A Hidden Markov Model Information Retrieval System

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

Abstract

We present a new method for information retrieval using hidden Markov models (HMMs). 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 standard *tf.idf* ranking on both the TREC-6 and TREC-7 ad hoc retrieval tasks. We go on to present a novel method for performing blind feedback in the HMM framework, a more complex HMM that models bigram production, and several other algorithmic refinements. Together, these methods form a state-of-the-art retrieval system that ranked among the best on the TREC-7 ad hoc retrieval task.

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 [9], named entity finding [2], optical character recognition [10], and topic identification [19]. In the present work, we describe an application of this technology to the problem of ad hoc information retrieval.

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 notion (either rough or precise) 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 [11], and first seeing real application in the "standard probability model" pioneered by Robertson and Sparck-Jones [15]. More recently, however, the introduction of ad hoc constants and non-linear smoothing functions have improved performance steadily at the cost of drifting 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 (Section 4.1) and a more complex HMM which models the production of two-word phrases (Section 4.2); Section 4 also presents experimental results with these and other techniques used singly and jointly; lastly, Section 5 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

SIGIR '99 8/99 Berkley, CA USA

more easily estimated:

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

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 set of probabilities for transitions between the states, and a probability distribution on output symbols for each state. An observed sampling of the process (*i.e.* the sequence of output symbols) is produced by starting from some initial state, transitioning from it to another state, sampling from the output distribution at that state, and then repeating these latter two steps. The transitioning and the sampling are non-deterministic, and are governed by the probabilities that define the process. The term "hidden" refers to the fact that an observer sees only the output symbols, but doesn't know the underlying sequence of states that generated them (since the same symbol could come from any of the states). See [14] 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. There is a separate process for each individual document which generates the words of the query by traversing a random sequence of states, 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.

While one can employ as many HMM states as one can imagine, we will restrict our discussion here to the simple but powerful two-state HMM shown in Figure 1 (in Section 4.2 we discuss our experience with a three-state HMM). The first state, labelled "Document" represents choosing a word directly from the document.

[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.

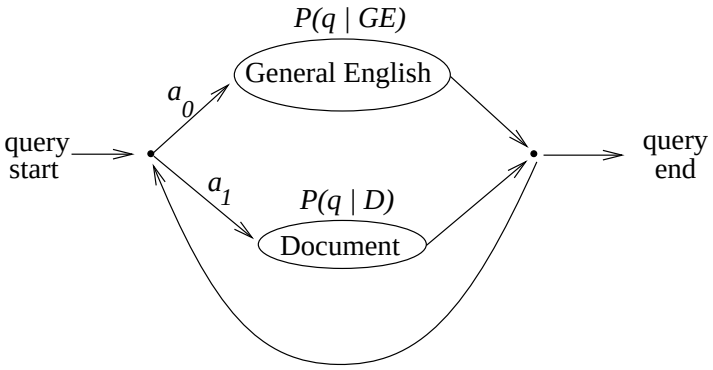

Figure 1: A simple two-state HMM.

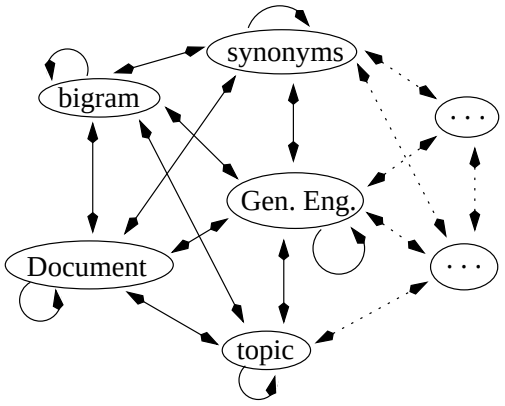

Figure 2: An expanded multi-state HMM.

The second, labelled "General English", represents choosing a word that is unrelated to the document, but that occurs commonly in natural language queries. More generally, the model can be easily extended to accommodate a broad variety of word generation mechanisms involving synonyms, topic lexicons, or proper names (see Figure 2).

The model pictured in Figure 1 carries the further simplification that there are only two transition probabilities, a_0 and a_1 instead of the possible four. This corresponds to an assumption that the choice of which kind of word to generate next is independent of the previous such choice. This assumption (which does not preclude staying in the same state for several words in a row) allows us to introduce two null states (states producing no output) and simplify the drawing of the model.

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. The EM (Estimation-Maximization) algorithm [6, 3] is the canonical method for computing these parameters. Performing EM re-

quires examples of sequences that have been generated by the HMM, which in this case means examples of 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 made two practical simplifications. First, we assume that the transition probabilities between the states are the same for all documents. Second we abandon EM entirely for the estimation of the output distributions, and instead use simple maximum likelihood estimates for each document. Though not entirely satisfactory, these assumptions make the parameter estimation problem tractable while still delivering excellent performance.

