 use the W3C collection and two sets of topics. TREC 2007 and 2008 use the CSIRO collection and two sets of topics. Table 1 shows some of the characteristics of the test collections.

Table 1. TREC Test Collections

	W3C		CSIRO	
	TREC 2005	TREC 2006	TREC 2007	TREC 2008
# of Documents	297,110	297,110	370,716	370,716
# of Candidates	1,092	1,092	2,910	2,910
# of Docs with Candidates	103,291	103,291	108,159	108,159
# of Topics	50	49	50	55

For all the analysis and experiments in this paper, we use the well-known $BM25$ function as the initial ranking function. Therefore, all of the aggregation functions, ranks and scores are based on the $BM25$ function. Furthermore, we use $*$ to denote statistical significance at the 0.05 level and $**$ to denote statistical significance at the 0.01 level (two-tailed t-test). The MAP (mean average precision) of relevant experts is used as the main measure of performance as it is a standard retrieval metric that is stable and well-known in the area.

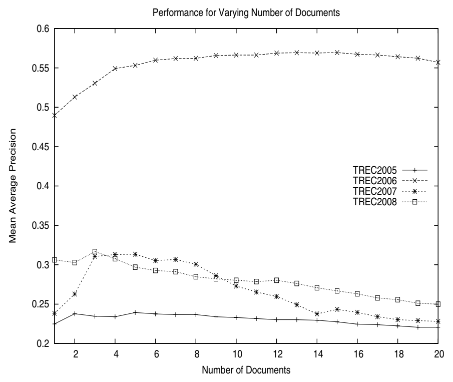

Figure 1. Performance (MAP) for varying N on different collections

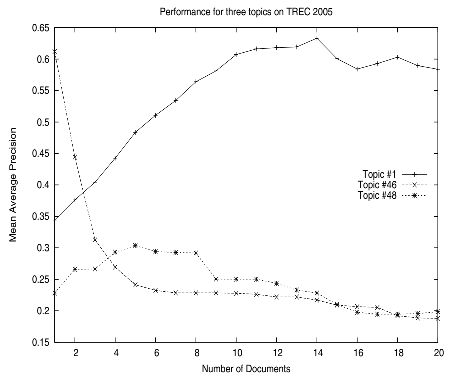

Figure 2. Performance (MAP) for varying N on different topics

As described in section 2, the aggregation function (equation 1) for the expert search model is used to combine information from different documents. We wish to determine how the effectiveness of the expert search task changes as we change the number of documents (N) to use in the aggregation process. Figure 1 shows the MAP (Mean Average Precision) of the aggregation function ($combSUM$) for different values of N on the four TREC collections. We can see that the performance varies quite significantly for various N. Using $N = 1$ (i.e. only using the top associated document to rank the candidates) penalises experts that may have a number of documents that are highly related to the topic. For all of the collections, choosing an N greater than 1 (i.e. the top ranked document) improves performance. However, if N is increased beyond a certain point, performance decreases for all collections.

Varying the number of documents to be aggregated has not been widely studied, although it has been noted by some researchers that performance differs as N changes on a specific collection [9]. The best *mean* MAP across all the TREC collections is when $N = 5$ and indeed, it is optimal on two of the four collections. This is a useful benchmark function for later experiments and it is worth noting that it has been found only after an exhaustive search.

Furthermore, Figure 2 shows that the optimal value of N is also very different for various topics within the same collection. We show a sample of three topics for the TREC 2005 dataset. We can see that for these topics, the performance varies substantially as the number of documents used in the aggregation (N) changes. Therefore, if the most optimal value of N for individual queries for a specific collection could be predicted, we could substantially improve the performance of the expert search system. This phenomenon has not been reported before. Table 2 shows the performance improvement that could be achieved if we could predict the best N to use for each query for each of the four collections. The optimal strategy is labelled $N = opt(Q)$. The optimal strategy (i.e. $N = opt(Q)$) is found using an exhaustive search for N for values from 1 to 20 for each query on each collection[4].

