Automatic Feature Selection in the Markov Random Field Model for Information Retrieval

Donald Metzler
metzler@cs.umass.edu

Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts
Amherst, MA 01003

ABSTRACT

Previous applications of the Markov random field model for information retrieval have used manually chosen features. However, it is often difficult or impossible to know, *a priori*, the best set of features to use for a given task or data set. Therefore, there is a need to develop automatic feature selection techniques. In this paper we describe a greedy procedure for automatically selecting features to use within the Markov random field model for information retrieval. We also propose a novel, robust method for describing classes of textual information retrieval features. Experimental results, evaluated on standard TREC test collections, show that our feature selection algorithm produces models that are either significantly more effective than, or equally effective as, models with manually selected features, such as those used in the past.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms, Experimentation, Theory

Keywords

Feature selection, parameter estimation, Markov random field model

1. INTRODUCTION

Developing effective retrieval models is a core problem in the field of information retrieval. Many different types of models have been proposed throughout the years, including Boolean, vector space, logic-based, probabilistic, and feature-based. One critical factor that must be considered when developing information retrieval models is the type of features to be used or modeled. Term frequency, inverse document frequency, document length, and term proximity are the fundamental features that are used in most of the modern information retrieval models including BM25 [22], language modeling [25], divergence from randomness [1], axiomatic approach to IR [8], and the Markov random field (MRF) model for IR [16].

However, most of these models make use of hand selected, probabilistically-inspired, or implicit features. Therefore, it is often difficult to adapt these types of models to new tasks, especially when the task has new, completely different types of features associated with it. Applying these models to new tasks typically requires an information retrieval expert to modify the underlying model in some way in order to properly account for the new types of features. This is a common theme in information retrieval modeling. Examples include incorporating PageRank as a prior into the BM25 model [5], allowing term proximity information as evidence in BM25 [4], modeling document structure in both language modeling and BM25 [19, 23], including term dependence in the DFR model [20], and allowing term associations in the axiomatic model [9]. These examples illustrate that incorporating new types of evidence and features into existing retrieval models is often non-trivial and can require significant amounts of human involvement.

Therefore, it is desirable for models to be flexible and robust enough to easily handle a wide range of features and provide a mechanism for automatically selecting relevant features. Then, given a large pool of candidate features, it would be possible to automatically learn the best model. Under this model learning paradigm, there would no longer be a need to manually tune or modify some existing retrieval model whenever a new task or data set is encountered. Instead, attention could be paid to developing a rich pool of features that are widely applicable.

We argue that the MRF model for information retrieval [16] and similar types of models, such as Gao et al.'s Linear Discriminant Model [12], are the correct types of models to use when model flexibility and robustness are important. These models provide a way of combining evidence from a wide range of textual features (e.g., term frequency, inverse document frequency, and term proximity measures) and non-textual features (e.g., PageRank, spam probability, and numeric attributes).

In this work, we propose an automatic, supervised feature

CIKM'07, November 6–8, 2007, Lisboa, Portugal.

selection algorithm that can be used in conjunction with any linear feature-based retrieval model, such as the MRF model [17]. The algorithm is novel, in that it is the first algorithm of its kind to be specifically designed for information retrieval tasks. The algorithm is also robust, since it can be applied to a wide range of feature sets, evaluation metrics, and methods for learning to rank. Lastly, but most importantly, the algorithm produces highly effective models. We show that models constructed using our algorithm are often significantly more effective than hand built models.

Although the primary contribution of this paper is our proposed feature selection algorithm, we also propose a canonical method for describing textual information retrieval features. The representation makes it easy to generate large sets of features that can be used with our algorithm.

The remainder of this paper is laid out as follows. Section 2 provides background material on the MRF model for IR and discusses previous feature selection work. In Section 3 we explain our novel method for representing features. Next, Section 4 describes our proposed feature selection approach. Then, in Section 5 we formally evaluate various aspects of our proposed algorithm and compare the effectiveness of the learned models to several other retrieval models. Finally, in Section 6 we summarize our contributions and outline potential directions for future work.

