BBN at TREC7: Using Hidden Markov Models for Information Retrieval

David R. H. Miller, Tim Leek, Richard M. Schwartz
BBN Technologies
Cambridge, MA USA
{tleek,dmiller,schwartz}@bbn.com

Abstract

We present a new method for information retrieval using hidden Markov models (HMMs) and relate our experience with this system on the TREC-7 ad hoc task. We develop a general framework for incorporating multiple word generation mechanisms within the same model. We then demonstrate that an extremely simple realization of this model substantially outperforms *tf.idf* ranking on both the TREC-6 and TREC-7 ad hoc retrieval tasks. We go on to present several algorithmic refinements, including a novel method for performing blind feedback in the HMM framework. Together, these methods form a state-of-the-art retrieval system that ranked among the best on the TREC-7 ad hoc retrieval task, and showed extraordinary performance in development experiments on TREC-6.

1 Introduction

Hidden Markov models have been applied successfully over the last two decades in a wide variety of speech and language related recognition problems including speech recognition [8], named entity finding [2], optical character recognition [9], and topic identification [18]. For TREC-7, we applied this technology for the first time to the problem of ad hoc information retrieval. On the TREC-7 ad hoc task, our entry ranked among the top tier of systems in average non-interpolated precison [22]. Moreover, our strong development results on TREC-6 hold out the promise of even higher performance from a mature HMM retrieval system.

In all HMM applications, the observed data (*e.g.* audio recording, image bitmap) is modeled as being the output produced by passing some unknown key (*e.g.* words, letters) through a noisy channel. In the ad hoc retrieval problem, we take the observed data to be the query Q, and the unknown key to be a desired relevant document D. The noisy channel is the mind of a user, who is imagined to have some fuzzy notion of which documents he wants, and who transforms that notion into the text of the query Q. Thus, we compute for each document the probability that D was the relevant document in the user's mind, given that Q was the query produced, *i.e.* $P(D \text{ is } R|Q)$, and rank the documents based on this measure.

Using probability models for information retrieval has a history almost four decades long, beginning with the work of Maron and Kuhns [10], and first seeing real application in the "standard probability model" pioneered by Robertson and Sparck-Jones [14]. More recently, however, the introduction of ad hoc constants and non-linear smoothing functions have improved performance steadily at the cost of straying further and further from the probabilistic framework. What started as a reasonable probability model is now masked by numerous heuristics. We believe our new hidden Markov model is more closely tied to its formal probabilistic underpinnings, making it easier to extend and reason about. In addition, the HMM's performance is on a par with the best automatic query systems.

The remainder of this paper is organized as follows: Section 2 lays out the basic theory of the hidden Markov model system and develops the formulas for a simple realization of it; Section 3 presents experimental results for the basic system on the TREC-6 and TREC-7 ad hoc tasks, and compares the system with the familiar *tf.idf* ranking; Section 4 develops several refinements of the basic HMM system, including a novel method of blind feedback and a more complex HMM which models the production of two-word phrases; Section 5 briefly explores the difference in performance between the TREC-6 and TREC-7 tests; lastly, Section 6 offers some conclusions regarding the system.

2 Probability Model

Given a user-generated query and a set of documents, we wish to rank the documents according to the probability that D is relevant, conditioned on the fact that the user produced Q, *i.e.* $P(D \text{ is } R|Q)$. Applying Bayes' rule, we decompose this into quantities that may be more easily estimated:

$$P(D \text{ is } R|Q) = \frac{P(Q|D \text{ is } R) \cdot P(D \text{ is } R)}{P(Q)} \tag{1}$$

where $P(Q|D \text{ is } R)$ is the probability of the query being posed, under the hypothesis that the document is relevant; $P(D \text{ is } R)$ is the prior probability that document D is relevant; and $P(Q)$ is the prior probability of query Q being posed.

Since $P(Q)$ will be identical for all documents D, we can safely disregard it for the purposes of sorting documents. We will return in Section 4.4 to the question of estimating the prior probability $P(D \text{ is } R)$, but for now we shall assume that it, too, is constant across all documents. We focus our attention on the remaining term $P(Q|D \text{ is } R)$.