Table 2. MAP for different aggregation functions on TREC data

	W3C	CSIRO		
N	TREC 2005	TREC 2006	TREC 2007	TREC 2008
$N = 1$	0.2249	0.4896	0.2382	0.3062
$N = 5$	0.2392	0.5531	0.3133	0.2969
$N = opt(Q)$	0.2811$**$	0.6038 $**$	0.3776 $**$	0.3817$**$

Another observation is that the performance of topics on TREC 2006 is substantially greater than any of the other years. The optimal value for N ($N = 15$) on this data (TREC 2006) is also substantially larger than the other collections. This may suggest that the performance of the topic is related to the optimal number of documents to aggregate. The intuition is that for easy queries (i.e. those that have a high performance in terms of MAP) there are more relevant document in the high ranks. Therefore, by including more documents in the aggregation process (i.e. choosing a higher value of N), more of these *relevant* documents will be included in the aggregation process for each expert. It is ultimately the aggregation of *relevant* information that is crucial in determining the relevance of an expert.

3.2 Individual Query Features

We have shown in the previous section that the performance of different queries are optimal for different values of N. Therefore, if we wish to predict the best N for a particular query, we must look at features which vary across different queries. A number of query-based feature have previously been developed to predict the performance of individual queries [8, 7]. A number of these have been shown to be correlated with query performance.

We now look at these query-based features to see if they are correlated with the optimal value of N for each query. Table 3 shows the Spearman correlation of a number of query-based features with the optimal value of N used in the aggregation function for each query. Firstly, none of the measures are strongly correlated (i.e. 0.5 - 1.0) with the optimal value of N per query and secondly, only a few of the measures are weakly correlated (i.e. 0.0 - 0.5) with an optimal value of N to a significant degree.

It would seem from this analysis that the features chosen may not, in general, be able to predict the optimal number of documents to aggregate for each query. However, it may be possible that certain non-linear combinations of such features can be discovered by a learning approach that may, in turn, be correlated with an optimal aggregation strategy for each query. We return to this problem in section 4.1.

3.3 Individual Expert Features

Previously, we have shown that different queries have different optimal document aggregation functions. Aggregating different numbers of documents for different candidate experts may also lead to improved performance. This would essentially mean choosing a different N for each expert. However, an exhaustive search for the optimal values of N (i.e. the number of documents to include in the aggregation) for a set of experts is extremely costly as there are over a thousand experts in each corpus and the experts are ranked with respect to each other. This creates a huge number of possible combinations and therefore an optimal strategy is extremely difficult to find (although an optimal must exist).

Following this argument, we now chose a number of expert-based features (i.e. features that change from expert to expert). We analyse these expert-based features by examining them with respect to relevant and non-relevant experts on a number of topics. The features chosen relate to individual experts and to the list of documents that can possibly be aggregated for each expert. The $top\_exp\_rank_{x_i}$ feature is the rank of the highest ranked associated document for a specific expert x_i. The $no\_docs_{x_i}$ feature is the maximum number of documents that could possibly be aggregated for a specific expert and the $top\_exp\_score_{x_i}$ feature is the score of the highest ranked associated document for an expert.

To analyse these simple features, we take the top 50 experts for each topic (ranked using the baseline approach $combSUM$ where $N = 5$) and analyse their associated documents. Table 4 shows the three expert related features used, where R is the set of relevant candidate experts and NR is the set of non-relevant candidates. The $\%$ column shows the consistency of the relationship $R < NR$ for the features across the topics in the data set. For example, for the $top\_exp\_score_{x_i}$ feature on TREC 2008, the % of topics where a relevant candidate's top document score is a *lower* than a non-relevant candidate's top document score, is only 4%.

The table shows that for all of the collections, relevant candidate experts are associated with a higher ranked document than non-relevant candidate experts ($top\_exp\_rank_{x_i}$). We can also see that the score of the highest ranked document associated with a candidate expert is greater for relevant candidate experts ($top\_exp\_score_{x_i}$). However, more interestingly, it can be noticed that relevant candidate experts are associated with less documents ($no\_docs_{x_i}$) on average

[4] Although we only report results from $BM25$, we can also confirm that other ranking functions tested behave similarly and have optimal values of N that are highly correlated with those found using $BM25$.