2. RELATED WORK

We now briefly introduce the MRF model for IR and describe previous work that is related to our feature selection algorithm.

2.1 Markov Random Field Model for IR

Recently, a new information retrieval model based on Markov random fields was proposed. The model goes beyond the bag of words assumption that underlies BM25 and (unigram) language modeling [22, 25]. The MRF model generalizes various dependence models that have been proposed in the past [16]. Most previous term dependence models have failed to show consistent, significant improvements over baseline bag of words model, with few exceptions [11]. The MRF model, however, has been shown to be highly effective across a variety of tasks, such as ad hoc retrieval [18], named-page finding [18], and Japanese web search [6].

Markov random fields are undirected graphical models. They provide a compact and flexible way of modeling joint distributions. The MRF model for IR models the joint distribution over a query $Q = q_1, \ldots, q_n$ and a document D. The underlying distribution over pairs of documents and queries is assumed to be a relevance distribution. That is, sampling from the distribution gives pairs of documents and queries, such that the document is relevant to the query.

A MRF is defined by a graph G and a set of non-negative potential functions over the cliques in G. The nodes in the graph represent the random variables and the edges define the independence semantics of the distribution. A MRF satisfies the Markov property, which states that a node is independent of all of its non-neighboring nodes given observed values for its neighbors.

Given a graph G, a set of potentials ψ_i, and a parameter vector Λ, the joint distribution over Q and D is given by:

$$P_{G,\Lambda}(Q, D) \quad = \quad \frac{1}{Z_\Lambda} \prod_{c \in C(G)} \psi(c; \Lambda)$$

where Z_Λ is a normalizing constant and $C(G)$ is the set of cliques in G. We follow common convention and parameterize the potentials as $\psi_i(c; \Lambda) = \exp[\lambda_i f_i(c)]$, where $f_i(c)$ is a real-valued feature function.

Under this parameterization, documents are ranked in descending order according to $P(D|Q)$, which can be shown to be rank equivalent to:

$$P(D|Q) \quad \stackrel{rank}{=} \quad \sum_{c \in C(G)} \lambda_c f_c(c)$$

Therefore, the ranking function is a weighted linear combination of features defined over the cliques of the MRF. The automatic feature selection algorithm proposed here associates new feature functions and weights (parameters) with cliques in G, which results in new $\lambda f(\cdot)$ components being added to the ranking function. Nothing about our selection algorithm is specific to the MRF model, and hence it can be applied to any ranking function that is a weighted linear combination of features.

2.2 Feature Selection and Combination

A number of feature selection techniques for random field models have been proposed in the machine learning literature [14, 21]. Our proposed algorithm is an adaptation of the feature induction technique proposed by Pietra et al. in [21]. Pietra et al. propose a greedy approach for adding induced features to the underlying model. During each iteration, the information gain for each induced feature is computed. The feature with the highest information gain is then added to the model and the entire model is retrained. Although we do not actually induce new features in our present work, we use a similar algorithm for selecting from a large pool of features. Another difference is that our algorithm scores each feature according to any information retrieval metric of interest. The feature that improves the metric the most is the one that is added to the model.

There has also been some information retrieval research into automatically learning ranking function using genetic programming [7]. These algorithms attempt to find a locally optimal ranking function by iteratively "evolving" a population of ranking functions using mutations and crossovers. Ranking functions are represented as arithmetic trees that consist of arithmetic operators and standard bag of words information retrieval features (e.g., term frequency, document length, etc.). The learned ranking functions have been shown to be significantly more effective than baseline ranking algorithms for several data sets [7].

Finally, result fusion techniques are another way of combining evidence from multiple types of features [2, 10]. If each individual feature is used as a ranking function, then data fusion techniques can be used to determine the best way to combine the rankings. However, using these techniques in this way does not directly address the feature selection problem, which is the primary focus of our work.

