Combining Families of Information Retrieval Algorithms using Meta-Learning

Michael Cornelson, Robert L. Grossman, Ed Greengrass
Ron Karidi and Dan Shnidman

August 14, 2002

This is a draft of the paper M. Cornelson, E. Greengrass, R. L. Grossman, R. Karidi, and D. Shnidman, Combining Information Retrieval Algorithms Using Machine Learning, Survey of Text Mining: Clustering, Classification, and Retrieval Michael W. Berry, editor, Springer-Verlag, 2003.

Abstract

This paper describes some experiments which use meta-learning to combine families of information retrieval (IR) algorithms obtained by varying the normalizations and similarity functions. By meta-learning, we mean the following simple idea: a family of IR algorithms is applied to a corpus of documents in which relevance is known to produce a learning set. A machine learning algorithm is then applied to this data set to produce a classifier which combines the different IR algorithms. In experiments with TREC-3 data, we could significantly improve precision at the same level of recall with this technique. Most prior work in this area has focused on combining different IR algorithms with various averaging schemes or has used a fixed combining function. The combining function in meta-learning is a statistical model itself which in general depends on the document, the query, and the various scores produced by the different component IR algorithms.

1 Introduction

This paper describes some experiments which use meta-learning to combine families of information retrieval (IR) algorithms obtained by varying the normalizations and similarity functions. In experiments with TREC-3 data, we could significantly improve precision at the same level of recall with this technique.

In more detail, our goal is to satisfy a query q by returning a ranked list of documents d from a collection C which are relevant to the query. Our point of view is that there is no single best algorithm for solving this problem. On the contrary, we assume that we have m algorithms $A_1, \ldots, A_m$ and that some work well for certain queries and certain sets of documents, while others work well for other queries and other document sets.

Our approach is to apply meta-learning [2, 13] to learn how to select which algorithm A_j or combination of algorithms to use, where the selection is a function of the query q and the document d. In more detail, we assume that we are given a collection C of documents d and queries q for which we know the relevance; that is, for each document-query pair (d, q), C also contains a human judgment of whether or not d is relevant to q. This is the "training set" from which we will learn how to select the best algorithm(s). If we apply all the algorithms $A_1, \ldots, A_m$ to each document-query pair (d, q), this produces a learning set ML. ML denotes meta-learning set, a terminology which we will explain below. Our approach is to use machine learning to create a predictive model from ML which can be used on previously unseen queries and documents to determine which algorithm or combination of algorithms to use to determine relevance. We assume that each algorithm A produces a score $A(d, q)$ between 0 and 1 when applied to a document d and query q. The higher the score the more relevant the document. In general, the resulting ML will contain, for each document-query pair (d, q), a set of features characterizing d and q, the relevance score computed by each of the algorithms A_i, and the human relevance judgment or "truth."

We introduce this approach in this paper and describe an experimental study which provides evidence regarding its effectiveness. We confine ourselves to the simplest case in which we use the scores $A(d, q)$ themselves and not features of the query or the document to determine which algorithm to use. (But it should be stressed that we do NOT use the relevance judgments in selecting an algorithm for a given test document. The model we build is genuinely predictive, not retrospective.) We will treat the more general case where we use the document and query features as the independent (predictor) variables in a subsequent paper.

Our point of view is that meta-learning provides a natural framework to combine similar or quite different IR algorithms. We believe that the following aspects of this work are novel:

1. Our meta-learning approach allows us to select a distinct algorithm or combination of algorithms for each new document/query pair submitted to our meta-model. Most previous work in algorithm combination assumes a fixed combining function, which in most cases is simply an average of the scores.

2. Our approach allows us to create a combining function that summarizes variability across both documents and queries, while prior work of which we are aware has aggregated data across documents alone.

3. We have investigated several nonlinear combining functions, while prior work of which we are aware has investigated linear combining functions.

This paper is organized as follows: Section 2 describes related work. Section 3 describes background work in information retrieval, while Section 4 describes background work in data mining. Section 5 describes our implementation. Section 6 contains our experimental results. Section 7 contains the summary and conclusion.

The background material in Section 3 and 4 is provided so that researchers in the data mining community can easily read this paper and vice versa. Since the material is short, we feel that this is justified, although we realize we risk the ire of both communities for reviewing this material.

2 Related Work

There has been some prior work on combining information retrieval algorithms.

