RC22314 (W0201-105) January 23, 2002
Computer Science

IBM Research Report

In Question-Answering, Hit-List Size Matters

John M. Prager
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

In Question Answering, Hit-List Size Matters

John M. Prager
IBM T.J. Watson Research Ctr.
P.O. Box 704, Yorktown Heights NY 10598
jprager@us.ibm.com

Abstract

We look at the effect of hitlist size in a Question-Answering system. When the goal is to find a single answer to a fact-seeking question, it becomes readily apparent that looking at too many documents can be a source of more noise than useful information; looking at too few documents can miss the desired answer.

We develop a probabilistic model of the Answer-Selection component that extracts candidate answers from passages returned by the search engine. We show with this model: how we can predict performance as a function of hitlist size after making observations of the system's operation using a single such size; that an optimum hitlist size exists; how the performance depends on a parameter we call the discrimination ratio, and the benefit to be derived from improving this ratio by more training. We show that the performance curves generated by our model agree very well with empirical results.

1. Introduction

The field of Question-Answering (QA) falls squarely in the intersection of Information Retrieval (IR) and Natural Language Processing (NLP). Most papers in the field deal with issues in linguistics, ontologies or AI (see e.g. Moldovan et al, 2000; Srihari and Li, 2000; Hovy et al. 2001; Prager et al. 2001b; Chaudri & Fikes, 1999), but relatively few with IR; this may appear surprising since every QA system uses a search engine (at least, those that participate in TREC [TREC8, TREC9, TREC10]). NLP is so computationally expensive it is inconceivable (currently) to perform linguistic analysis on an entire large corpus at run time. The TREC corpus, for example, consists of about a million documents, or 3GB of data (see [Voorhees and Tice, 2000]). The "typical" QA system is built around a pipeline which consists of a question-analysis component to massage the question into a query suitable for search, a search engine to return documents or passages for further consideration, and an Answer Selection component to extract and rank candidate answers from these passages (see Figure 1). There is increasing interest in inserting loops around this pipeline to return to earlier stages for revised processing (such as query expansion) if some decision criterion fails to be met, but this doesn't change the basic architecture (see e.g. [Harabagiu et al., 2000]).

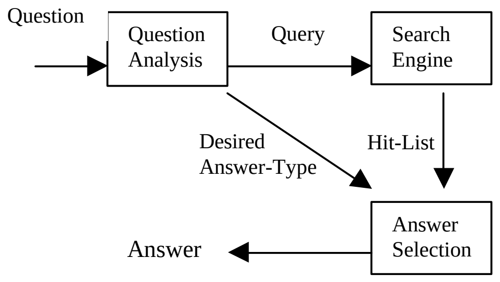

Figure 1. Generic Question-Answering pipeline.

The Answer Selection component (AS) takes the passage hit-list returned by the search engine and looks in the text for answers (typically named entities) that are called for by the question. The analysis performed here can range from very little to question-word density measurements to predicate-argument structure analysis to logical inference. In all cases it is true that the more passages the AS sees the more opportunity it has to find the right answer. By the same token, though, the more passages it sees the more noise there is to obscure the right answers. This is exactly the precision-recall tradeoff that permeates IR.

However, there is a significant difference in how this tradeoff manifests, due to the different goals of IR and QA. To put it simply, QA systems are presented with a natural-language question (currently almost always seeking facts) and are considered to succeed if they find a single correct answer to the question. On the other hand, IR systems are presented an information need in the form of a query, and they attempt to find as many documents as possible that are relevant to this need. There is no concept of an exact answer, or "enough" of an answer, in this articulation of IR. Because of this, it is difficult to come up with a single satisfactory measure of performance for IR systems. Often, a precision-recall curve is drawn, allowing a late binding readout of the system's performance characteristics based on the user's particular precision or recall needs. At other times, average precision is computed over a range of recall values. In none of these measures, though, is there the notion of the "optimal" number of documents

documents to return. This is because in this definition of the IR task there is no subsequent analysis of the returned documents for which a cost-benefit analysis of providing extra documents can be performed.

With QA, though, there is a clear measure of success: the right answer is found from analysis of the output of the search engine. It is reasonable to hypothesize that there is an optimum number of documents or passages for the Answer-Selection module to examine. Such a number may well vary considerably from system to system, or within a system, from corpus to corpus or question-set to question-set.