Table 3. Spearman correlation of Query-Based Features with the optimal N

		W3C		CSIRO	
Feature	Description	TREC 2005	TREC 2006	TREC 2007	TREC 2008
idf_{min}	min idf query term	-0.20	-0.02	-0.32*	0.16
idf_{max}	max idf query term	-0.26	-0.01	-0.33*	-0.05
idf_{dev}	std dev of idf of query terms	0.06	0.04	-0.07	-0.10
SCS	query clarity score	-0.27 *	-0.09	-0.36*	0.04
ql	query length	0.15	0.05	-0.02	-0.02
top_score	score of top ranked doc	-0.08	0.17	-0.23	-0.01
$query\_scope$	fraction of docs returned	0.07	-0.04	0.31*	-0.09

for three of the four test collections. For the TREC 2006 topics, it can be seen that relevant experts are associated with more documents, on average, than non-relevant ones (for 58% of the topics on this collection). This explains why a high value of N increases performance for this data set only (TREC 2006) as aggregating more documents will promote candidates that are associated with many documents (see end of section 3.2). Now that we have outlined a number of expert-based features, we can attempt to incorporate them into an aggregation function.

4 Learning Framework

In this section we outline the framework in which we automatically learn aggregation functions for our expert search approach. In particular, we can now outline two separate approaches. Our first approach aims to learn the optimal number of document to aggregate for a specific query (query-based aggregation). Our second approach aims to learn the optimal number of documents to aggregate for each specific expert (expert-based aggregation).

4.1 Learning Query-Based Aggregation Functions

We can view the learning of an aggregation function that aims to learn the best N as a weighting problem. Figure 3 shows an example of how such a function may look compared to a discrete function that either includes the document in the aggregation (i.e. 1) or ignores the document (i.e. 0). The x-axis shows the rank of each document for a query, while the y-axis shows the weighting (i.e. 1 to include the document and 0 if we ignore the document). A continuous function (as shown in Figure 3) is more suitable for a number of reasons: (1) it is more expressive as it allows different weightings to be applied to the ranked document scores and (2) it is more informative for use with a supervised learning approach. In our approach, we allow the top 30 documents associated with each candidate expert to be weighted accordingly for the learning approach. The top 30 documents will afford the learning approach ample information to weight and aggregate these documents correctly.

Our learning approach will learn the best $weight(Q, d)$ in the following framework:

$$combSUM(Q, x_i, N) = \sum_{d \in R(Q) \cap D(x_i)}^{N} (weight(Q, d) \cdot S(Q, d)) \tag{2}$$

where $weight(Q, d)$ is the weighting given to a document d with regard to certain features of Q and $S(Q, d)$ is the score of a document d with regard to the query Q.

In order to learn this function shape for individual queries (i.e. a different function for each query-based on the query features),

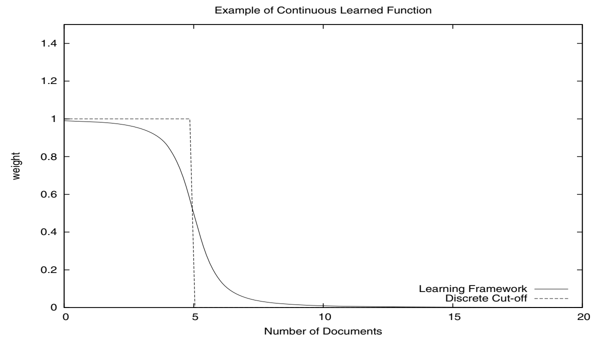

Figure 3. Weighting Documents Using a Continuous Function

we use genetic programming (GP) [10]. Genetic programming is a population-based learning paradigm that can automatically define functions given parameters and functions as inputs. It optimises for a particular performance metric (i.e. MAP for the expert search problem) on sample of training data using a *survival of the fittest* approach and produces a symbolic representation of the function. We use all of the features outlined in Table 3 (first column) and the document rank as inputs to our learning process. We also use the following functions: $\{ \times, \surd, /, +, -, exp(), log(), sq() \}$ as inputs to the GP.

