Pheromonic Representation of User Quests by Digital Structures

Endre Boros, Paul B. Kantor, Dave J. Neu
Rutgers University
4 Huntington St, New Brunswick, NJ 08903
kantor@scils.rutgers.edu http://aplab.rutgers.edu/ant/

Abstract

In a novel approach to information finding in networked environments, each user's specific purpose or "quest" can be represented in numerous ways. The most familiar is a list of keywords, or a natural language sentence or paragraph. More effective is an extended text that has been judged as to relevance. This forms the basis of relevance feedback, as it is used in information retrieval. In the "Ant World" project (Ant World, 1999; Kantor et al., 1999b; Kantor et al., 1999a), the items to be retrieved are not documents, but rather quests, represented by entire collections of judged documents. In order to save space and time we have developed methods for representing these complex entities in a short string of about 1,000 bytes, which we call a "Digital Information Pheromone" (DIP). The principles for determining the DIP for a given quest, and for matching DIPs to each other are presented. The effectiveness of this scheme is explored with some applications to the large judged collections of TREC documents.

Vector space model

The vector space model represents a document as a bag of words, in which the order of occurrence is ignored, but the frequency with which a word appears in the text is considered (Salton, 1971). We use this approach as the starting point for our analysis.

Let us denote by $f_t(d)$ the relative frequency of term t in document d. We shall also use f_t as a generic operator, denoting always the the relative frequency of term t in the document to which it will be applied.

Documents are represented by the vectors $F(d) = (f_t(d) \mid t \in \mathcal{T})$, where $\mathcal{T}$ denotes a collection of terms, considered in the particular dataset. Usually $\mathcal{T}$ will contain all non-stop words that occur in the training documents.

Logical rules describing relevance

We shall consider *logical rules* to describe relevance with respect to a specified topic or quest. For instance, if someone is looking for information on pets, the simple rule

$$\textbf{If} \quad ((f_{cat} \geq 0.1 \textbf{ or } f_{dog} \geq 0.1) \textbf{ and } f_{wild} < 0.1) \textbf{ then RELEVANT}$$

could be applied to obtain documents about non-wild cats and dogs. Of course, the full scope of relevance cannot be grasped with one simple rule; one may need several rules for describing relevance, as well irrelevance.

Since all such rules can be represented as disjunctions (*or*) of conjunctions (*and*), we shall concentrate on the tasks of learning "good" conjunctive rules from a training set of documents, and combining them in such a way that the resulted classification will make only a few errors.

C4.5

One possible way of finding good logical rules is essentially mimicking the popular word-game in which one has to determine a hidden subject by a series of yes-or-no questions. By asking questions, one can at each step divide the large family of possible subjects into two subfamilies, and one can achieve the results the fastest (in the sense of having to ask the smallest number of questions) if this division is as even as possible at each step.

C4.5 is an algorithm developed by Quinlan (Quinlan, 1993), which asks questions of the form *is the term t occurring more than c_t times?* and tries to identify the term t and the threshold frequency c_t in such a way that either possible answer yields the most information on the training set. To achieve this, this method proposes various information theoretic measures, and applies them in a sequential fashion, with no backtracking.

As a result, one obtains a so called *binary decision tree*, the nodes of which represent subsets of the training documents for which a particular logical statement is true. An example is included in Figure 1, for a topic about *cats as pets*. The tree in this example represents three rules: one for relevance and two for irrelevance. The rule

$$\textbf{if } f_{cat} \geq 0.1 \textbf{ and } f_{wild} < 0.15 \textbf{ then RELEVANT}$$

corresponds to the second leaf node, while the rules

$$\textbf{if } f_{cat} \geq 0.1 \textbf{ and } f_{wild} \geq 0.15 \textbf{ then IRRELEVANT}$$

and

$$\textbf{if } f_{cat} < 0.1 \textbf{ then IRRELEVANT}$$

correspond to the first and third leaf nodes, representing documents in which either there are no *cat*s mentioned with a high frequency, or both *cat*s and *wild* occur frequently.