The system of Predictive Annotation [Prager et al. 2000] indexes semantic categories as well as words, and includes in the bag-of-words queries the semantic category called for by the question (e.g. PERSON, COUNTRY etc.). This guarantees that the returned passages will contain candidate answers of the right type, so that the hit-list can be relatively small (e.g. 10-20 passages). Systems that delay search for the candidates of the right answer type until after the search engine returns must look at a much larger number of passages. Indeed, for the benefit of TREC participants who do not have their own search engine, NIST makes available the top 1,000 documents returned by a standard search engine.

We have previously examined the issue of hit-list size in QA from the point of view of how much incremental gain there is in looking at extra documents, but without paying any regard to the associated cost [Anon., 2001a]. In our post-TREC9 analysis, we examined, for a given hit-list size *N*, for how many of the 693 questions was there at least one document known to contain a correct answer in the hit-list. This determined an upper bound to the system's performance. We let *N* range from 1 to 50. We did this experiment using both our own search engine and the document sets provided by AT&T. We reproduce here in Figure 2 the resulting graphs.

We observe that, for both search engines, the number of questions with an answer document in the hit-list rises very quickly with hit-list size at first, and then levels off. There is a knee in the curves at about 8 documents, after which there is relatively little increase in answerable questions with hit-list size. These observations suggest that the payoff from increasing hit-list size may start going negative from this point onwards, due to encountering more noise than new answers, but such an assertion can not be proven from this data alone.

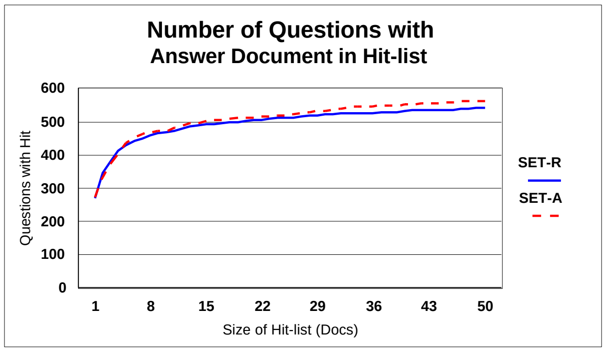

Figure 2. Comparison of number of questions with a "correct" document in the hit list against hit-list depth, for our search engine (SET-R) and AT&T's (SET-A).

We hypothesize that we can develop a reasonably accurate model of the Answer Selection process; this process takes as input the hit-lists produced by the search engine and produces in turn an output that can be evaluated. This system performance measure is accordingly a function of the hit-lists; all other parameters being fixed, the measure is a function of the hit-list size. We further hypothesize that the system performance is a concave function, guaranteeing the existence of an optimum hit-list size.

This paper is in 5 sections. In Section 2 we describe the experimental method we are adopting and the data-sets we study. In Section 3 we develop a probabilistic model of performance as a function of hit-list size. We use observations of actual experimental conditions to determine numerical parameters of the model. In Section 4 we use the model to generate predictions and compare them with experimental results, and in Section 5 we conclude and discuss future directions this work may take.

2. Experimental Method and Data

We will present in the next Section a theoretical model of hit-list performance, determine parameters empirically and use the model to predict optimal hit-list size. We will then compare this prediction with the actual performance of our system on the TREC QA task. In this Section we describe our experimental method and the data we use.

We have available the set of questions for each of TRECs 8, 9 and 10, the answer pattern sets and our system's performance figures. We are not stressing here the effectiveness of our QA system, for which a demonstration of training on TREC8 and TREC9, say, and testing on TREC10 would be appropriate. Rather, we are presenting the effectiveness of our model of how hit-list size affects performance. To do this, we deter-

determine some parameters of our system by observation and measurement, and then compare how the model configured with these parameters predicts performance *of the same system and data-set* as a function of hit-list size. We should also make clear that this is not the same as testing on training data: the training is in effect the development of the theoretical model, which is done without seeing the data. The developed model is a mapping of a small set of observed parameters to a complete predicted performance curve. The testing consists of inputting to the model these few observations and comparison of the predicted performance curves with those observed experimentally.

For the TREC9 set we only considered the first 500 questions (that is, we omitted the subset which were paraphrases of earlier questions). For all sets we omitted the questions that had no answer.