We propose to model the generation of a query by a user as a discrete hidden Markov process dependent on the document the user has in mind.[1] A discrete hidden Markov model is defined by a set of output symbols, a set of states, a matrix of probabilities for transitions between the states, and a probability distribution of output symbols for each state. See [13] for an excellent introduction to hidden Markov models and their application. In the present application, we take the union of all words appearing in the corpus as the set of output symbols, and posit a separate state for each of several mechanisms of query word generation. For example, the states might represent choosing:

- a word from the desired document.
- a word that shares a root with a word appearing in the document.
- a word belonging to a lexicon specific to the topics of the document.
- a word commonly used for information requests.

In fact, the model can be easily extended to accommodate a broad variety of word generation mechanisms.

The process generates the words of the query by traversing a random sequence of states (with probabilities governed by the transition matrix), and at each state producing a word according to the output distribution of the state. Knowing the query that was produced, we can easily compute the probability of its being produced by each of the documents in the corpus. This is the $P(Q|D \text{ is } R)$ term that appears in Equation 1.

In order to use the HMM proposed, we must estimate the transition probabilities and the output distributions for every document in the corpus, since we have a separate HMM for each document. One typically computes estimates of HMM parameters with the EM (Estimation-Maximization) algorithm [5, 3] using training examples, in this case documents paired with queries to which they were relevant. However, such training examples are difficult to come by, and in practice it is usually the case that for the overwhelming majority of documents there are no training queries available. In the face of this difficulty we have proceeded by tying the transition probabilities between states across all models (*i.e.* making the transition matrix document independent), and by

1. In reality, a user rarely has only a single document in mind. However, we assign a probability to each hypothesis "the user has D in mind", and rank the documents using that probability.

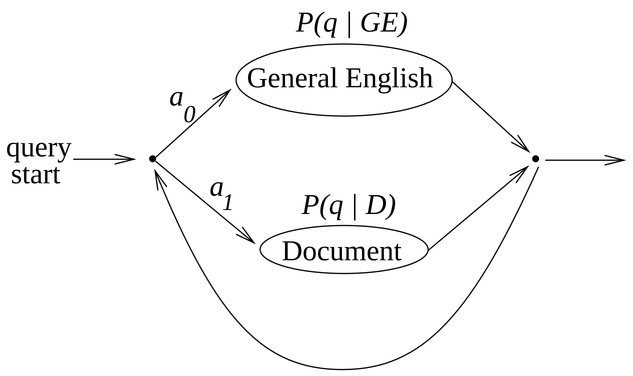

Figure 1: A simple two-state HMM.

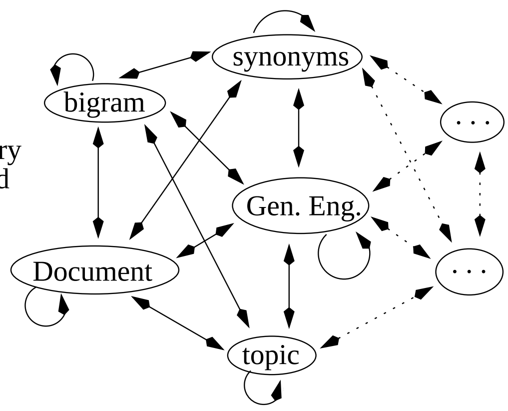

Figure 2: An expanded multi-state HMM.

abandoning EM entirely in favor of simple unigram estimation for the output distributions.

The number and purpose of the HMM states is left open to the practitioner, as is the topology of the transition graph connecting them. A very simple yet remarkably powerful configuration is the two-state, ergodic HMM shown in Figure 1. The first state represents choosing a word directly from the document; the second state represents choosing a word from "general query English." The transition probabilities are constrained so that $P(s_i \to s_j)$ is the same for all states s_i, which allows us to introduce two null states (states producing no output) and simplify the model. The values of the transition probabilities are estimated by EM. Since $a_0 + a_1 = 1$, there is only a single free parameter to be estimated, for which there is usually ample training data.

The output distribution for the document state is set to be a unigram on words appearing in the document. Ideally, we would like the distribution of the second state to be the unigram on words appearing in all queries. However, since we do not have sufficient training queries to estimate this distribution well, we use the unigram of the entire document corpus as an approximation to this ideal.