4.2 Learning Expert-Based Aggregation Functions

Similarly to section 4.1, we can view the learning of an aggregation function that aims to learn the best N for individual experts as a weighting problem and model it as follows:

$$combSUM(Q, x_i, N) = \sum_{d \in R(Q) \cap D(x_i)}^{N} (weight(x_i, d) \cdot S(Q, d)) \tag{3}$$

where $weight(x_i, d)$ is the weighting given to a document d with regard to certain features of x_i (i.e. the i^{th} candidate expert). Again, we allow the learning approach to use the top 30 documents associated with each expert to be weighted accordingly. The inputs to the GP for this approach are the expert features outlined in Table 4 and the current document rank (R). We also use the following functions: $\{ \times, \surd, /, +, -, exp(), log(), sq() \}$ as inputs to the GP. This variation of the problem can be viewed similarly to a normalisation problem for each candidate expert (as different candidate experts have different numbers of associated documents). Recent work [1] has indicated that a simple inverse document frequency (idf) type feature for the

Table 4. Expert-Based Features correlation with Relavant (R) and Non-Relevant Experts (NR)

	W3C						CSIRO					
Features	TREC 2005			TREC 2006			TREC 2007			TREC 2008		
	R	NR	%	R	NR	%	R	NR	%	R	NR	%
$top\_exp\_rank_{x_i}$	43.9	79.6	87%	66.2	104.3	91%	18.5	122.9	94%	40.7	143.3	98%
$no\_docs_{x_i}$	1106	1602	74%	1508.5	1277.8	42%	59.0	479.6	90%	362.2	911.2	94%
$top\_exp\_score_{x_i}$	6.37	5.94	20%	6.42	5.71	7%	8.82	6.38	5%	7.51	5.58	4%

candidate expert has been useful way of normalising expert scores on some TREC collections.

5 Experiments

In this section, we will present the results of the two learning approaches to aggregating documents. We use the TREC collection from 2005 as the training collection and the remaining data as the test data. As a result, we are likely to see improvements for the learning approach on TREC 2005. However, more important are the results on the unseen test data (TREC 2006-2008). We also report precision at 10 (P@10) experts in parenthesis in the tables in this section. As benchmark aggregation approaches, we use $combSUM$ (equation 1) when $N = 1$ and $N = 5$. Statistical significance is measured against the best benchmark ($N = 5$). We also use the best normalisation approach (i.e. an idf normalisation approach) outlined in recent research [1]. For each of the two approaches undertaken in this work (query-based and expert-based), we report the results from the best two runs (out of four) of the GP. All GP runs use a population of 2000 randomly generated individuals run for 50 generations and optimsed for MAP.

5.1 Results: Query-Based Aggregation

Table 5. MAP (P@10) for query-based functions on TREC training and test data

	W3C		CSIRO	
Scheme	**TREC 2005**	TREC 2006	TREC 2007	TREC 2008
$N = 1$	0.2249 (0.292)	0.4896 (0.585)	0.2382 (0.112)	0.3062 (0.289)
$N = 5$	0.2392 (0.33)	0.5531 (0.642)	0.3133 (0.132)	0.2969 (0.303)
idf_{xi}	0.2021 (0.298)	0.3373 (0.367)	0.2815 (0.108)	0.3299 $*$ (0.301)
QGP_2	0.2659 (0.354)	0.5449 (0.646)	0.2570 (0.114)	0.3127 (0.327)
QGP_4	0.2652 $**$ (0.348)	0.5457 (0.630)	0.2578 (0.122)	0.3023 (0.298)

Table 5 shows the results of the best two GP runs (QGP_2 and QGP_4) for the query-based aggregation approach. We can see that there is a significant improvement on the training data (TREC 2005) for the learned functions. However, this does not generalise to unseen data. Although all learned functions improve performance over a poor baseline aggregation function (i.e. when $N = 1$), they are not better than the best benchmark ($N = 5$). This seems to confirm the previous analysis (section 3.3) that showed that the query-based features were not strongly correlated to the optimal value for N. Furthermore, it suggests that any combination (even non-linear) of the query-based features is not likely to bring about an improvement in performance. Therefore, although different queries have different optimal numbers of documents for aggregation, it cannot be predicted on a per-query basis by the features we have examined.