We perform the same exercise for all three of the TREC question sets. These sets are not equivalent tests of our system: the TREC8 questions were created by the community from the TREC corpus, while TREC9 and TREC10 questions were derived from actual query logs. Those logs were from different systems, and have different distributions of question types (especially definitional questions: 5% for TREC9 vs. 26% for TREC10) and as a whole appear to mirror actual user questions to different degrees (Ellen Voorhees, personal communication). The system parameters mentioned above appear not to be intrinsic to the system alone but to be a function of both the system and the characteristics of the problem set it is presented with. Thus we will derive different parameters from observation of running each of the question sets, which will in turn lead to different predictions of performance as a function of hit-list size.

We employ two different implementations of answer selection, both derivatives of that reported in [Anon. et al., 2000]. The first retains the scoring algorithm via linear regression, and so will be denoted LR. The second uses an entirely different scoring algorithm, via a decision tree, and is denoted DT. Combined with the three different question sets we have in all six experimental conditions; these will be denoted LR8, LR9, LR10, DT8, DT9 and DT10.

The runs reported in this paper were all performed with the same version of our system. The runs were made shortly after TREC10. Because (1) the questions processed were (for TRECs 9 and 10) a proper subset of the total, (2) the system was not exactly the same one that participated in TREC, and (3) for these runs our system produced an "exact answer", rather than the more liberal 50 bytes allowed up to now in TREC, the results reported here do not correspond to our official TREC results. Training of the system was performed on TREC9 data, so the performance there will be seen to be greater than on the TREC8 and 10 questions. We suspect that the decision tree was over-trained – this is exhibited by the greater differential between DT9 and DT8/10 than between LR9 and LR8/10.

3. The Model

In order to determine the optimum hit-list size (if it exists), we need to develop a probabilistic model of the process by which the passages returned by the search engine are examined and selected. For our purposes, we assume that there is a single answer (of the sought type) in the returned passage and so we can reduce the Answer Selection (AS) process to that of passage selection.

We will suppose that the search engine returns a hit list of candidate answer passages (which may be anything from a single term to a sentence or even a paragraph). Any number of these passages may contain a correct answer. We will define the AS process to be one which outputs the passage that it considers to be the most likely to be correct, based on criteria internal to it. We approximate its operation by saying that it categorizes the passages as correct or incorrect, and chooses one of the former set at random.

Since we will be modelling probabilities with mathematical functions, we will use a functional notation in developing this model[1]. Let H_N be the event that the hit-list size is N. Let X be the event that AS picks out a correct passage, and Y the event that it exists in the hit-list. Then we will use $P(N)$ to denote the probability $Pr(X|Y \& H_N)$, and $Q(N)$ to denote $Pr(Y| H_N)$. Let Z_k be event that the kth passage actually is correct; we will use $R(k)$ to denote $Pr(Z_k)$. Then the probability that at least one passage in the hit-list is correct is[2]

$$Q(N) = 1 - \prod_{k=1}^{N} (1 - R(k)) \qquad (1)$$

Let $S(N)$ denote $Pr(X| H_N)$. Then the probability that a correct passage is identified by AS is computed by[3]

$$S(N) = P(N)Q(N) \qquad (2)$$

We call the value $S(N)$ the *first-place score (FPS)*, to distinguish it from the *mean reciprocal rank (MRR)*,

[1] We used the Maple V product from Waterloo to perform the simulations.
[2] This assumes independence of the Z_i
[3] Assuming independence of X and there being at least one correct passage in the hitlist.

described below, up until 2001 used in TREC QA evaluation. In TRECs 8-10, QA systems submitted their top 5 answers for evaluation by MRR. Since it has been generally found to be true that QA systems get most of their correct answers in first place, next most frequently in second place, and so on, (as we will later see for our own system in Table 2), and since the MRR scoring scheme awards a non-linear score (*1/k*) to a first correct answer in *k*th place, we see that a large contribution to the MRR score is the FPS. We observe in practice that the FPS generally tracks the MRR quite well, although maybe about 20% below it. In the 2002 TREC evaluation, QA systems will submit only one answer, so the MRR and FPS will then be equivalent. We present in Table 1 the FPS and MRR scores for our system with hit-lists of size 10.

	TREC8	TREC9	TREC10
MRR on LR	.442	.470	.342
FPS on LR	.348	.388	.270
MRR on DT	.353	.577	.265
FPS on DT	.256	.535	.201

Table 1. System performance. The Mean Reciprocal Rank and First-Place Scores for a hit-list of size 10.