3. FEATURE REPRESENTATION

In this section we propose a novel method for describing textual information retrieval features. Textual features are defined to be any type of feature that can be extracted directly from text. Simple examples include the frequency of some term in a document and document length. More complex examples include the average positional distance

between two terms in a document and the BM25 weight of some term in a document. There currently exists no canonical way of representing rich sets of information retrieval features. In the remainder of this section we propose a canonical representation of textual feature representation that is compact, intuitive, and capable of representing a wide range of useful information retrieval features.

Since most information retrieval models are concerned with ranking documents in response to a query, we focus on textual features defined over query/document pairs. Thus, the input to our features is a query/document pair and the the output is a real value. Within our proposed representation, each feature is represented using a 3-tuple of the form (*dependence model type*, *clique set type*, *weighting function*). Each component of the tuple is described in detail below.

The input to our feature induction algorithm will be a set of 3-tuples, each of which represents a single feature. We note, however, that our algorithm does not require features to be represented this way. Instead, as we will show, this representation simply allows for large classes of features to be enumerated and explained compactly and easily.

3.1 Dependence Model Type

The first entry in the tuple is the *dependence model type*, which specifies the dependencies, if any, that are to be modeled between query terms. In the MRF model, dependencies are encoded by the edges in the graph. Different edge configurations correspond to different types of dependence assumptions.

In this work, we focus on three dependence model types. The three types are full independence (FI), sequential dependence (SD), and full dependence (FD). Examples of each type are given in Figure 1. We restrict ourselves to these three types because they have been used successfully in previous work and because they encompass the most common dependence assumptions used in information retrieval [16]. However, we note that query term dependencies can be inferred in other ways, such as constructing a dependence tree [27] or using the approach described by Gao et al. [11].

3.2 Clique Set Type

The second entry in the tuple, the *clique set type*, describes the set of cliques within the graph that the feature is to be applied to. Each feature is applied to one or more cliques within the graph. If it is applied to more than one clique, then all of the cliques that share the feature also share the weight for that feature. Therefore, clique sets act to tie parameters together, which is essential to avoid overfitting within the model.

In this work, we focus on three clique sets that have been found to be useful in previous work. These include:

- *single term* – set of cliques containing the document node and exactly one query term.
- *ordered terms* – set of cliques containing the document node and two or more query terms that appear in sequential order within the query.
- *unordered terms* – set of cliques containing the document node and two or more query terms that appear in any order within the query.

It should be noted that the unordered terms are a superset of the ordered terms and that the cliques that make up each set may change for different dependence model types. For example, the ordered terms and unordered terms clique sets are empty under the full independence assumption since that would result in a graph where there are no cliques with two or more query terms nodes. However, under the sequential dependence assumption, and with a query of length 2 or more, such cliques will exist and the ordered terms and unordered terms clique sets will not be empty.

If a clique set is empty, then its feature value is 0. However, if the clique set contains one or more cliques, then the feature value is the sum of the feature weights for each clique in the set. For example, given the query *new york city*, using the full independence model and the single term clique set, we would obtain a feature value of $f(new, D) + f(york, D) + f(city, D)$. Therefore, clique sets act to anonymize query terms. In this way, clique sets can be thought of as *feature types*. Thus, we have defined single term, ordered terms, and unordered terms feature types.

It is possible to define many different types of clique sets. For example, another clique set may be defined as "the clique that contains the first query term and the document node". Given enough training data, it may be possible to define such fine grained clique sets. However, given the limited amount of training data, we focus our attention on these coarse grained features.

Table 1 summarizes our discussion, provides example cliques for each clique set, and shows how a feature value would be computed for each.

3.3 Weighting Function

Finally, the third entry in the tuple is the *weighting function*, which describes how the feature values are computed. Going back to the *new york city* example, the weighting function describes how $f(new, D)$, $f(york, D)$, and $f(city, D)$ are computed.