The two-state HMM shown in Figure 1 entails an independence assumption which we believe to be a reasonable approximation to the truth. In order to capture context effects better, we later incorporate a third state which models the production of bigrams. Indeed, the HMM framework easily generalizes to include additional states for synonyms, topic specific lexicons, etc., (see Figure 2). Sticking with the two-state model for now, the formula for $P(Q|D \text{ is } R)$ corresponding to Figure 1 is

$$P(Q|D_k \text{ is } R) = \prod_{q \in Q} (a_0 P(q|GE) + a_1 P(q|D_k)). \quad (2)$$

While this formula bears some resemblance to ones used by Ponte/Croft [11] and by Hiemstra/Kraiij [7], it involves a different smoothing term and is arrived at through a different theoretical derivation. Moreover, when extended along the theoretic lines suggested by the HMM (in Section 4) it diverges from these other works considerably.

3 Baseline Performance

In this section we report on ad hoc retrieval experiments we performed on the TREC-6 and TREC-7 test collections using the simple two-state system described in Section 2. We indexed each corpus separately to create inverted index files recording the number of times each word appears in each document. For this indexing, we ignored case and used Porter's algorithm [12] to conflate words with the same stem. We used a list of 397 "stop" words, and replaced all occurrences of these words with the special token *STOP*. In addition, we replaced cer-

$$
\begin{aligned}
tf.idf(Q,D) &= \sum_{q_i \in Q} wtf(q_i,D) \cdot idf(q_i) \\
wtf(q,D) &= \frac{tf(q,D)}{tf(q,D)+0.5+1.5\frac{l(D)}{al}} \\
idf(q) &= \frac{\log \frac{N}{n_q}}{N+1} \\
N &= \text{number of documets in the corpus} \\
n_q &= \text{number of documents in the corpus containing } q \\
tf(q,D) &= \text{number of times } q \text{ appears in } D \\
l(D) &= \text{length of } D \text{ in words} \\
al &= \text{average length in words of a document in the corpus}
\end{aligned}
$$

Figure 3: Comparison *tf.idf* formula.

tain 4-digit strings by the token **YEAR**, suspected dollar amounts by **DOLLAR**, and remaining digit strings by **NUMBER**. We applied the same preprocessing to the queries, and then excluded the stop words from further computation.

Keeping these indices fixed, we ranked documents for each query using the HMM measure of Equation 2, and compared this ranking with that given by the well known *tf.idf* measure. In particular, we used the *tf.idf* measure presented in [15] and reproduced in Figure 3. For the HMM transition probabilities, we used the EM algorithm to train the value of $a_1 = 0.3$ using training examples from the TREC-4 collection.

Table 1 shows the non-interpolated average precision (AveP) achieved by each ranking measure for a variety of test conditions. In all cases, the HMM system dramatically outperforms *tf.idf*, exceeding it by as much as 8 percentage points in absolute terms. Others [23] have reported somewhat better performance from this same *tf.idf* formula (though still not nearly as high as the HMM's performance[2]), which we attribute to differences in indexing which would degrade our results equally for both ranking formulas (e.g. exclusion of different SGML sections, different stop words, different stemming). Since we used the same index for both systems in Table 1, we feel this is a valid comparison.

It is puzzling that the score for the HMM on the full query decreases considerably (3.2%) from TREC-6 to TREC-7, whereas for *tf.idf* it increases slightly. See Section 5 for a discussion of this point.

4 HMM Refinements

Most IR systems do more than just compare the query words with the documents. This section describes four refinements we have added to our system: blind feedback, bigram modeling, feature dependent priors, and query section weighting. We describe and present experimental results for each method separately, and then present the results from using all the methods together.

4.1 Blind Feedback

Blind feedback is a well known technique for enhancing the performance of a retrieval system by conducting a preliminary search with the user's query, automatically constructing a new query based on the top-ranked documents from that initial search, and then conducting a second search with this new query before presenting anything to the user. The Rocchio algorithm [16] is perhaps the best known implementation of this idea, although there are many others as well [17, 4, 24]. We have developed a novel algorithm

2. Dr. J. Xu reported 23.2 AveP on the TREC-6 full queries and 22.6 on the TREC-7 full queries in [23] using the formulas in Figure 3 and the UMass INQUERY indexing.