To estimate the transition probabilities a_0 and a_1, we use training examples and the EM algorithm as described above, pooling the training data to get a single estimate used across all documents.

The output distribution for the "Document" state, $P(q|D)$, is set to be the sample distribution on words appearing in that document. Explicitly, for document D_k we set

$$P(q|D_k) = \frac{\text{number of times } q \text{ appears in } D_k}{\text{length of } D_k} \quad (2)$$

which is the distribution having the maximum likelihood of producing D_k itself through repeated sampling.

Ideally, we would like the probability distribution of the second state to be the distribution of words appearing in natural language queries. However, since we do not have sufficient numbers of training queries to estimate this distribution well, we use instead the sample distribution of the entire document corpus as a poor approximation to this ideal. Thus, we label this state "General English", and estimate it by

$$P(q|GE) = \frac{\sum_k \text{number of times } q \text{ appears in } D_k}{\sum_k \text{length of } D_k} \quad (3)$$

where the sum is taken over all documents in the corpus.

With these parameters estimated, we may now state the the formula for $P(Q|D_k \text{ is } R)$ corresponding to Figure 1:

$$P(Q|D_k \text{ is } R) = \prod_{q \in Q} (a_0 P(q|GE) + a_1 P(q|D_k)). \quad (4)$$

This expression, in turn, is used in Equation 1 to compute $P(D_k \text{ is } R|Q)$, which is the value we use to rank documents.

While Equation 4 bears some resemblance to ones used by Ponte/Croft [12] and by Hiemstra/Kraiij [8], 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 System 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. The TREC-6 collection comprises 556,077 documents from a variety of news and governmental agencies [20]. The TREC-7 collection is a subset of the TREC-6 collection containing 528,155 documents [22]. A set of 50 test topics (*i.e.* queries) accompanies each collection, as does a set of relevance judgments listing the documents relevant to each topic. The topics contain "Title", "Description", and "Narrative" sections, with an average total length (including repeated words) of 88.4 words for TREC-6 and 57.6 words for TREC-7[20, 22].

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 [13] 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 certain 4-digit strings by the token **YEAR**, suspected dollar amounts by **DOLLAR**, and remaining digit strings by **NUMBER**. We applied the same pre-processing to the queries, and then excluded the stop words from further computation. After removing all stop words, the TREC-6 queries had an average of 26.5 unique terms, and the TREC-7 had an average of 17.6 unique terms.

Keeping these indices fixed, we ranked documents for each query using the HMM measure of Equation 4, and compared this ranking with that given by the well known *tf.idf* measure. In particular, we used the *tf.idf* measure presented in [16] 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 [7].

Table 1 shows the non-interpolated average precision (AveP) achieved by each ranking measure for a variety of test conditions.[2] In all cases, the HMM system dramatically outperforms *tf.idf*, exceeding it by as much as 8 percentage points in absolute terms. Others [24] have reported somewhat better performance from this same *tf.idf* formula (though still not nearly as high as

[2] If $r(D)$ is the rank of document d and Rel is the set of relevant documents for a query Q, then non-interpolated average precision for Q is defined as

$$\frac{1}{|Rel|} \sum_{D \in Rel} \frac{|\{D' \in Rel, r(D') \leq r(D)\}|}{r(D)}.$$

This quantity is then averaged across all 50 queries.

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

Figure 3: Comparison *tf.idf* formula.

	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 scoring vs. *tf.idf* on TREC-6 and TREC-7.

the HMM's performance[3]), 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.

We are puzzled by the observation 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. Since both measures use only exact matches on the stems in the query, we can think of no characteristic change in the query set that would hurt one system disproportionately more or less than the other. Statistical variance may, in the end, be the only explanation for the apparent inverse movement in results between the two systems.

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.

[3]Dr. J. Xu reported 23.2 AveP on the TREC-6 full queries and 22.6 on the TREC-7 full queries in [24] using the formulas in Figure 3 and the UMass INQUERY indexing.

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 [17] is perhaps the best known implementation of this idea, although there are many others as well [18, 4]. We have developed a novel algorithm for blind feedback that is particularly suited for use with hidden Markov models. We have also used this algorithm for true, user-guided, relevance feedback with great success [5].

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 (a_0 and a_1) 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 a_1, the 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[4]

$$P(q' \in D' | D' \text{ is rel. to Q}') = \frac{1}{|\mathcal{Q}|} \sum_{Q_i \in \mathcal{Q}} \sum_{w \in Q_i} \frac{|D \text{ s.t. } w \in D, D \text{ is rel. to } Q_i|}{|Q_i| \cdot |D \text{ is rel. to } Q_i|}. \quad (5)$$

where $\mathcal{Q}$ is the set of available training queries.