Given a set of training documents with known classification of their relevance, C4.5 recursively searches for a term t and a threshold c_t for which the branching, i.e. splitting the documents into two groups according to the truth value of the logical proposition $f_t \geq c_t$, improves the most on the homogeneity, with respect to relevance, of these groups. At the end, all groups of documents corresponding to the leaves of the tree are homogeneous, i.e. each contains either only relevant, or only irrelevant documents. In practice, this may result in a large number of branches, with very small groups of documents, i.e. generating rules that apply only to small subsets of the training documents. To prevent this, C4.5 stops earlier, even if perfect homogeneity of the leaf groups is not achieved yet. Rules generated in this way are not perfect, in the sense that they may make (a small number of) mistakes on the training set, but each of them is "active" on a larger number of training documents.

For each generated rule R, C4.5 reports a measure of "correctness" $c(R) \in [0, 1]$. In our experiments we use the sum $S(d)$ of the $c(R)$ values of those rules which are active for document d, as the *score* of d.

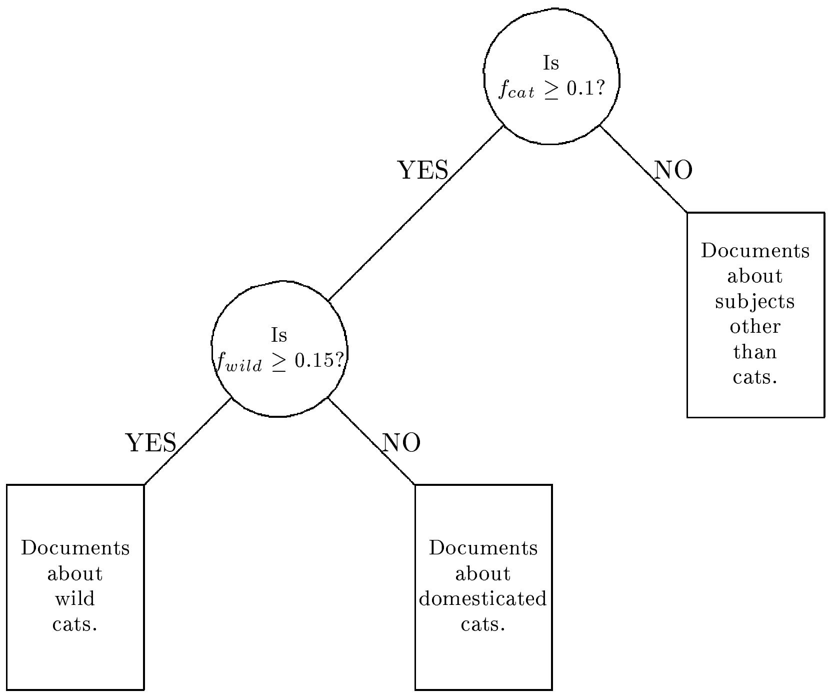
Is
$f_{cat} \geq 0.1$?
YES
NO
Documents about subjects other than cats.
Is
$f_{wild} \geq 0.15$?
YES
NO
Documents about wild cats.
Documents about domesticated cats.

Figure 1: A binary decision tree

Logical Analysis of Data

LAD is another technique to find logical rules characterizing the difference between relevant and irrelevant documents, based on a set of training documents (Crama et al., 1988; Boros et al., 1999).

The distinguishing feature of LAD is that it searches in an exhaustive manner for all highly relevant terms, and then separately for all reasonable rules formulated with those terms, and applies optimization models to select the best ones.

In a typical problem arising from the TREC collection (NIST, 1999), the training set consists of a few hundred relevant and irrelevant documents, containing about 12,000–15,000 different terms.