	TREC-6			TREC-7		
	HMM	*tf.idf*	Diff	HMM	*tf.idf*	Diff
Title	21.6	15.9	+5.8	16.1	11.6	+4.5
Desc	18.1	11.9	+6.2	18.3	14.2	+4.1
Narr	21.5	15.8	+5.7	17.7	14.7	+3.0
Full	27.1	18.9	+8.2	23.9	19.0	+4.9

Table 1: HMM ranking vs. *tf.idf* on TREC-6 and TREC-7.

for blind feedback that is particularly suited for use with hidden Markov models.

Our approach is to augment the initial query with words appearing in two or more of the top N documents, and to adjust the HMM transition probabilities for each word to account for how unexpected those appearances are. For example, seeing the word "very" in 90% of the top N retrieved document carries little information, while seeing "Nixon" in 90% of those same documents is highly informative. We develop a method below that captures this distinction in a principled fashion.

For the two-state HMM, the transition probabilities between the two states can be estimated by the EM algorithm using training queries. For each observation, the EM algorithm distributes the count for that observation to the two-states in proportion to the likelihood of each state's generating that word. Since the document state typically contains only hundreds of words, while the general English state contains hundreds of thousands, whenever the document state has a non-zero probability for a word it usually dwarfs the probability from the general English state. As a result, the estimate from EM for transition into the document state is very close to that obtained by calculating the probability that a query word is in a document, given that the document is relevant. This is[3]

$$P(q' \in D' | D' \text{ is rel. to Q}') = \frac{1}{|\mathcal{Q}|} \sum_{Q' \in \mathcal{Q}} \sum_{q' \in Q'} \frac{|D' \ s.t. \ q' \in D', D' \text{ is rel. to } Q'|}{|Q'| \cdot |D' \text{ is rel. to } Q'|}. \tag{3}$$

where $mathcalQ$ is the set of available training queries.

Using this insight as our motivation, we consider the case where we have additional query terms taken from the top N ranked documents from a preliminary search for query Q. Given these top N documents, we partition the complete corpus lexicon into $N+1$ disjoint sets of words that we call m-intersections:

$$I_{m,Q} = \left\{ \begin{array}{c} w \text{ appearing in exactly } m \text{ of the} \\ \text{top } N \text{ documents for query } Q \end{array} \right\} \tag{4}$$

for $m = 0, 1, 2, \ldots, N$. For those words $q \in I_{m,Q}$, it is tempting to set the transition probability into the document state to be

$$P(q' \in D' | D' \text{ is rel. to } Q', q' \in I_{m,Q'}). \tag{5}$$

But merely being in a high-order m-intersection is not enough to be an important term. The most common words in the corpus like "the", "a", and "is" would turn up in $I_{N,Q}$ nearly all the time merely as a result of their document frequency, and these carry no information about the query Q.[4]

To compensate for this phenomenon, we condition on and subtract out the baseline document frequency of the words. We define $df(w)$ to be the percentage of documents in the corpus containing word w. We then define

$$\gamma_{m,Q',x} = P\left(q' \in D' \;\middle|\; \begin{array}{l} D' \text{ is rel. to } Q', \\ q' \in I_{m,Q'}, df(q') = x \end{array} \right) \tag{6}$$

3. Here and in discussions below, we use a prime ($'$) to indicate a variable that refers to a generic or training object, while an unmodified variable refers to a specific or test object. Thus Q' is some abstract query while Q is the current query posed to the system.

4. The words "the", "a" and "is" are in our stop list, of course, but the same argument applies for any word with high document frequency that is not in the stop list.