A number of researchers have conducted experiments to study the effect and possible benefit of satisfying a given information need from a given document collection by combining the results of multiple IR algorithms. Typically, two [8] to six [9] algorithms have been combined. In most cases, each algorithm is applied to a given collection, for a given query, resulting in a ranked list of document scores for each algorithm. Then, some function is evaluated that combines the scores computed by the different algorithms for a given document. Often, the scores are normalized before combination. The end result is a single ranked list of document scores. The combination formula is usually a function of the scores computed by the participating algorithms and (sometimes) the number of scores, i.e., the number of algorithms that retrieved a given document. In most studies averages of one type or another are used to combine the scores.

Multiple algorithms can be obtained by using the same basic IR method, e.g., the vector space method with cosine similarity, but executing the method with multiple term weighting schemes [8, 9]. Instead of varying the term weighting scheme, one can vary the kind of terms used as document/query descriptors, e.g., words vs. phrases. [11]. Alternatively, different basic IR methods can be employed, e.g., P-norm extended Boolean and vector similarity [3].

Hull [7] investigating combining three learning algorithms (nearest neighbor, linear discriminant and a neural network) with one IR algorithm (Rocchio query expansion) by averaging the scores produced by the different algorithms.

Some limited theoretical analyses of these combination results have been performed. Lee [8] studied the properties of several classes of term weighting scheme, and offered reasons for expecting that each scheme would favor a different class of documents so that certain pairs of weighting functions, each drawn from a different class should perform better than one weighting function alone. Vogt and Cottrell [16] studied the properties that IR systems should possess in order that linear combinations (that is, linear combinations of their normalized scores) should produce a better document ranking than either system by itself. Their results were limited to pairs of system and linear combination. The performance measures they use as predictors of effective combined performance include both measures of individual systems, e.g., average precision, and pairwise measures, e.g., Guttman's Point Alienation (GPA), a measure of how similar the document rankings produced by two systems are to each other.

On rare occasions, an effort is made to develop a meta-model, i.e., to select the best IR model(s) for a given document. For example, Mayfield et al. [11] ran a test where the document and query vectors were based on either words or phrases, depending on the document to be evaluated. They found that for some queries, this strategy produced a significant gain, while for others it did not or actually degraded performance. Unfortunately, they did not develop a meta-model that would accurately predict when the strategy should be employed [10].

3 Information Retrieval

For completeness and to fix notation, in this section we briefly review some standard definitions and techniques in information retrieval (IR) for those not familiar with this subject. For additional information, see [4].

Setup. Given a corpus of C documents d and a query q, the goal is to return the documents which best satisfy the query. Assume that r of the p documents returned by the IR system are relevant to the query and that overall R of the documents in the corpus C are relevant to the query. Typically, the IR system returns a list of documents, ordered (ranked) according to the probability or degree of relevance to q, as measured by some system metric. The system may return, or the user may examine, all documents for which the system-computed metric exceeds a specified threshold. Let p be the number of documents above this threshold. The recall is defined to be r/R, while the precision is defined to be r/p. The goal is to find algorithms with high precision or high recall, or some desired tradeoff between the two.

Vector Space Method. Rather than work with the documents directly, we define a feature vector $F(d) \in R^n$ for each document d, as is standard [14]. If we view a query as a document, then in exactly the same way we can define a feature vector $F(q)$ of the query q. Fix a relevance measure

$$m(u, v) : R^n \times R^n \longrightarrow [0, 1]$$

which ranges from 0 to 1, with 1 indicating that u and v are highly relevant and 0 indicating that they are not. Given a query q, our strategy is to return the p documents d such that $u = F(d)$ is closest to $v = F(q)$ with respect to the measure $m(\cdot, \cdot)$. These are simply the p highest scoring document-query pairs.

Document-term matrix. A document-term matrix is created from a collection of documents and contains information needed in order to create a feature vector $F(d)$ for a document d. From the collection C of documents, we create what is called a dictionary by extracting terms, where the terms are used to characterize the documents. For example, the terms can consist of words in the collection, phrases in the collection, or n-grams in the collection. An n-gram consists of n-consecutive characters, including white space and punctuation. From this data, we form a document-term matrix $d[i, j]$, where, at the simplest, $d[i, j]$ represents the number of times the term j occurs in document i. Usually, a weighting is used to normalize the terms in the document-term matrix.