The optimum hit list size *N'* is given by

$$N' = \arg\max_N S(N)$$

Our hypothesis that a maximum to *S(N)* exists can be argued qualitatively as follows. If passage *k* is more likely to be correct than *j*, for *k<j*, then the probability that AS will pick out a correct passage, *P(N)*, will decrease with *N*, since a larger *N* means more noisy data to deal with. The probability that there exists some correct passage somewhere in the hit-list, *Q(N)*, is an increasing function of *N*. The initial slope of *Q* is relatively high, as the first few passages are considered, but will ultimately flatten out (cf. Figure 2). If *Q*'s initial slope is high enough (relative to *P*), then the product of *P* and *Q* will initially increase, but will peak and eventually decrease as *P*'s decline takes over.

We will now proceed to derive formulas for *P* and *Q* based both on theoretical arguments and empirically-derived parameters, and demonstrate that the theoretical model agrees quite well with our actual results.

Let *p* be the probability that when AS looks at a correct passage, it internally says that it is correct. Let *q* be the probability that it says that an incorrect passage is correct. Let us suppose that on average *m(N)* of the passages in the hit-list are correct. Then the probability that AS will output a correct passage is

$$P(N) = \frac{m(N)\,p}{m(N)\,p + (N - m(N))q}$$

$$= \frac{f(N)}{f(N) + \{1 - f(N)\}r} \qquad (3)$$

where *f(N) = m(N)/N* and *r = q/p*. *r* is an internal parameter of AS which we will attempt to estimate from observations of system performance[4]. To make the dependency of *P* on *r* clear, we will henceforth write *P(N, r)*.

We note that the quantity *f(N)* is the fraction of passages in the hit-list that are correct, and can be calculated by

$$f(N) = \frac{1}{N}\sum_{k=1}^{N} R(k) \qquad (4)$$

r measures how well AS is able to discriminate between correct and incorrect passages. We will call *r* the *discrimination ratio* of AS. Clearly, the lower *r* is, the better AS will perform.

NIST and individuals in the TREC community have made available pattern sets and scripts so that researchers can re-evaluate their systems as they improve them. We have used these pattern sets both to evaluate our system with different parameter settings, as described in the next section, and to evaluate intermediate results. In particular, we evaluated the output of the search engine (prior to Answer Selection) to see if these passages contained the correct answer, and if so where in the hit-lists these correct passages were distributed.

For all three TREC QA question sets, we counted how often the *k*th passage contained a correct answer. We examined hit-lists of size 50; the counts for the top 10 passages are presented in Table 2.

Times Passage correct	TREC8	TREC9	TREC10
Passage 1	87	219	136
Passage 2	81	159	117
Passage 3	64	145	103
Passage 4	63	140	104
Passage 5	51	117	99
Passage 6	51	123	92
Passage 7	46	124	78
Passage 8	38	108	81
Passage 9	41	94	73
Passage 10	39	95	59
Total questions	198	492	432

[4] If *a* is the answer selection process precision, then *r* = 1/*a* -1

Table 2. Passage accuracy. The number of times each passage in a hit-list of 10 passages contained the correct answer.

We note that the fractional performance represented by these counts is precisely the value *R(k)* from Equation (1), where *k* is the passage number. These values let us calculate *Q(N)* from Equation 1, *f(N)* from Equation 4 and *P(N)* from Equation 3. Then, by substituting the FPS scores from Table 1 into Equation 2, we can calculate *r* for each run. The calculated values are presented in Table 3.

Discrimination ratio *r*	TREC8	TREC9	TREC10
LR	.61 (.62)	.45 (.69)	.52 (.66)
DT	.98 (.51)	.24 (.81)	.79 (.56)

Table 3. The discrimination ratio *r* (and corresponding precision in parentheses) for our Answer Selection component calculated for each of the 3 TREC QA question sets and each version of Answer Selection.

We can now model the passage correctness scores in Table 3 and complete our probabilistic model. By inspection, these passage scores seemed to be following an exponential decline. Therefore a simple exponential curve $R(k) = Ae^{-Bk} + C$ was fitted to them (for passage number *k*). To make the fitting easier, we observe that as the passage number gets very large, the probability that the passage is correct tends to zero. Thus we can set *C=0*, take logarithms and perform the fitting by linear regression. The fitted curves have the equations:

$$\hat{R}_{TREC8}(k) = .28e^{-.026k}$$

$$\hat{R}_{TREC9}(k) = .28e^{-.029k}$$

$$\hat{R}_{TREC10}(k) = .24e^{-.026k}$$

The notation $\hat{R}$ is used to indicate the estimated value, as distinct from the measured value R. These functions model the search-engine hit-list passage correctness, and are independent of the AS algorithm.