In the first phase of LAD a much smaller subset of terms $\mathcal{S}$ is selected based on the fact that the frequency vectors $(f_t(d) \mid t \in \mathcal{S})$ expressed in this basis are significantly different for relevant training documents as opposed to irrelevant ones. For each of these terms a "best" threshold value c_t, $t \in \mathcal{S}$ is generated, and a document d will be represented in the following analysis by the binary vector $X(d) = (X_t(d)|t \in \mathcal{S})$, where X_t is defined by

$$X_t = 1 \textbf{ if and only if } f_t(d) \geq c_t$$

for $t \in \mathcal{S}$. To decide whether relevant documents are "significantly" different from irrelevant ones, we use the minimum Hamming distance between the binary vectors $X(d)$ corresponding to relevant and irrelevant training documents. The selection of these terms is based on a set covering type optimization model, in which the size $|\mathcal{S}|$ of the set $\mathcal{S}$ of terms is minimized, subject to the constraints that the Hamming distance

$$\rho(d,d') = \sum_{\substack{t \in \mathcal{S} \\ X_t(d) \neq X_t(d')}} 1$$

between relevant documents d and irrelevant documents d' in the training collection is large enough. We call the set $\mathcal{S}$ of terms a *support* set, since these terms are supporting the claim that relevant documents are different from irrelevant ones.

In a typical easy example (where most of the learning algorithms considered were able to learn well from the training set) the size $|\mathcal{S}|$ of the resulting support set was 3–5 times the required minimum Hamming distance, e.g. requiring a minimum Hamming distance of 10 resulted in a smallest support set of size between 30 and 50. Larger ratios typically occurred for more difficult training sets. This implies that even when we demand a minimum Hamming distance of 20 or 30, for a clear, "noiseless" separation between relevant and irrelevant training documents, we had to consider only 60–150 different terms for the "good" training sets (and a somewhat larger number for the others.)

In the second phase of LAD, we consider the training documents represented by the reasonable short binary vectors $X(d) = (X_t(d) | t \in \mathcal{S})$, where $\mathcal{S}$ is a support set chosen in the previous phase. Simple rules in the form of elementary conjunctions are considered, e.g.

$$R = X_a \wedge \overline{X}_b \wedge X_c$$

where $\overline{X}$ denotes the logical complement (negation) of X. For each of these rules we associate the numbers $C^+(R)$ ($C^-(R)$) denoting the number of those relevant (irrelevant) training documents for which R is active or "fires". We call a rule R a *positive pattern* if $C^+(R) > 0$ and $C^-(R) = 0$, and we call it a *negative pattern* if $C^+(R) = 0$ and $C^-(R) > 0$. In other words, positive patterns "fire" for some relevant documents and not for any irrelevant documents.

The number of propositions X_t in the definition of the rule R is called the *degree* of R, while the non-zero number $C^+(R)$ for positive patterns and $C^-(R)$ for negative patterns is called the *coverage* of R.

The second phase of LAD has several steps. First, all patterns (with degree lower than some preset limit) are generated, subject to the requirement that the coverage of each pattern be "large enough". We remark that, in the worst case, the number of such patterns can be huge (unless we restrict the degree to be quite small!) However, as recent results show (Boros and Yagiura, 1999; Purdom, 1999), the expected number of acceptable patterns in randomly generated input data is limited, regardless of any degree restriction, and moreover such patterns can be generated in polynomial expected time.

Next, from this typically large set of "good looking" rules, we select a smallest subset $\mathcal{R}$ of rules, which still can explain the differences between relevant and irrelevant documents. This step is formulated again as a set covering type optimization problem.

For a subset $\mathcal{R}$ of the possible rules and for every pair d and d' of relevant and irrelevant documents we associate a *separation* measure

$$\sigma(d,d') = \sum_{\substack{R \in \mathcal{R} \\ R(d) \neq R(d')}} 1$$

and we minimize the size $|\mathcal{R}|$ of the selected rule-set subject to the constraints that $\sigma(d,d')$ is never smaller than some preset positive parameter. We call the resulting smallest set $\mathcal{R}$ of rules a *theory.*

Given such a theory $\mathcal{R}$, we associate to any document d a score $S(d)$ by defining

$$S(d) = \sum_{\substack{R \in \mathcal{R} \\ R(d)=1 \\ R \text{ is a positive pattern}}} C^+(R) - \sum_{\substack{R \in \mathcal{R} \\ R(d)=1 \\ R \text{ is a negative pattern}}} C^-(R).$$

This is the score we used to produce the ranked lists of *retrieved documents* in evaluating the rules for both training and test sets in our experiments.