	TREC-6	TREC-7
basic HMM	27.1	23.9
w/blind feedback	30.6	27.4
improvement	+3.5	+3.5

Table 2: Performance gain from blind feedback.

and set the transition probability for query terms in $I_{m,Q}$ with a particular document frequency $df(q)$ to be

$$a_0 = \gamma_{m,Q,df(q)} - df(q). \tag{7}$$

In order to estimate this parameter, we take many training queries and run a preliminary search with each of them to obtain the top N ranked documents. We then count the number of documents each query term appears in. Since these are training queries, we know the complete set of documents that are relevant to each query. With this information we can estimate $\gamma_{m,Q',x}$ by the formula

$$\frac{1}{|Q|}\sum_{Q'\in Q}\sum_{q'\in Q'}\frac{\left|\begin{array}{l} D' \ s.t.\ q'\in D', D' \text{ is rel. to } Q', \\ q'\in I_{m,Q'}, df(q')=x \end{array}\right|}{|Q'|\cdot\left|\begin{array}{l} D' \ s.t.\ D' \text{ is rel. to } Q', \\ q'\in I_{m,Q'}, df(q')=x \end{array}\right|}. \tag{8}$$

Blind feedback produced a large and robust performance improvement. We used the top 6 documents from the first retrieval to form m-intersections. We discarded the terms in I_0 and I_1 unless they appeared in the original query as well. We trained the transition probabilities using the 50 queries of the TREC-6 collection, and tested on both the TREC-6 and TREC-7 collections. The improvement of 3.5 AveP on the TREC-6 queries (unfair test on training) carried over exactly to the fair test condition of the TREC-7 queries (see Table 2), indicating that we have not overtuned our parameters to the training data.

4.2 Bigrams

Many words have a distinctive meaning when used in the context of another word, or in a larger phrase. For example, a query using the phrase "white house" is much more likely to be satisfied by a document using those two words in sequence than by one that has them separately. Other systems have attempted to model this phenomenon by fusing selected phrases into a new single term (*e.g.* "white_house", "Pope_John_Paul_II") and using it either instead of or in addition to the individual words [1]. This approach, however, requires that all sentences, whether in documents or queries, be segmented into terms (*e.g.* is "white house secretary" transformed into "white_house secretary" or "white house_secretary"?).

We have taken an alternate approach, in which the words of a query are modeled as always being generated one at a time, but the probabilities governing this generation are conditioned on the identity of the previous word generated. This is accomplished by adding to our HMM a third, document-dependent bigram state (see Figure 4). The output distribution of this state[5] is given by

$$P(q_n|D,q_{n-1}) = \frac{|q_{n-1}q_n \text{ in } D|}{|q_{n-1} \text{ in } D|} \tag{9}$$

where q_n is the current word of the query and q_{n-1} is the previous word. In the event that a document does not contain the previous word of the query the computation backs off to the two-state model, as the denominator of the bigram state output probability would be zero.

Generating a word via this state corresponds to the user's continuing a two-word phrase that was initiated in the previous word. Since the bigram state output probabilities are typically one to three orders of magnitude greater than those in the unigram states, a document containing a bigram that matches the query gains a big boost in likelihood.

The three-state system has a second free parameter, a_2, in the transition probabilities. We optimized the values for a_1 and a_2 to maximize AveP on the TREC-6 task, arriving at $a_1 = 0.29, a_2 = 0.01$. Table 3 shows the effect of using the bigram-state with these transition values for both the TREC-6 and

5. Strictly speaking, the output distribution of an HMM state cannot be dependent on any of the previous outputs. However the unorthodox HMM presented here is equivalent to a strict HMM having one state per distinct word of the document in place of the the single bigram state shown in Figure 4.

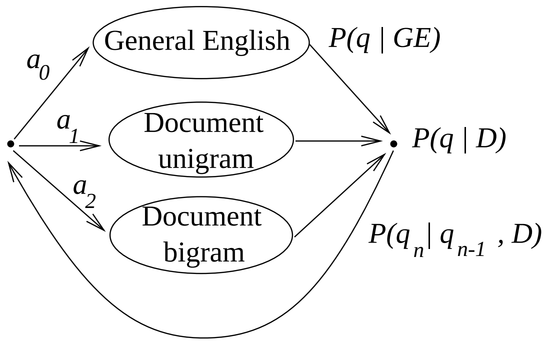

Figure 4: An HMM that models bigram production.

	TREC-6	TREC-7
basic HMM	27.1	23.9
w/bigrams	28.1	24.4
improvement	+1.0	+0.5

Table 3: Performance gains from adding a bigram state.

TREC-7 tasks. The fair gain, while solid, is only half as big as the unfair improvement seen on the TREC-6 task. As there are only two free parameters being tuned, statistical variance between test sets seems a more likely explanation for the discrepancy than overtraining.