4. Analysis of Model and Empirical Results

We are now in a position to calculate *S(N)* (from Equation (2)) for various values of *N*, and the fitted curves $\hat{R}$. Figure 3 shows in the solid lines the value of *S(N)* for LR8 for *N* ranging from 1 to 50, for the calculated parameter *r* = .61. For comparison we also show in dashed lines the family of curves generated by the same equation but with *r* ranging from .1 to 1.0 in steps of .1 (with the .1 curve at the top, the others lowering progressively). Figures 4 and 5 show the analogous curve sets for the LR9 and LR10 data.

The peak in Figure 3 suggests an optimum hit-list for TREC8 data of 7 or 8, while the peak in Figure 4 suggests an optimum hit-list size for TREC9 of 9 or 10: this accords very well with our intuition of using a hit-list of size 10, and as we will see shortly, it also accords with direct empirical data. We note in both cases that the right-hand tail is much flatter than the left, suggesting that if in doubt one should err on the side of making the hit-list too large. In any case, for our system, it should not be 5 or lower.

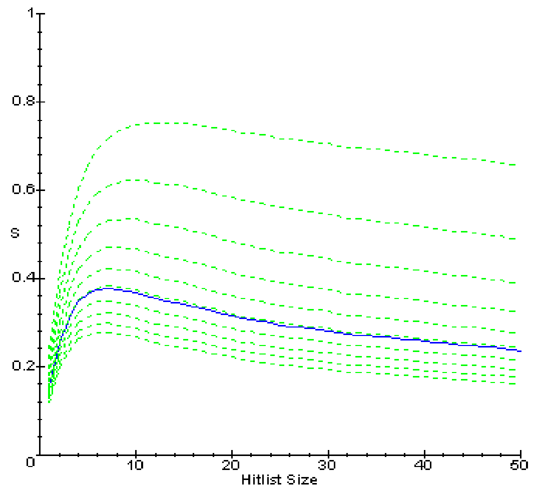

Figure 3. First-place scores *S(N)* against hit-list size *N*, as predicted by our model, for LR8. The dashed lines are, from top to bottom, with parameter *r* ranging from .1 to 1.0. The solid line is with *r*=.61. The peak is at *N* = 7.

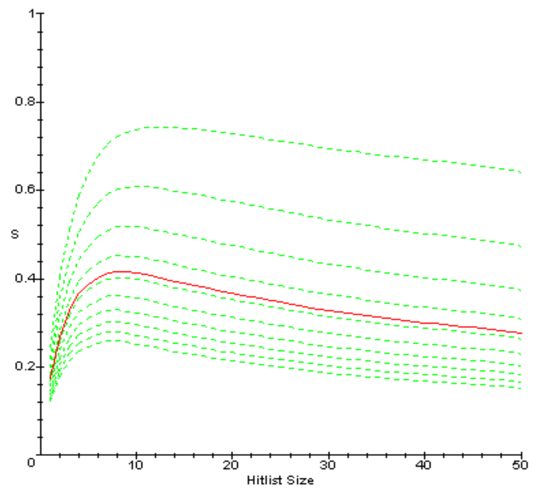

Figure 4. First-place scores *S(N)* against hit-list size *N*, as predicted by our model, for LR9. The dashed lines are, from top to bottom, with parameter *r* ranging from .1 to 1.0. The solid line is with *r*=.45. The peak is at *N* = 9.5.

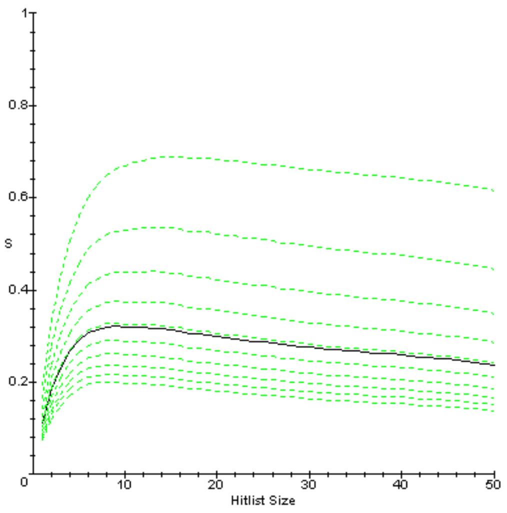

Figure 5. First-place scores *S(N)* against hit-list size *N*, as predicted by our model, for LR10. The dashed lines are, from top to bottom, with parameter *r* ranging from .1 to 1.0. The solid line is with *r*=.52. The peak is at around 10.