Experiments

We have selected the first 20 topics from the TREC7 competition, and carried out a large number of experiments on those. Some of these experiments were also carried out for several other topics, however many of those other topics are "extreme" in the sense that the number of relevant training documents is very small (i.e. there is almost nothing to learn from), and the different learning algorithms were not able to handle such extreme cases in consistent ways.

Another limiting factor in our computations is due to the C4.5 procedures themselves. In the current implementation of C4.5, the very high number of potential terms (typically in the range of 12,000 – 15,000) was just impossible to handle. We had to choose specific ways of providing a small, but meaningful subset of the terms, which then were used in C4.5. We explored three different approaches: the **terms appearing in the topic description** seemed to be a natural choice, and proved to be the best one. We also tried a **corpus-driven** approach, considering only those terms which appear at least in 3% of the training documents. Even though this choice seems very reasonable, the computational results are very poor. As a third option, we also considered an **LAD-C4.5 hybrid**, in which we generate a small support set by LAD, and then use C4.5 to develop the best rules using these terms. This option proved to be reasonably good, though slightly worse than the topic term set (perhaps due to the fact that LAD generated, in these examples, a smaller set of terms than those appearing in the topic description.)

The test set in TREC7 consists of about 160,000 documents, and for each topic there are only a few hundred documents with a known relevance classification (*judged documents*). Using the indicated scoring functions, with each variant of the algorithms we scored all 160,000 documents, considered the top 1000, and computed the average score p_{ave} as used at TREC: Let N denote the number of judged relevant documents for a particular topic, let $\{d_1, d_2, ..., d_{1000}\}$ denote the ranked list of the top 1000 documents produced by the considered algorithm, and let g_i denote the number of judged relevant documents among the first i documents in this list. Then p_{ave} is defined by

$$p_{ave} = \frac{1}{N} \sum_{\substack{i=1 \\ d_i \text{ is judged relevant}}}^{1000} \frac{g_i}{i}.$$

This corresponds to the measure called *exact averaged precision versus recall.* In the following graphs we compare the obtained p_{ave} scores to the median scores for all TREC7 submissions. In these graphs, each topic is represented by a single point, and the value of p_{ave} for each of two is used to position it in the graph. Thus, if most of the points lie below the diagonal, the system corresponding to the horizontal axis has better performance for most topics.

In the first two runs, Ant Route 1 and 2, we used LAD, setting the minimum required Hamming distance in the support set generation to 35 and in the theory generation to 15. In Ant Route 1 we used 100 relevant judeged and irrelevant judged training documents, while in Ant Route 2 we used up to 300 relevant judged and up to 500 irrelevant judged training documents (see Figures 2 and 3.)

In all of the other LAD runs we have used a minimum Hamming distance of 15 in the support set generation step. As a preliminary filtering, we considered only terms which appeared in more than 5% of the training documents. Furthermore, in the exhaustive pattern generation step we generated all patterns having a coverage of at least 10% of the training documents. Finally, in the theory generation step we required a minimum Hamming distance of 5.

Since we observed that C4.5 produced quite weak results when used alone (see Figure 4), we also considered a hybrid strategy that we call the *two-phase method*.

In the first phase, for each topic we used LAD to generate a logical formula explaining the differences between judged and unjudged documents. In the test set and in the training set, the latter are a clear majority. Using this logic, we scored all documents, and choose the top 8100 (for technical reasons). This is an approximation to the set of judged documents.

Then, in the second phase, we used C4.5, or LAD, or a hybrid of these, to develop a logical explanation distinguishing between judged relevant and judged irrelevant documents, and used this logic to score out only the 8100 test documents preselected in the first phase. Then again we chose the top 1000, and computed the p_{ave} average, as before (see Figures 5, 6 and 7.)

Finally, in Figure 8 we compare the best LAD results (one phase LAD run of Ant Route 2) with the best C4.5 results (hybrid run, using topic terms and tested only on the Phase I results produced by LAD.)