Using the fact that Equation 5 and the EM algorithm give similar estimates for a_1 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:

$$I_{m,Q} = \left\{ \begin{array}{c} w \text{ appearing in exactly } m \text{ of the} \\ \text{top } N \text{ documents for query } Q \end{array} \right\} \quad (6)$$

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

$$P(q' \in D' | D' \text{ is rel. to } Q', q' \in I_{m,Q'}). \quad (7)$$

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.[5]

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

$$\gamma_{m,Q',x} = P\left(q' \in D' \;\middle|\; \begin{array}{l} D' \text{ is rel. to } Q', \\ q' \in I_{m,Q'}, df(q') = x \end{array} \right) \quad (8)$$

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

$$a_1 = \gamma_{m,Q,df(q)} - df(q). \quad (9)$$

In truth, this is no longer a probability (and indeed not even not guaranteed to positive), but rather the excess likelihood of a word's appearing due to relevance. In practice, $\gamma_{m,Q,df(q)}$ typically dwarfs $df(q)$, so the estimate for a_1 is close to a true probability. We arbitrarily floor the estimate at 0 to avoid negative values.

To compute an estimate for γ, 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

$$\frac{1}{|\mathcal{Q}|} \sum_{Q_i \in \mathcal{Q}} \sum_{w \in Q_i} \frac{\left| \begin{array}{l} D \text{ s.t. } w \in D, D \text{ is rel. to } Q_i, \\ w \in I_{m,Q_i}, df(w) = x \end{array} \right|}{|Q_i| \cdot \left| \begin{array}{l} D \text{ s.t. } D \text{ is rel. to } Q_i, \\ w \in I_{m,Q_i}, df(w) = x \end{array} \right|}. \quad (10)$$

	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.

Blind feedback produced a large and robust performance improvement (see Table 2). We used the top 6 documents from the first retrieval to form m-intersections. We discarded the terms in $I_{0,Q}$, $I_{1,Q}$, and $I_{2,Q}$ unless they appeared in the original query as well. The expanded TREC-6 queries had an average of 99.2 unique terms (up from 26.5 unexpanded) and the TREC-7 queries had an average of 85.2 unique terms (up from 17.6 unexpanded). 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, 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

[4]Here and in discussions below, we use a prime (') to indicate a variable that refers to a generic object, while an unmodified variable refers to a specific training or test object. Thus, Q' is some abstract query while Q is the current query posed to the system and Q_i is one of several training queries.

[5]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.

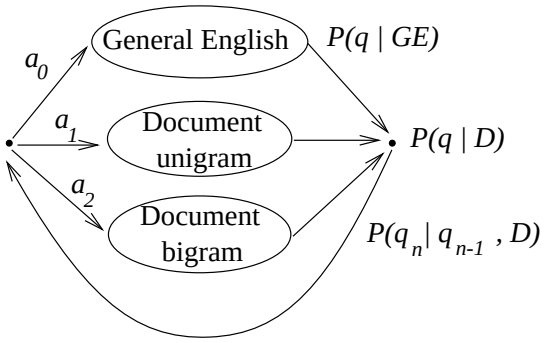

Figure 4: An HMM that models bigram production.

	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.

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[6] is given by

$$P(q_n|D_k, q_{n-1}) = \frac{\text{number of times } q_{n-1}q_n \text{ appears in } D_k}{\text{number of times } q_{n-1} \text{ appears in } D_k} \quad (11)$$

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 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.

[6]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.

	TREC6	TREC7
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.

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 Equation 4 yields

$$P(Q|D_k \text{ is } R) = \prod_{q \in Q} (a_0 P(q|GE) + a_1 P(q|D_k))^{\nu_{s(q)}} \quad (12)$$

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

	TREC6	TREC7
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.

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.

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.

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. For both tasks, then, the improvement in AveP coming from unfairly tuning the parameters of these 4 techniques is roughly +5 - +6 AveP, while the improvement on TREC-7 from a fair application of these techniques was +4 AveP. Thus, a more realistic estimate of our fair performance on TREC-6 is $27.1 + 4 = 31.1$ AveP

5 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 [23]. 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 [21].

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.

[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] S. Colbath, "Rough'n'Ready: A meeting recorder and browser". A research note of the *Perceptual User Interfaces Conference*, San Francisco, CA, November 1998 (1998).

[6] 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.

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

[8] 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). To appear 1999.

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

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

[11] 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).

[12] 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).

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

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

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

[16] 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).

[17] 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.

[18] 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).

[19] 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).

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

[21] 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).

[22] E. Voorhees and D. Harman, "Overview of the Seventh Text REtrieval Conference." In D. K. Harman, editor, Proceedings of the Seventh Text Retrieval Conference (TREC-7). To appear 1999.

[23] E. Voorhees and D. Harman, "Appendix A: Adhoc Results." In D. K. Harman, editor, Proceedings of the Seventh Text Retrieval Conference (TREC-7). To appear 1999.

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