Table 6. MAP (P@10) for learned expert-based functions on TREC training and test data

	W3C		CSIRO	
Scheme	**TREC 2005**	TREC 2006	TREC 2007	TREC 2008
$N = 1$	0.2249 (0.292)	0.4896 (0.585)	0.2382 (0.112)	0.3062 (0.289)
$N = 5$	0.2392 (0.33)	0.5531 (0.642)	0.3133 (0.132)	0.2969 (0.303)
idf_{xi}	0.2021 (0.298)	0.3373 (0.367)	0.2815 (0.108)	0.3299 $*$ (0.301)
GP_2	0.2821 $**$ (0.386)	0.5466 (0.604)	0.3546$*$ (0.130)	0.3634 $**$ (0.345)
GP_3	0.2843 $**$ (0.38)	0.5185 (0.589)	0.3426 (0.132)	0.3697 $**$ (0.361)

5.2 Results: Expert-Based Aggregation

Table 6 shows the results of the best two runs of the GP for the expert-based aggregation approach. Firstly, we can see that the performance increases significantly on the training data. Furthermore, on TREC 2007 and 2008 test data, performance also increases significantly. We can see that for TREC 2006 (which achieves an exceptionally high MAP overall) the performance decreases slightly. In general for the expert-based approach, the learned aggregation functions tend to improve performance over a good benchmark function ($N = 5$). Interestingly, the idf approach, which normalises the experts based on the number of documents in which an expert x_i occurs ($no\_docs_{x_i}$), is poor on all but one collection.

5.2.1 *Robustness*

As we have seen, choosing the most optimal N document to aggregate significantly improves performance. We have also seen that choosing the best N is not a trivial problem. Therefore, a robust (or at least predictable) aggregation function conveys many advantages. In this section, we analyse the performance of the expert-based aggregation functions for vastly different values of N.

Figures 4 and 5 show the robustness of the learned schemes for varying numbers of documents (N). We can see that for nearly all of the collections the performance of the learned schemes (GP_2 and GP_3) increase as N increases. Both learned schemes have a poor performance at $N = 1$ but as more documents are included the performance increases and is quite predicable and stable (i.e. a single high value of N can be used with these learned schemes for good stable performance). This is most likely because these aggregation functions were learned when N was high ($N = 30$), and therefore learned how to properly weight documents at these lower ranks. Conversely, we can see that if one aggregates documents using the baseline approach (denoted $BM25$), the performance tends to peak early and then degrades.

Another interesting observation is that the learned schemes approach the performance of the best benchmark on the TREC 2006 data as the number of documents increases. Indeed, we have already indicated (section 3.4) that for this collection (TREC 2006), candidate experts with more associated documents are more relevant. However, it is also interesting that there is only a small drop (if any) in performance on all the other TREC collections when using a large number of documents for the aggregation (e.g. N = 60). From inspection of our learned formula (shown as GP_2), it can be shown that the $no\_docs_{x_i}$ feature (i.e. the number of documents associated with a expert) is used to normalise the scores for each expert. This prevents experts, that are associated with many documents, being unfairly promoted. The learning approach adopted has learned to combine this in a way so that documents are weighted correctly at lower ranks (e.g. 30, 40, 50 and 60 etc). This is also achieved using the rank (R) of the document.

$$GP_2 = \frac{\sqrt{\sqrt{2/no\_docs_{x_i}/(\sqrt{(10/R)}+R)}}}{\sqrt{sq(10/R)+R+sq(10/R)+\sqrt{R}*2}} \tag{4}$$

where R is the rank of the document in the initial ranking and $no\_docs_{x_i}$ is the total number of documents associated with expert x_i.

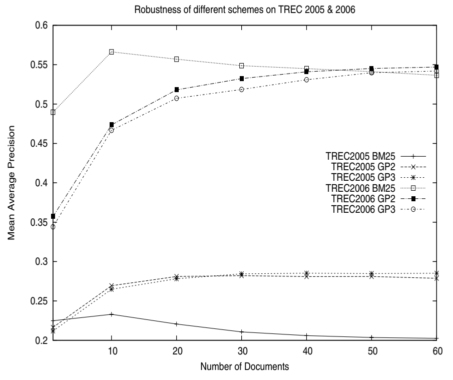

Figure 4. Robustness of aggregation functions (TREC 2005 & 2006)