In this work, we define weighting functions that can be used with our clique sets. We explore weighting functions based on language modeling estimates (Dirichlet smoothing [28]) and the popular BM25 weighting model. It is straightforward to use the standard forms of these weighting functions for the single term clique set. However, we must define how to match the query terms within documents when applying these weighting functions to the ordered terms clique set and the unordered terms clique set.

For the ordered term case, we match terms in documents using the Inquery ordered window operator (`#`M), where the parameter M determines how many non-matching terms are allowed to appear between matched terms. This rewards documents for preserving the order that the query terms occur in.

In the unordered term case we match terms using the Inquery unordered window operator (`#uw`N), where N defines the maximum size of the window that the terms may occur (ordered or unordered) in. Here, we reward documents in which subsets of query terms occur within close proximity of each other. For more details on the matching semantics of these operators, please refer to [15].

Table 2 summarizes the weighting functions used in this work. Of course, many different types of weighting functions could easily be used within the model. If new, more effective term weighting functions are developed in the future, then they can be easily used instead of, or in addition to, the Dirichlet or BM25 weights.

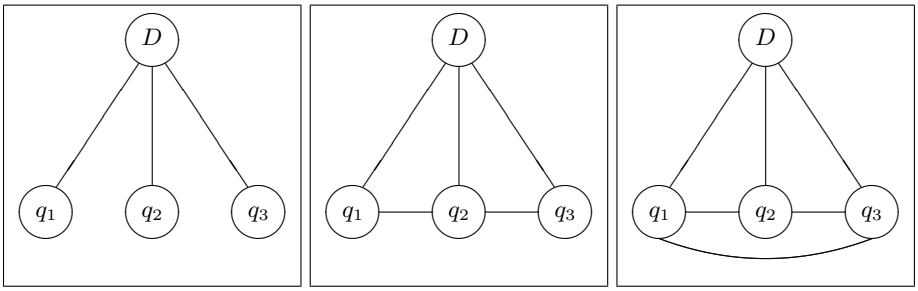


Figure 1: Example Markov random field model for three query terms under various independence assumptions. (left) full independence, (middle) sequential dependence, (right) full dependence.

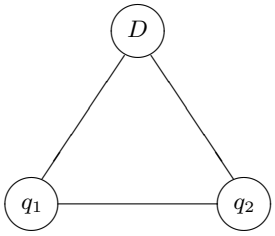
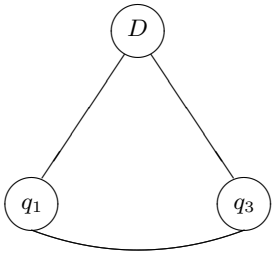
Clique Set	Example Clique	Feature Value
single term clique set		$f(q_1, D) + f(q_2, D) + f(q_3, D)$
ordered terms clique set		$f(q_1, q_2, D) + f(q_2, q_3, D) + f(q_1, q_2, q_3, D)$
unordered terms clique set		$f(q_1, q_2, D) + f(q_2, q_3, D) + f(q_1, q_3, D) + f(q_1, q_2, q_3, D)$

Table 1: Illustration of clique sets for query *q1 q2 q3* under full dependence assumption. For each clique set we provide an example clique within the set and show how the value of a feature applied to the clique set is computed.