4.3 Query Section Weighting

Examining the topics from past TREC evaluations, it was clear that the words in the "Title" section were more important than those in the remainder of the topic (although it was unclear whether the "Description" section was more or less useful than the "Narrative" section). In a more general context, a user may wish to designate some portions of his query as more important than others. To exploit this observation, we imagine a simple model in which a user repeats a word multiple times in a query to indicate its greater importance. Under this model, the "Title" declaration is taken simply as shorthand for "repeat these words ν times". Applying these repetitions to

	TREC-6	TREC-7
basic HMM	27.1	23.9
w/query weights	30.0	25.1
improvement	+2.9	+1.2

Table 4: Performance gains query section weighting.

Equation 2 yields

$$P(Q|D_k \text{ is } R) = \prod_{q \in Q} (a_0 P(q|GE) + a_1 P(q|D_k))^{\nu_{s(q)}} \tag{10}$$

where $\nu_{s(q)}$ is the weight (*i.e.* number of repetitions) for the section of the query in which q appears.

We optimized the weights to maximize AveP for the TREC-6 task, which produced values of $\nu_{title} = 5.7$, $\nu_{desc} = 1.2$, $\nu_{narr} = 1.9$. The gain from applying these weights to the TREC-6 task is unfairly optimistic, but Table 4 shows that using these same query section weights improves AveP by 1.2 on TREC-7 in a fair test. In an interactive setting, it would be easy to make term or section weights available to user manipulation.

4.4 Document Priors

In the discussion of Section 2, we made the simplifying assumption that the prior probability of relevance, $P(D \text{ is } R)$, is constant for all documents. However, it is reasonable to think that longer documents may be more useful in general than short ones, or that articles from a refereed journal may be more informative than those from a supermarket tabloid. With this in mind, we searched for features that could predict prior relevance on TREC-6. The most predictive features we found were source, length, and average word-length. Conditioning the document prior on these features and estimating the marginals on TREC-6 yielded a small gain for that corpus, but this gain did not carry over to the fair test set of TREC-7 (see Table 5). Nonetheless, we believe that using a non-constant prior is a good idea and have retained this mechanism in our system.

	TREC-6	TREC-7
basic HMM	27.1	23.9
w/non-constant prior	27.6	24.0
improvement	+0.5	+0.1

Table 5: Performance with non-constant prior.

	TREC-6	TREC-7
basic HMM	27.1	23.9
w/blind feedback	+3.5	+3.5
w/query weights	+2.9	+1.2
w/non-constant prior	+0.5	+0.1
w/bigrams	+1.0	+0.5
HMM w/all refinements	33.2	28.0

Table 6: Performance gains from refinements to the HMM system.

4.5 Additivity of Refinements

Table 6 summarizes the improvements in AveP due to the various extensions described in this section. The first row shows AveP for the basic HMM system, the next four rows show the gain from using any one of the techniques by itself, and the final row shows the result of using all four techniques together. The overall improvement (+6.1 for TREC-6, +4.1 for TREC-7) is roughly 77% of the sum of the individual improvements (+7.9 for TREC-6, +5.3 for TREC-7), indicating that the information captured by these techniques are largely orthogonal to each other.

5 Discussion

Although the HMM system performed well in TREC-7, it did not yield the extraordinary performance we saw in our development work on TREC-6 (our AveP of 33.2 is a full 4.4 points better than the best result reported in the TREC-6 conference [20]). We have tried to understand what accounts for the difference in these results.

Considering first the performance of the basic HMM presented Section 3, we are puzzled by the observation that the score for the HMM on the full query decreases considerably (3.2% absolute) from TREC-6 to TREC-7, whereas for *tf.idf* it increases slightly (see Table 1). In this comparison, the index was held fixed, as was the query pre-processing. Only the ranking function was changed. Since both measures use only exact matches on the stems in the query, problems with stemming, stopping, or document handling should have affected both systems equally.