Conclusions

Overall, the result of this preliminary analysis show that the TREC routing task is a very challenging machine learning task. None of the schemes involving C4.5 performed better than a pure LAD scheme (AntRout2) that was our official submission to the TREC7 competition. That specific run, in turn, did not fare well in comparison to the median of TREC participants. Thus there is a great deal to be learned about this specific problem. We are encouraged by the fact that the two-phase method does somewhat better than a one-phase method. This may be an artifact, tracing to the fact that documents judged at TREC form some specific subclass of all documents, defined by an unwritten consensus regarding retrieval methods. However, in a real application, such as learning from the actions and judgments of an individual searching the WWW, the two-phase approach may be quite appropriate, as one theory may be needed to explain whether the user even chooses to judge a document, and another may be needed to capture the relevance feedback information represented by the user's judgments.

Acknowledgments

This work is supported by the Defense Advanced Research Projects Agency (DARPA) Contract Number N66001-97-C-8537.

References

Ant World (1999). The Ant World web site, `http://aplab.rutgers.edu/ant/`.

Boros, E., P.L. Hammer, P., Ibaraki, T., Kogan, A., Mayoraz, E., and Muchnik, I. (1999). An implementation of logical analysis of data. *IEEE Transactions on Knowledge and Data Engineering*. In print.

Boros, E. and Yagiura, M. (1999). Generating strong patterns in expected polynomial time. Manuscript.

Crama, Y., Hammer, P., and Ibaraki, T. (1988). Cause-effect relationships and partially defined boolean functions. *Annals of Operations Research*, 16:299–326.

Kantor, P. B., Boros, E., Melamed, B., and Meñkov, V. (1999a). The information quest: A dynamic model of user's information needs. In *The 1999 ASIS Annual Meeting*. To appear.

Kantor, P. B., Melamed, B., Boros, E., Meñkov, V., Neu, D. J., Kim, M.-H., and Shi, Q. (1999b). Ant World. In *SIGIR'99 Proceedings*. To appear.

NIST (1999). The TREC web site, `http://trec.nist.gov/`.

Purdom, P. (1999). Average case analysis of the apriori algorithm. Conference talk, SA23, *INFORMS*, Cincinnati, May 2-5, 1999.

Quinlan, J. (1993). *C4.5: Programs for machine learning*. Morgan Kaufmann.

Salton, G. (1971). *The SMART retrieval system; experiments in automatic document processing*. Prentice-Hall, Englewood Cliffs, N.J.