In practice, not all terms end up in the dictionary. Certain common terms, e.g., articles, prepositions, etc., occur so widely in the corpus that they have little value for classifying document relevance relative to a given query. Hence, such words are placed in a "stop list." Words in the stop list are excluded from the dictionary. Also, it is common to use a stemming algorithm to reduce words into a common root and only include the root in

Euclidean	$1/[sqrt(\sum((FV1_i - FV2_i)^2))) + 1)]$
City Block	$1/[\sum(FV1_i - FV2_i)) + 1)]$
Maximum	$1/[max(FV1_i - FV2_i)) + 1)]$
Dice's Coefficient	$2 * w/(n_1 + n_2)$
Jaccard's Coefficient	$w/(N - z)$

Table 1: This table lists some common similarity functions. These functions were used in our meta-learning experiments. We used the transformation $s = 1/(d + 1)$ to define a similarity function from a distance function [15]. Here n_1 equals the number of non-zero terms in FV_1, n_2 equals the number of non-zero terms in FV_2, and w equals the number of terms common to FV_1 and FV_2. Also z equals the number of distinct terms that are neither in FV_1 nor in FV_2, and N equals the total number of distinct terms in the FV space. Therefore, $N - z$ equals the total number distinct terms that occur in FV_1 or FV_2 or both.

the dictionary. For n-grams, stop-lists and stemming are inapplicable, but statistical techniques can be employed to eliminate n-grams having little value for predicting relevance.

Normalization. In the Table 1 below, we list some of the common term weighting functions (normalizations) used. In our meta-learning experiments, we will vary these normalizations.

Similarity Scores. In the Table 2 below, we list some common similarity metrics found in the literature, i.e., metrics which are used to measure the similarity of two feature vectors, FV_1 and FV_2. Distance metrics d have been converted to similarity metrics by the transformation $s = (d+1)^(- 1)$ [15]. In our meta-learning experiments, we will vary these similarity scores.

4 Meta-Learning

For completeness and to fix notation, in this section we briefly review some standard definitions and from meta-learning for those not familiar with this subject. For additional information, see [2].

Supervised Learning. In supervised learning, we are given a data set of pairs (x, y) called a learning set. Commonly, $x \in R^n$ is an n-dimensional feature vector containing the independent variables or attributes, and $y \in \{0, 1\}$ is the dependent variable or truth. The goal is to construct a function

term frequency	tf
word count	tf/wc
log	$\log(tf + 1)$
binary	$0 or 1$
maximum normalization	tf/TF
average normalization	$tf/(TF - \text{avg}(tf))$
Tf*IDF	$tf * (N/n)$
Tf*ln(IDF)	$tf * ln(N/n)$

Table 2: This table lists some common normalizations. These functions were used in our meta-learning experiments. Here tf = number of times the term appears in the document, wc = total number of terms in document, N = total number of distinct terms in dictionary, IDF is the Inverse Document Frequency (a measure of how rare a term is in C), TF is the frequency of the most frequently occurring term in the given document, and n = number of times that term occurs in collection C.

feature vector	truth
x_1	y_1
$\vdots$	$\vdots$
x_k	y_k

Table 3: A learning set.

or model f

$$y = f(x) = f_a(x) = f(x; a), \qquad a \in A,$$

where $f = f_a$ is defined by specifying parameters $a \in A$ from an explicitly parameterized family of models A [6]. There are well known parameterizations for a wide variety of statistical models, including linear regression, logistic regression, polynomial regression, generalized linear models, regression trees, classification trees, and neural networks.

Fix a model $f = f_a$. The model is applied to previously unseen feature vectors x (that is, feature vectors outside of the learning set) to predict the truth $y = f(x)$. We measure success by using a test set distinct from the learning set and measuring the deviation of $f(v)$ from t over the test set.

Meta-learning. Broadly speaking, learning is concerned with finding a model $f = f_a$ from a single learning set, while meta-learning is concerned

feature vector	predicted truth	truth
v_1	$f(v_1)$	t_1
$\vdots$	$\vdots$	$\vdots$
v_j	$f(v_j)$	t_j

Table 4: A test set.

with finding a meta-model $f = f_a$ from several learning sets $\{L_1, \ldots, L_n\}$, each of which has an associated model $f = f_a[j]$. The n component models derived from the n learning sets may be of the same or different types. Similarly, the meta-model may be of a different type than some or all of the component models. Also, the meta-model may use data from a (meta-) learning set, which is distinct from the data in the individual learning sets L_j.