The peak for TREC10 data (Figure 5) is quite flat and essentially ranges from 8 to 14, but as we shall see, this too agrees with experiment. For reasons of space, the corresponding figures for DT are omitted, but the peak positions are summarized later in Table 4; the curves themselves are similar-looking to those for LR.

The dashed curves in Figures 3-5 indicate how the hit-list size should vary as the performance of the Answer-Selection module changes. As the performance decreases (*r* increases and the curves become lower), AS is less and less able to distinguish good answers from bad, and the optimum hit-list size decreases. This makes good sense given that the earlier the passage in the hit-list, the more likely it is to contain the correct answer; avoiding the extra noise which would come from considering more passages than necessary is a good strategy. Alternatively, the parameter *r* decreasing indicates the AS module is becoming more discriminating, and performs better by considering a larger set of passages without being overcome by noise from the weaker passages.

We ran our QA system on the three question sets for every hit list size from 1 to 50 and calculated the first-place scores with the NIST-provided patterns. The resulting graphs are presented in Figure 6 for LR and Figure 7 for DT.

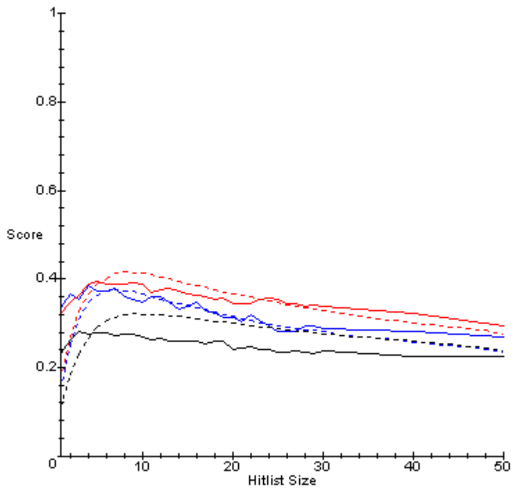

Figure 6. Actual system performance against hit-list size for LR. The top two curves are for TREC9 questions, the middle two for TREC8 and the bottom two for TREC10. The solid lines are actual first-place scores, the dashed lines are generated by the model.

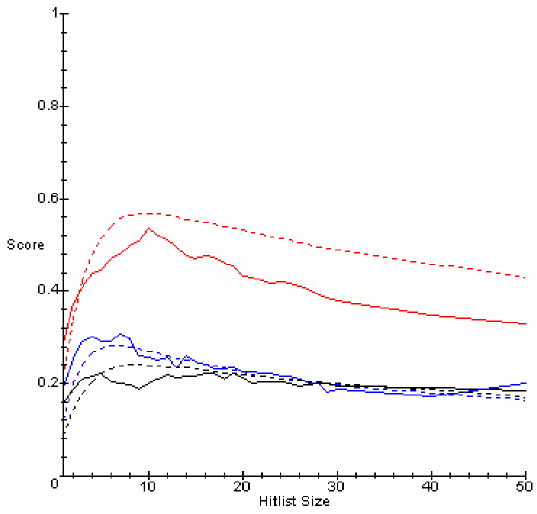

Figure 7. Actual system performance against hit-list size for the DT version of Answer Selection.

We summarize the predicted and measured peak positions for these six conditions in Table 4.

	LR8	LR9	LR10	DT8	DT9	DT10
Predicted	7	8-9	8-14	7	9-11	8-13
Measured	4-7	5-10	3-12	4-8	10	5-19

Table 4. Predicted vs. measured peak position of performance curves. In many cases the exact position

of the peak is not particularly clear, so a range is given. In every case the predicted value or range lies inside or considerably overlaps the observed value or range.
In trying to match predicted with measured peak positions, we run into difficulty trying to be exact, due to the flat peaks of the predicted curves and flat but jagged plateaus of observed data. For our purposes of selecting an optimum hit-list size, we can finesse the problem somewhat by erring on the side of selecting a larger size, due to the very gradual performance drop-off to the right of the maxima.