Weighting Function	Feature Value
LM	$f_{LM,T}(q_i, D) = \log \left[\frac{tf_{q_i,D} + \mu \frac{cf_{q_i}}{\lvert C \rvert}}{\lvert D \rvert + \mu} \right]$
LM-O-M	$f_{LM,U,M}(q_1, \ldots, q_k, D) = \log \left[\frac{tf_{\#M(q_1 \ldots q_k),D} + \mu \frac{cf_{\#M(q_1 \ldots q_k)}}{\lvert C \rvert}}{\lvert D \rvert + \mu} \right]$
LM-U-N	$f_{LM,O,N}(q_1, \ldots, q_k, D) = \log \left[\frac{tf_{\#uwNk(q_1 \ldots q_k),D} + \mu \frac{cf_{\#uwNk(q_1 \ldots q_k)}}{\lvert C \rvert}}{\lvert D \rvert + \mu} \right]$
BM25	$f_{T,BM25}(q_i, D) = \frac{(k_1+1)tf_{w,D}}{k_1\left((1-b)+b\frac{\lvert D \rvert}{\lvert D \rvert_{avg}}\right)+tf_{w,D}} \log \frac{N - df_w + 0.5}{df_w + 0.5}$
BM25-O-M	$f_{BM25,O,M}(q_1, \ldots, q_k, D) = \frac{(k_1+1)tf_{\#M(q_1 \ldots q_k),D}}{k_1\left((1-b)+b\frac{\lvert D \rvert}{\lvert D \rvert_{avg}}\right)+tf_{\#M(q_1 \ldots q_k),D}} \log \frac{N - df_{\#M(q_1 \ldots q_k)} + 0.5}{df_{\#M(q_1 \ldots q_k)} + 0.5}$
BM25-U-N	$f_{BM25,U,N}(q_1, \ldots, q_k, D) = \frac{(k_1+1)tf_{\#uwNk(q_1 \ldots q_k),D}}{k_1\left((1-b)+b\frac{\lvert D \rvert}{\lvert D \rvert_{avg}}\right)+tf_{\#uwNk(q_1 \ldots q_k),D}} \log \frac{N - df_{\#uwNk(q_1 \ldots q_k)} + 0.5}{df_{\#uwNk(q_1 \ldots q_k)} + 0.5}$

Table 2: Summary of Dirichlet and BM25 weighting functions. Here, M and N act as weighting function parameters that affect how matching is done, $tf_{e,D}$ is the number of times expression e matches in document D, $cf_{e,D}$ is the number of times expression e matches in the entire collection, df_e is the total number of documents that have at least one match for expression e, $|D|$ is the length of document D, $|D|_{avg}$ is the average document length, N is the number of documents in the collection, and $|C|$ is the total length of the collection. Finally, μ, k_1, and b are weighting function hyperparameters.

3.4 Examples

Now that we have described each element that makes up the 3-tuple feature representation, we now give a set of examples in order to clarify the previous discussion and make it more concrete. For these examples, we continue to use the query *new york city*.

The first example feature is (FI, single term, BM25). This makes use of the full independence variant, the single term clique set, and the BM25 weighting function. The resulting feature value is then:

$$f_{BM25,T}(new, D) + f_{BM25,T}(york, D) + f_{BM25,T}(city, D)$$

where $f_{BM25,T}$ takes on the BM25 form as given in Table 2. We see that this feature value is simply the BM25 score of query for document D.

The next example feature is (SD, ordered terms, LM-O-4), which uses the sequential dependence variant, the ordered terms clique set, and a Dirichlet weighting function. The resulting feature value is then:

$$f_{LM,O,4}(new, york, D) + f_{LM,O,4}(york, city, D)$$

where $f_{LM,O,4}$ takes on the Dirichlet form and M, the ordered window size, is set to 4.

The feature (SD, unordered terms, LM-U-32) is computed in a similar fashion, except the Dirichlet form of $f_{LM,U,32}$ is used and N, the unordered window size, is set to 32. As these examples illustrate, this canonical form allows us to compactly define a large, rich set of feature functions.