We begin by describing two simple examples of meta-learning. For the first example [5], we are given a large data set L and partition it into n disjoint subsets $\{L_1, \ldots, L_n\}$. Assume that we build a separate classifier on each subset independently to produce n classifiers $\{f_1, \ldots, f_n\}$. For simplicity, assume that the classifiers are binary so that each classifier takes a feature vector and produces a classification in $\{0, 1\}$. We can then produce a meta-model simply by using a majority vote of the n classifiers.

As the second example, given a learning set L, we replicate it n times to produce n learning sets $\{L_1, \ldots, L_n\}$ and create a different model f_j on each learning set L_j, e.g., by training the replicated data on n different model types. Given a feature vector x, we can produce n scores $(f_1(x), \ldots, f_n(x))$, one for each model. Given a new learning set ML, we can build a meta-classifier f on ML using the data $\{(f_1(x), \ldots, f_n(x), y) : (x, y) \text{ in } ML\}$.

In the work below, the fj are different information retrieval algorithms trained on the same learning set L as in the second example. We combine them using a polynomial model or other basic model. The combining model uses the scores produced by the base algorithms and not any document or query vector features directly.

5 Implementation

In this section, we briefly describe the system we used for these experiments. The experiments were done using a text mining module added to the PATTERN [12] data mining system.

PATTERN is designed around ensemble based learning. That is, the basic structure in the system is an ensemble of models. Ensembles in PATTERN naturally arise by partitioning the learning set, by sampling the learning set, and by combining multiple classifiers. For the experiments described below, we simply viewed the outputs of experiments involving different IR algorithms as a meta-learning set. We then used PATTERN to compute a (meta-) classifier.

The PATTERN text mining module creates a dictionary and document-term matrix by scanning a collection of documents. The dictionary can be composed of either words or n-grams. In either case, stop lists can be applied to remove common terms. In the case of words, a stemming algorithm can also be applied.

PATTERN's document-term matrix and feature vectors use sparse data structures, since several of the experiments used over 150,000 terms. This size of the feature vectors was a problem for PATTERN initially and some effort was required for PATTERN to work with feature vectors of this size.

6 Experimental Results

We created a learning set using FBIS documents from the TREC [1] collection and estimated the parameters for four different combining functions - linear regression, two different quadratic regressions, and cubic regression.

For the validation, we used 300 FBIS documents and 29 queries. This produced 8700 query document combinations. One hundred and ten of these combinations were classified as relevant in the TREC data set, i.e., in 110 of the 8700 query-document combinations, the query was labeled as relevant to the corresponding document. For each document-query pair, we computed 40 = 8 x 5 scores, by using the 5 similarity metrics and the 8 term weight normalizations from the Tables 1 and 2 below. These 40 scores were the independent variables used for each combining function.

To simplify the interpretation of our experiments, we selected one of the best performing IR algorithms as a base line. The second column represents the number of documents required by the baseline algorithm in order to obtain the number of relevant documents given in the first column. The remaining columns give the number of the documents required by the different combining methods (linear regression, cubic regression, and two different quadratic regressions) in order to obtain the number of relevant documents indicated. The 40 scores generated for each query-document pair were combined using each of the four combining methods to obtain a single combined

No. Relevant	Base Line	Linear	Cubic	Quad. 1	Quad. 2
10	592	380	145	75	76
20	1352	931	299	212	195
30	1825	1417	820	400	372
40	2453	1820	1122	670	622
50	4013	2306	1422	933	920
55	4383	2442	1598	1168	1137
60	5044	2601	1730	1458	1345
70	5817	3136	2101	1910	1734
80	6733	4164	2775	2668	2312
90	7420	4880	3472	3645	3035
100	8328	5552	5211	4875	5353
110	8644	7597	8101	7780	7948

Table 5: This table contains the results of applying four different meta-learning algorithms to the forty algorithms obtained by combining each of the similarity functions with each of the normalization functions. The best performing single algorithm was used as the base line.

score for the given pair. The query-document pairs were ordered by these combined scores.

As can be seen from this Table 3, the various combining functions significantly outperformed the baseline. Note that the baseline did not perform particularly well on this data set. Only very simple term weighting and similarity metrics were employed in this proof of concept, including some that are found in the literature, but are rarely used today. This points out that combining even relatively weak IR algorithms using meta-learning can produce an IR algorithm which is strong enough to be useful.