On the whole, we note very good agreement between theory and experiment, regarding both the shapes of the curves and the positions of the maxima. There are magnitude offset discrepancies in LR10 and DT9, which deserve further attention.

The mild disparities between prediction and observation can have arisen from a number of causes. The model of AS operation is clearly simplistic. Amongst many factors that the present model does not take into consideration is the fact that AS gives more weight to a candidate answer that shows up in more than one passage (this may well explain why the empirical LR data does better than predicted with very small hit-lists). The assumption that there is only one candidate answer of the right type per passage is clearly not generally true. The modeling of passage correctness as a function of position by an exponential function was a simple expedient that worked quite well, but other functions might fit better – this is a matter for further exploration.

Even though we have demonstrated both theoretically and empirically that an Answer Selection component’s performance characteristics do vary from question-set to question-set, the observed variation within the three TREC sets is rather minimal. Especially as all observed performance curves have a very mild slope to the right of the maximum point, we can choose for our system a single hit-list size in the region of 10 that should either be optimal or near-optimal for even an unseen set of questions of the style used by TREC. We plan to do such experiments in the near future to test this assertion.

Finally, we can make some observations about potential improvements in overall system scores. Our model, as illustrated in the performance curves in Figures 3-5, predicts FPS performance as a function not only of hit-list size but discrimination ratio *r*. By improving (lowering) *r*, the maximum achievable performance increases. Our system currently performs best on TREC9 data, with an FPS of *r*=0.45 for LR and .24 for DT. It is not unreasonable to suppose that with more training, this value will be achievable on TREC8 and TREC10 questions too. With the LR9 value of *r*, the peak projected performance of LR8 improves from .38 to .44, of LR10 from .32 to .34 (see Figure 8), while with the DT9 value of *r*, DT8 peak performance improves from .50 to .59, and DT10 from .24 to .29 (see Figure 9). Thus we see that an additional benefit of the model is its ability to indicate the possible value to be achieved from investing effort in Answer Selection training. The large differences between projected improvement in the LR and DT cases stem from the overtraining of DT on TREC9 questions. We believe the LR performance figures are more robust, and hence the improvement projections for LR more reliable.

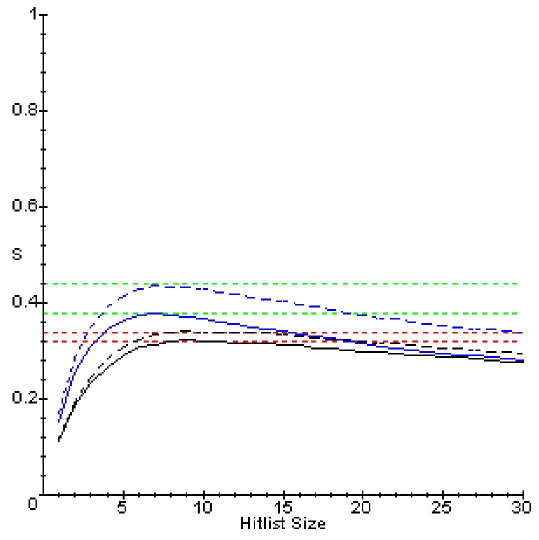

Figure 8. LR performance improvement from current (solid) to projected (dashed). The TREC8 curves are above the TREC10 curves.

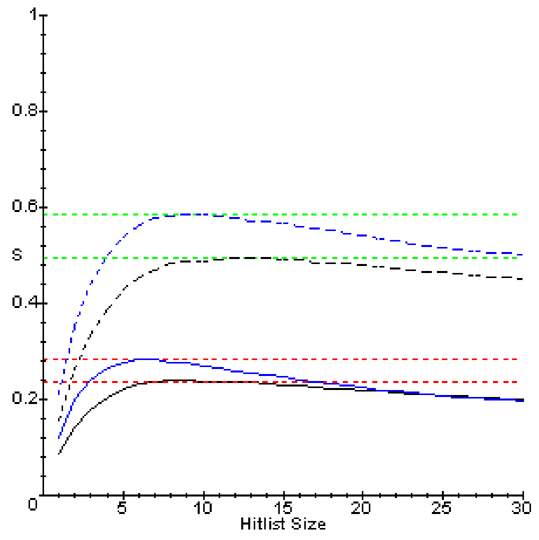

Figure 9. As Figure 8 but for DT.