We decomposed the results on TREC-6 and TREC-7 into query groups having similar numbers of relevant documents (see Table 7). We found that the degradation of the HMM performance across TREC's is entirely localized to those queries that have fewer than 20 relevant documents in the entire corpus (34.5% AveP on TREC-6, 14.6% AveP on TREC-7). Moreover, the standard deviation of per-query AveP on all 50 queries for the HMM (23 on TREC-6, 19 on TREC-7) is considerably higher than the per-query standard deviation for *tf.idf* (15 for TREC-6, 14 for TREC-7). In the end, statistical variance may be the only explanation for the apparent inverse movement in results between the two systems.

Since the overall AveP is an unweighted average across queries (not across documents), a small change in the documents retrieved for the most specific queries has a disproportionate effect on the overall AveP. In search of a measure less sensitive to the location of any single document, we computed the weighted AveP for our systems:

$$\text{wAveP} = \frac{\sum_Q r(Q)\text{AveP}(Q)}{\sum_Q r(Q)}, \qquad (11)$$

where $r(Q)$ is the total number of documents in the corpus judged relevant to Q. As Table 8 shows, the results for both *tf.idf* and the HMM are much more stable using this measure than using the unweighted AveP. This is to be expected, since our training is on query/relevant document pairs, without regard to numbers of documents per query. Perhaps a weighted training method is needed to optimize the model's performance when averaged across queries rather than documents.

Looking next at the extensions and refinements presented in Section 4, we see that the overall improvement was +6.1 for TREC-6, but only +4.1 for TREC-7. Of course, many of the parameters of the

	TREC-6			TREC-7		
#rel docs	#q	HMM	*tf.idf*	#q	HMM	*tf.idf*
0-20	15	34.5	20.4	9	14.6	13.5
21-80	17	25.5	18.4	19	30.5	23.4
> 80	18	22.5	18.2	22	22.3	17.5
All queries	50	27.1	18.9	50	24.0	19.0

Table 7: AveP as a function of total number of relevant documents for a query.

	HMM	*tf.idf*
TREC-6	22.1	18.0
TREC-7	23.2	18.9

Table 8: Weighted AveP for TREC-6 and TREC-7.

algorithms were tuned for the TREC-6 data, making the performance there overly optimistic. To understand better the effect of unfairly tuning to TREC-6, we retuned the entire system to optimize performance on the TREC-7 test. The AveP increased by only 0.7 to 28.7. Put another way, a more realistic estimate of our fair performance on TREC-6 is $33.2 - 0.7 = 32.5$ AveP. This still leaves a gap of 1.3 between the gains from refinements on TREC-7 and the gains from refinements on TREC-7, for which we have no explanation at present.

6 Conclusion

We have presented a novel method for performing information retrieval using hidden Markov models. This framework offers a rich setting in which to incorporate a variety of techniques, both new and familiar. We have experimented with a system that implements blind feedback, bigram modeling, query weighting, and document-feature dependent priors. Our official submission for the ad hoc task of the TREC-7 conference achieved an AveP of 28.0 and was among the top tier of systems [22]. Our own, unofficial test results on the TREC-6 ad hoc task show an AveP substantially higher than any of the official results reported in [20].

We believe that this approach holds great promise beyond its already demonstrated success. The work we have reported represents BBN Technologies' initial foray into the field of information retrieval. The system was conceived, developed, and debugged with only 1.5 people working for eight months. Naturally, there are many familiar ideas that we were unable to incorporate into our system due to time and labor constraints. Among the most glaring examples are an absence of passage retrieval, explicit synonym modeling, and concept modeling. We believe that the HMM approach can be extended to accommodate these and many other ideas under a unified, well-grounded framework. More work still needs to be done.

References

[1] J. Allan, J. P. Callan, W. B. Croft, L. Ballestros, D. Byrd, R. Swan and J. Xu, "INQUERY does battle with TREC-6". In D. K. Harman, editor, Proceedings of the Sixth Text Retrieval Conference (TREC-6), NIST Special Publication 500-240 (1996).

[2] D. Bikel, S. Miller, R. Schwartz, R. Weischedel, "Nymble: a high-performance learning name-finder." Fifth Conference on Applied Natural Language Processing, (published by ACL), pp 194-201 (1997).

[3] W. Byrne, *Encoding and Representing Phonemic Sequences Using Nonlinear Networks*, Ph.D. Dissertation, University of Maryland, College Park, 1993.