4. AUTOMATIC FEATURE SELECTION

4.1 Overview

As we described before, feature selection techniques are commonly used in the machine learning community. In this section we propose a feature selection algorithm for information retrieval. Feature selection is important for a number of reasons. First, it provides a general, robust way of building models when there is little *a priori* knowledge about the types of features that may be important for a given task or data set. By using a feature selection algorithm, the model designer can focus less on building the best model and can instead focus on designing good features. Second, feature selection can reduce the number of noisy or redundant features in a large feature set. Such features may reduce training efficiency and may result in a model that contains a number of non-identifiable parameters. Non-identifiable parameters are those that cannot be reasonably estimated given the training data. This often results from having redundant or highly correlated features. Feature selection helps overcome the problems associated with non-identifiable parameters. Finally, feature selection can provide insights into the important features for a given task or data set. By inspecting the order in which features are selected, we can often learn what characteristics of a given task are the most important or the most exploitable. This knowledge can then be used by the feature engineer to construct better features.

4.2 Algorithm

We now describe our automatic feature selection algorithm. While our discussion will focus on how the algorithm can be applied to the MRF model for IR, it should be noted that it can also be applied to a variety of other models. In particular, it is well suited for linear feature-based models [17].

Let M_t denote the model learned after iteration t. Features are denoted by f and the weight (parameter) associated with feature f is denoted by λ_f. The candidate set of features is denoted by $\mathcal{F}$. The entire set of feature weights for a model is denoted by Λ. A model, then, is represented as set of feature/weight pairs. Finally, we assume that $SCORE(M)$ returns the utility or 'goodness' of model M with respect to some training data. The utility function and the form of the training data largely depends on the underlying task. For example, for *ad hoc* retrieval, it is likely that $SCORE(\cdot)$ would return the mean average precision of using model M against some set of training data, such as TREC topics and relevance judgments. For a homepage

finding task, $SCORE(\cdot)$ might be another metric, such as mean reciprocal rank. The important thing to note here is that any utility function, regardless of whether or not it is differentiable with respect to the model parameters, can be used. Therefore, the ultimate goal of our feature selection algorithm is to select features and set feature weights in such a manner as to maximize the metric imposed by $SCORE(\cdot)$.

The algorithm begins with an empty model (i.e., $M_0 = \{\}$). Then, we temporarily add a feature f to the model. We then hold all weights except λ_f fixed and find the setting for λ_f that maximizes the utility of the augmented model. This step can be done using any number of learning to rank techniques [3, 13, 17]. The utility of feature f ($SCORE_f$) is defined to be the maximum utility obtained during training. The feature's utility measures how good the current model would be if the feature were added to it. This process is repeated for every $f \in \mathcal{F}$, resulting in a utility being computed for every feature in the candidate pool. The feature with the maximum utility is then added to the model and removed from $\mathcal{F}$. After the new feature is added, we can, optionally, retrain the entire set of weights. The entire process is then repeated until either some fixed number of features have been added to the model or until the change in utility between consecutive iterations drops below some threshold. See Algorithm 1 for this algorithm written in pseudo-code.

Note that our algorithm is not guaranteed to find the global maximum for $SCORE(M)$. Instead, we are only guaranteed to find a local maxima. Many factors, including properties of $SCORE(M)$, the number of features used, and the properties of the feature used, will affect the quality of the learned model.

Algorithm 1 Feature selection algorithm.

1: $t \leftarrow 0$
2: $M_t \leftarrow \{\}$
3: **while** $SCORE(M_t) - SCORE(M_{t-1}) > \epsilon$ **do**
4: **for** $f \in \mathcal{F}$ **do**
5: $\hat{\lambda}_f \leftarrow \arg\max_{\lambda_f} SCORE(M \cup \{(f, \lambda_f)\})$
6: $SCORE_f \leftarrow SCORE(M \cup \left\{(f, \hat{\lambda}_f)\right\})$
7: **end for**
8: $f^* \leftarrow \arg\max_f SCORE_f$
9: $M \leftarrow M \cup \left\{(f^*, \hat{\lambda}_{f^*})\right\}$
10: $\Lambda \leftarrow \arg\max_{\Lambda} SCORE(M)$ (optional)
11: $\mathcal{F} \leftarrow \mathcal{F} - \{f^*\}$
12: $t \leftarrow t + 1$
13: **end while**

Although we do not formally evaluate the efficiency of our proposed algorithm, we note that our algorithm, without retraining, only requires a linear (in the size of $\mathcal{F}$) number of single parameter training steps each iteration. If we are to select k features, such that $k << |\mathcal{F}|$, then, depending on the time complexity of the training algorithm, it is likely that training using our algorithm will be more more computationally efficient than training a monolithic model with $|\mathcal{F}|$ features.