5. Summary and Conclusions

We have derived a probabilistic model of QA system performance, for fixed question analysis and search engine, but with varying hit-list size and Answer Selection accuracy. We were able to fully parameterize the model using empirical measurements, namely the passage correctness distribution and the first-place score for a single hit-list size. From this model we were able to derive a formula to show how performance varies with hit-list size.

The agreement between the shapes of all six generated theoretical and empirical curves for performance against hit-list size suggests that the model is sound. The agreement in location of the peaks reinforces this view. It is reasonable to suppose that with a less simplistic model of the Answer Selection process or the data, even greater accuracy can be obtained.

The model made very few assumptions about the internal workings of the Answer Selection component. We believe that the way it was modeled is sufficiently generic as to be applicable to a wide variety of actual implementations of this process. Unfortunately, it is not so easy to run experiments with alternative Question-Answering systems as it is with other IR domains, for example in document retrieval or classification where it can be just a matter of using different formulas within a fixed framework. In QA, there is a synergy between the Question Analysis, Search and Answer Selection processes which requires their simultaneous replacement. Thus for us to experiment with other QA designs means either building entirely different systems, or having access to some other group’s complete system, which is not easy to arrange. We hope that in the future such access might become available.

The analysis presented in this paper represents the beginnings of a formal model of the entire question-answering process. The author hopes that by performing similar analyses of each of the stages in the QA pipeline, a set of control parameters can be established to determine the performance of the entire system as a function of them. The model so derived will be enormously useful in pinpointing where intellectual effort and/or retraining most needs to be applied. It will also allow for rapid (and complete) exploration of the multi-dimensional parameter space without running large numbers of time-consuming experiments.

The author wishes to thank Jennifer Chu-Carroll, Eric Brown and Dave Ferrucci for their helpful comments and suggestions. This work was supported in part by the Advanced Research and Development Activity (ARDA)'s Advanced Question Answering for Intelligence (AQUAINT) Program under contract number MDA904-01-C-0988.

References.

[1] V. Chaudhri and R. Fikes. Question answering systems: Papers from the *1999 AAAI fall symposium. Technical Report FS-99-02*, AAAI Press, 1999.

[2] Hovy, H., Gerber, L., Hermjakob, U., Lin, Chin-Yew, Ravichandran, D. “Towards Semantic-Based Answer Pinpointing”, *Proceedings of Human Language Technologies Conference*, pp. 339-345, San Diego CA, March 2001

[3] Moldovan, D., Harabagiu, S., Pasca, M., Mihalcea, R., Girju, R., Goodrum, R. and Rus, V. The Structure and Performance of an Open-Domain Question Answering System. In *Proceedings of the Conference of the Association for Computational Linguistics* (ACL-2000), 563–570.

[4] Pasca, M.A. and Harabagiu, S.M. “High Performance Question/Answering” *Proceedings of SIGIR 2001*, New Orleans, LA, 2001.

[5] Prager, J.M., Brown, E.W., Coden, A.R., and Radev, D.R. "Question-Answering by Predictive Annotation", *Proceedings of SIGIR 2000,* pp. 184-191, Athens, Greece, 2000.

[6] Prager, J.M., Radev, D.R. and Czuba, K. “Answering What-Is Questions by Virtual Annotation.” *Proceedings of Human Language Technologies Conference,* San Diego CA, pp. 26-30, March 2001b.

[7] Srihari, R. and W. Li. 2000. A Question Answering System Supported by Information Extraction. In *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics* (NAACL-00), 166–172.

[8] TREC8 - “The Eighth Text Retrieval Conference”, E.M. Voorhees and D.K. Harman Eds., NIST, Gaithersburg, MD, 2000.

[9] TREC9 - “The Ninth Text Retrieval Conference”, E.M. Voorhees and D.K. Harman Eds., NIST, Gaithersburg, MD, 2001.

[10] TREC10 - “The Tenth Text Retrieval Conference”, E.M. Voorhees and D.K. Harman Eds., NIST, Gaithersburg, MD, to appear in 2002.

[11] Voorhees, E.M. and Tice, D.M. “Building a Question Answering Test Collection”, *Proceedings of SIGIR 2000*, pp. 184-191, Athens, Greece, 2000.