[4] W. Cohen and Y. Singer, "Context-sensitive learning methods for text categorization". In *Proceedings of the 19th Annual International ACM*

SIGIR conference on Research and Development in Information Retrieval, pp. 307-315, (1996).

[5] A. Dempster, N. Laird and D. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm", *Journal of the Royal Statistical Society (B)*, Vol. 39, No. 1, pp. 1-22, 1977.

[6] D. Harman, "Overview of the Fourth Text REtrival Conference." In D. K. Harman, editor, Proceedings of the Sixth Text Retrieval Conference (TREC-6), NIST Special Publication 500-236, pp. 1-24 (1996).

[7] D. Hiemstra and W. Kraaij, "TREC-7 working notes: Twenty-One in ad-hoc and CLIR" In D. K. Harman, editor, Proceedings of the Seventh Text Retrieval Conference (TREC-7). Elsewhere in this volume (1999).

[8] J. Makhoul and R. Schwartz, "State of the art in continuous speech recognition", Proc. Natl. Acad. Sci. USA 92, pp 9956-9963 (1995).

[9] J. Makhoul, R. Schwartz, C LaPre, I. Bazzi, "A script-independent methodolgy for optical character recognition." Pattern Recognition, Vol 31, No. 9, pp. 1285-1294 (1998).

[10] M. E. Maron and K. L. Kuhns, "On relevance, probabilistic indexing and information retrieval." Journal of the Associations of Computing Machinery, 7, pp. 216-244 (1960).

[11] J. Ponte and W. B. Croft, "A Language Modeling Approach to Information Retrieval." In *Proceedings of the 21st Annual International ACM SIGIR conference on Research and Development in Information Retrieval*, pp. 275-281, (1998).

[12] M. F. Porter, "An Algorithm for Suffix Stripping." Program, 14(3), pp. 130-137 (1980).

[13] L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition", Proc. IEEE 77, pp. 257-286 (1989).

[14] S. E. Roberston, and K. Sparck Jones "Relevance weighting of search terms." Journal of the ASIS, 27, pp. 129-146 (1976).

[15] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, M. Gatford, "Okapi at TREC-3." In D. K. Harman, editor, Proceedings of the Third Text Retrieval Conference (TREC-3), NIST Special Publication 500-226 (1995).

[16] J. J. Rocchio, "Relevance feedback in information retrieval". In *The SMART Retrieval System–Experiments in Automatic Document Processing*, pp. 313-323, Englewood Cliffs, NJ, 1971. Prentice Hall, Inc.

[17] R. Schapire, Y. Singer, A. Singhal, "Boosting and Rocchio Applied to Text Filtering". In *Proceedings of the 21st Annual International ACM SIGIR conference on Research and Development in Information Retrieval*, pp. 215-223, (1998).

[18] R. Schwartz, T. Imai, F. Kubala, L. Nguyen, J. Makhoul, "A maximum likelihood model for topic classification of broadcast news." Proc. Eurospeech '97, Rhodes, Greece, pp. 1455-1458 (1997).

[19] E. Voorhees and D. Harman, "Overview of the Sixth Text REtrival Conference." In D. K. Harman, editor, Proceedings of the Sixth Text Retrieval Conference (TREC-6), NIST Special Publication 500-240, pp. 1-24 (1998).

[20] E. Voorhees and D. Harman, "Appendix A: Adhoc Results." In D. K. Harman, editor, Proceedings of the Sixth Text Retrieval Conference (TREC-6), NIST Special Publication 500-240, Appendix A (1998).

[21] E. Voorhees and D. Harman, "Overview of the Seventh Text REtrival Conference." In D. K. Harman, editor, Proceedings of the Seventh Text Retrieval Conference (TREC-7). Elsewhere in this volume (1999).

[22] E. Voorhees and D. Harman, "Appendix A: Adhoc Results." In D. K. Harman, editor, Proceedings of the Seventh Text Retrieval Conference (TREC-7). Elsewhere in this volume (1999).

[23] J. Xu. Personal communication, October, 1998 (1998).

[24] J. Xu and B. Croft, "Improving the Effectiveness of Information Retrieval with Local Context Analysis", 1998. To appear in ACM Transactions on Information Systems (1999).