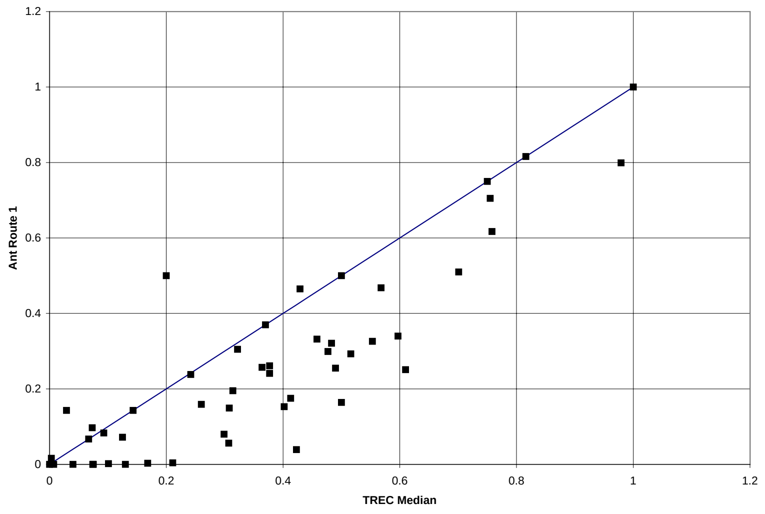

Figure 2: Ant Route 1 vs. TREC Median

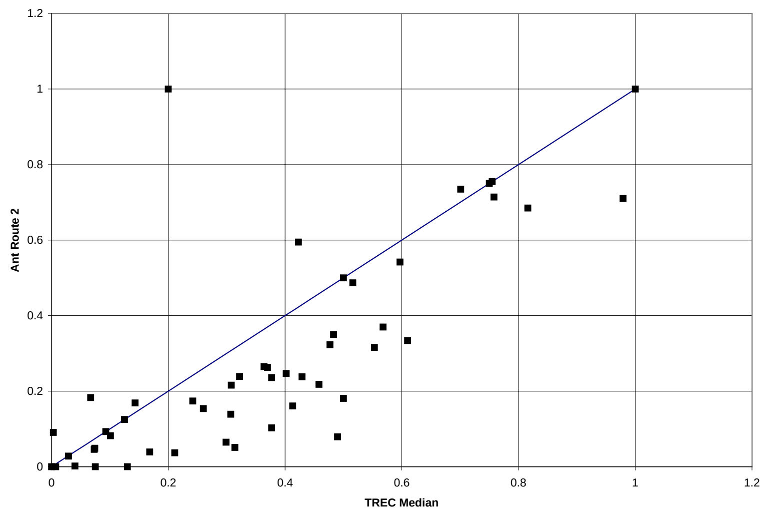

Figure 3: Ant Route 2 vs. TREC Median

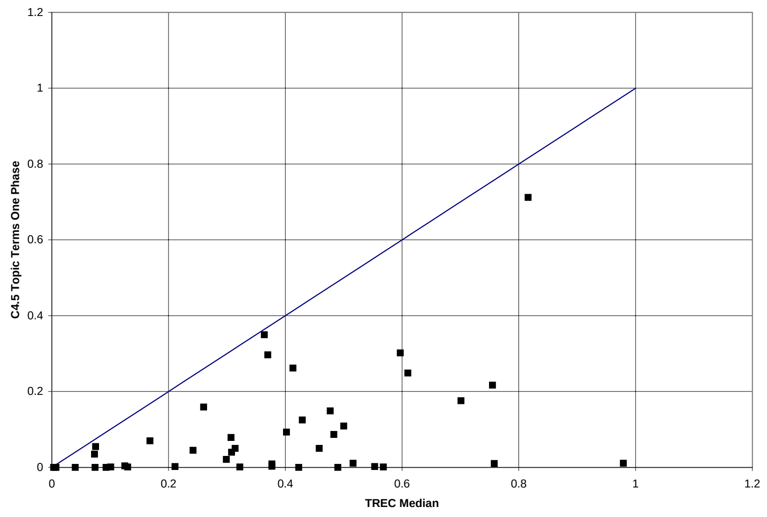

Figure 4: C4.5 with topic terms, tested on entire test set vs. TREC Median

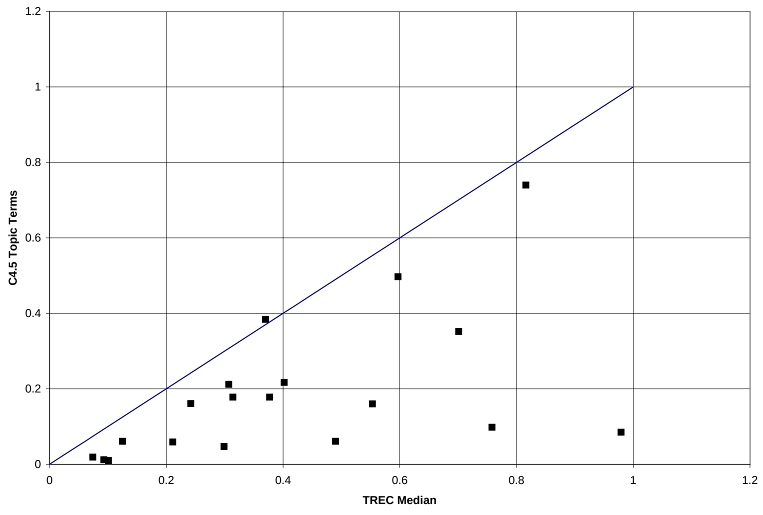

Figure 5: C4.5 with topic terms, tested on Phase I results vs. TREC Median