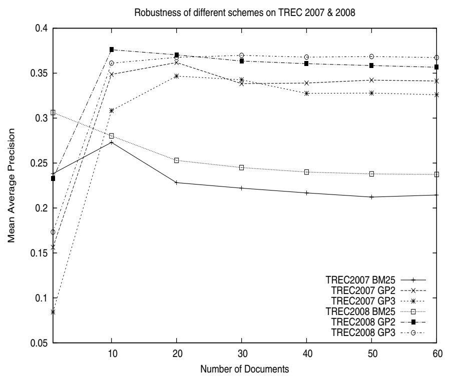

Figure 5. Robustness of aggregation functions (TREC 2007 & 2008)

6 Conclusion

We have shown that performance gains can be achieved by modifying the number of documents chosen for use in the aggregation function of an expert search system. We have performed an analysis of two approaches to aggregating document information in the expert search task. We used a learning approach to find suitable aggregation functions for both approaches. We have shown that there is no improvement when using query performance predictors to estimate a suitable aggregation function for individual queries.

However, we have shown that genetic programming can find useful aggregation functions on a per expert basis. These function outperform other analytical *idf* normalisation expert search approaches outlined in the literature. We have also shown that the aggregation functions produced from this learning approach are robust when using larger numbers of documents in the aggregation process.

7 ACKNOWLEDGEMENTS

Ronan Cummins is funded by the Irish Research Council for Science, Engineering and Technology, co-funded by Marie Curie Actions under FP7. Mounia Lalmas is currently funded by Microsoft Research/Royal Academy of Engineering.

REFERENCES

[1] K. Balog and M. de Rijke, ‘Associating people and documents’, in *30th European Conference on Information Retrieval (ECIR 2008)*, pp. 296–308, Glasgow, (April 2008).

[2] Krisztian Balog, Leif Azzopardi, and Maarten de Rijke, ‘Formal models for expert finding in enterprise corpora’, in *SIGIR ’06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 43–50, New York, NY, USA, (2006). ACM.

[3] Nick Craswell and Arjen P. De Vries, ‘Overview of the trec-2005 enterprise track’, in *In The Fourteenth Text REtrieval Conf. Proc. (TREC*, (2005).

[4] Ronan Cummins and Colm O’Riordan, ‘Evolving local and global weighting schemes in information retrieval’, *Information Retrieval*, **9**(3), 311–330, (2006).

[5] Weiguo Fan, Ming Luo, Li Wang, Wensi Xi, and Edward A. Fox, ‘Tuning before feedback: combining ranking function discovery and blind feedback for robust retrieval’, in *the Proceedings of the 27th Annual International ACM SIGIR Conference*, U.K., (2004). ACM.

[6] M. Gordon, ‘Probabilistic and genetic algorithms in document retrieval’, *Commun. ACM*, **31**(10), 1208–1218, (1988).

[7] Claudia Hauff, Djoerd Hiemstra, and Franciska de Jong, ‘A survey of pre-retrieval query performance predictors’, in *CIKM ’08: Proceeding of the 17th ACM conference on Information and knowledge management*, pp. 1419–1420, New York, NY, USA, (2008). ACM.

[8] Ben He and Iadh Ounis, ‘Query performance prediction’, *Inf. Syst.*, **31**(7), 585–594, (2006).

[9] Jiepu Jiang, Wei Lu, and Dan Liu, ‘Csir at trec 2007 expert search task’, in *TREC*, (2007).

[10] John R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, 1992.

[11] Craig Macdonald and Iadh Ounis, ‘Voting for candidates: adapting data fusion techniques for an expert search task’, in *CIKM ’06: Proceedings of the 15th ACM international conference on Information and knowledge management*, pp. 387–396, New York, NY, USA, (2006). ACM.

[12] Craig Macdonald and Iadh Ounis, ‘Searching for expertise: Experiments with the voting model’, *The Computer Journal*, (2008).

[13] Desislava Petkova and W. Bruce Croft, ‘Proximity-based document representation for named entity retrieval’, in *CIKM ’07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pp. 731–740, New York, NY, USA, (2007). ACM.

[14] Ian Soboroff, Arjen P. De Vries, and Nick Craswell, ‘Overview of the trec-2006 enterprise track’, in *In The Fifthteenth Text REtrieval Conf. Proc. (TREC)*, (2006).

[15] Dawit Yimam-seid and Alfred Kobsa, ‘Expert finding systems for organizations: Problem and domain analysis and the demoir approach’, *Journal of Organizational Computing and Electronic Commerce*, **13**, 2003, (2002).