Note that by looking at the first row in the table above, we see that the recall for the base line algorithm is $10/110 = 0.090$ and the precision is $10/592 = 0.016$. From the same row, we see for the quadratic combining function and the same recall, the precision is $10/75 = 0.133$. That is, for the same recall, the nonlinear quadratic combining function increases the precision by over 700

7 Further Work

The work described in the body of this paper has provided encouragement for pursuing and greatly expanding the meta-learning approach to IR. Below, we briefly describe the directions and successes that this expanded effort has taken. A further paper, to be published in the near future, will describe this expanded effort in much greater detail.

The study was expanded from the preliminary 300 to 11,188 FBIS documents, and 10 queries, chosen to have reasonable, but not excessive representation in the document set. The number of term weighting schemes and document/query similarity metrics was expanded, resulting in 54 possible IR algorithms; then, algorithms that made no sense or were too closely correlated to earlier algorithms were eliminated, resulting in 28 IR algorithms, ranging from state-of-the-art to found-only-in-textbooks to homegrown.

The combining functions described in the body of this paper were replaced by meta-learning functions drawn from the Machine Learning (ML) world, e.g., decision trees, logistic regression, and later Random Forests (RF), a novel tree-ensemble method developed at Berkeley. The results using these ML meta-models confirmed the results reported in this preliminary study. Retrieval using a meta-model trained on a variety of simple IR models significantly outperforms any of the individual IR models taken separately. We can express these results in slightly different terminology: The collection of IR algorithms combined by the meta-model is an ensemble. The individual IR models are base models. We found that an ensemble could consistently outperform a varied collection of simple base models. Interestingly, the base models that contributed most to the performance of the ensemble were not necessarily the models developed from the algorithms most recommended in the IR community.

8 Summary and Conclusion

To summarize, in this paper, we have introduced the basic idea of using meta-learning to combine several different information retrieval algorithms to improve precision and recall. The idea of meta-learning is simple: a family of IR algorithms applied to a corpus of documents in which relevance is known produces a learning set. A machine learning algorithm applied to this set produces a classifier which combines the different IR algorithms.

We have also presented evidence of the effectiveness of this approach when simple nonlinear combining models are used on a collection of IR algorithms produced by varying the similarity and normalizations.

The experimental study described here is promising, but additional work is required to understand the applicability of this approach on larger and more heterogeneous data sets and using a wider variety of IR algorithms.

Acknowledgements

This work was supported in part by the CROSSCUT Program from DOD and in part by Magnify.

References

[1] *Proceedings of the Third Text Retrieval Conference (TREC-3).* NIST Special Publication 500-226, 2000.

[2] Thomas G. Dietterich. Machine-learning research: Four current directions. *AI Magazine*, 18(4):97–136, 1997.

[3] E. A. Fox and J. A. Shaw. Combination of multiple sources. *The Second Text Retrieval Conference (TREC-2)*, pages 97–136, 1994.

[4] Ed Greengrass. Information retrieval: A survey. *DOD Technical Report TR-R52-008-001*, 2001.

[5] R. L. Grossman, H. Bodek, D. Northcutt, and H. V. Poor. Data mining and tree-based optimization. pages 323–326, Menlo Park, California, 1996. AAAI Press.

[6] Robert L. Grossman and Richard G. Larson. A state space realization theorem for data mining. page submitted for publication, 2002.

[7] D. A. Hull, J.O. Pedersen, and Schutze. Method combination for document filtering. 1996.

[8] J. H. Lee. Combining multiple evidence from different properties of weighting schemes. *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1995.

[9] J. H. Lee. Analyses of multiple evidence combination. *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1997.

[10] Mayfield. Personal communication.

[11] James Mayfield, Paul McNamee, and Christine Piatko. The jhu/apl haircut system at trec-8. *NIST Special Publication*, 2000.

[12] PATTERN. The pattern system version 2.6, magnify, inc.

[13] A.L. Prodromidis, P. K Chan, and S. J. Stolfo. Meta-learning in distributed data mining systems, issues and approaches. In *Advances in Distributed Data Mining, edited by Hillol Kargupta and Philip Chan, MIT*, pages 81–113. MIT Press, 2000.

[14] Gerald Salton. *Automatic Text Processing: The transformation, analysis, and retrieval of information by computer.* Addison-Wesley, Reading, MA, 1989.

[15] C. J. van Rijsbergen. *Information Retrieval 2nd edition.* Butterworths, London, 1979.

[16] C. C. Vogt and G. W. Cottrell. Predicting the performance of linearly combined IR systems. pages 190–196, 1998.