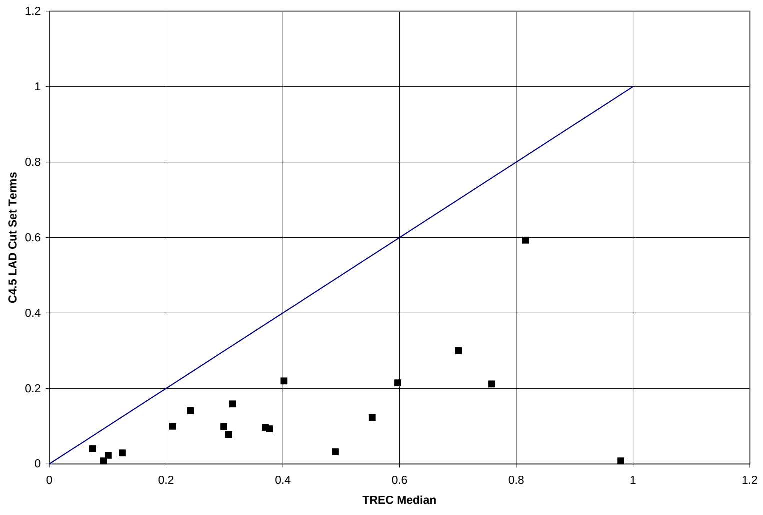

Figure 6: C4.5 with LAD cut set, tested on Phase I results vs. TREC Median

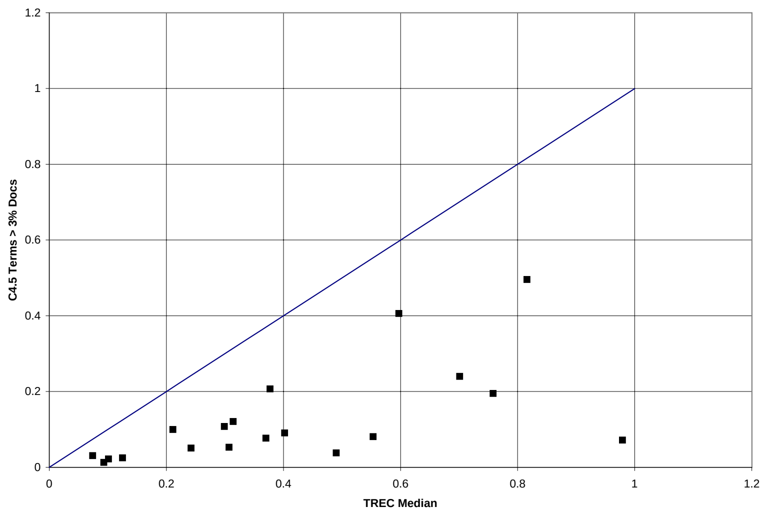

Figure 7: C4.5 with terms appearing in more than 3% of training documents, tested on Phase I results vs. TREC Median

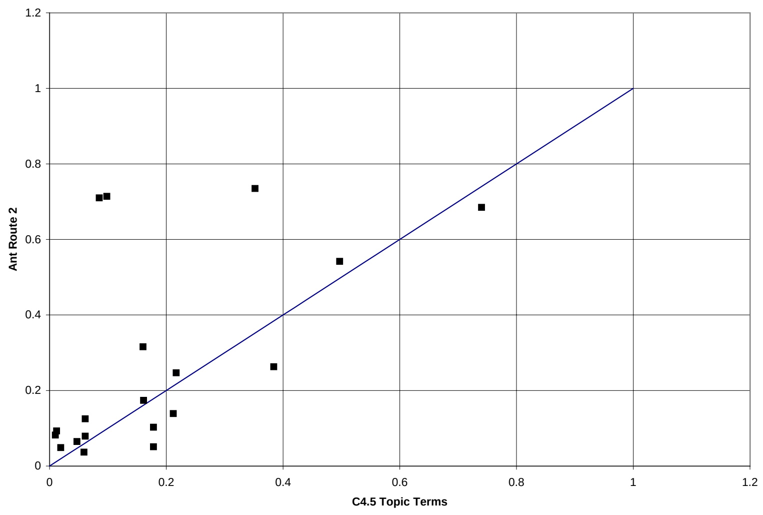

Figure 8: Ant Route 2 vs. C4.5 with topic terms (tested on 8100 documents selected by LAD in Phase I (2-phase model).)