5. EVALUATION

In this section we experimentally evaluate various aspects of our proposed feature selection algorithm.

Name	Description	# Docs	Train Topics	Test Topics
WSJ	Wall St. Journal 87-92	173,252	51–150	151–200
AP	Assoc. Press 88-90	242,918	51–150	151–200
ROBUST	Robust 2004 data	528,155	301–450	601–700
WT10g	TREC Web collection	1,692,096	451–500	501–550
GOV2	2004 crawl of .gov domain	25,205,179	701–750	751–800

Table 3: Overview of TREC collections and topics.

5.1 Setup

In order to investigate the strengths and weaknesses of our proposed feature selection algorithm, we evaluate its effectiveness on a wide range of *ad hoc* retrieval data sets. Table 3 summarizes the TREC data sets used in our experiments. The WSJ, AP, and ROBUST collections are smaller and consist entirely of newswire articles. The WT10g and GOV2 are large web collections. The topics for each data set are split into a training and test set.

Our experiments were done using Searching using Markov Random Fields (SMRF), which is a new Java-based toolkit that sits on top Indri [26] and provides a robust framework for experimenting using the MRF model. All collections were stopped using a standard list of 418 common terms and stemmed using a Porter stemmer. Only the title portion of the TREC topics are used to construct queries. Our primary evaluation metric is mean average precision. Statistical significance is determined using a one-tailed paired $t-test$ evaluated at the $p < 0.05$ level.

We now describe our feature candidate pool in terms of the feature representation scheme we proposed in Section 3. For dependence model type, the features may be either *FI* (full independence), *SD* (sequential dependence), or *FD* (full dependence). The clique set type may either be *single term*, *ordered terms*, or *unordered terms*. The weighting functions include LM, BM25, [LM, BM25]-O-[1, 2, 4, 8, 16, or 32], and [LM, BM25]-U-[1, 2, 4, 8, 16, 32, or unlimited]. As we see, the pool is very robust and covers many different types of important features. It includes features that span all three type of dependence, use all three types of clique sets, allow both Dirichlet or BM25 weighting, and vary the window sizes for the ordered and unordered window matchings across a range of values. After removing trivial and duplicate features, our candidate pool consists of 48 features.

For all of our experiments, features are added until there is no change in training set mean average precision between iterations (i.e., $\epsilon = 0$) or until we have added 5 features. Preliminary experiments showed that adding more than 5 features never resulted in significantly different training or test rest results. Therefore, for efficiency reasons, we chose to stop after 5 features have been added to the model.

5.2 No Retraining vs. Retraining

We wish to analyze what effect, if any, retraining (see Algorithm 1, line 10) has on training and generalization properties of the model. Table 4 summarizes the mean average precision obtained on the training and test set when retraining is used and when it is not.

	No Retrain		Retrain	
	Train	Test	Train	Test
AP	0.1863	0.2266	0.1865	0.2246
WSJ	0.2700	0.3553	0.2703	0.3543
ROBUST	0.2387	0.3079	0.2391	0.3065
WT10G	0.2344	0.2129	0.2357	0.2140

Table 4: Training and test set mean average precision values for no retraining and retraining.

We first investigate whether or not the models learned with retraining vary significantly from those learned without retraining. As Table 4 shows, the training set mean average precision values for no retraining and retraining are nearly equivalent for every data set. In fact, the differences are statistically indistinguishable. In addition, we discovered that the same set of features were added regardless of whether or not retraining was done or not. Therefore, it appears as though retraining has little effect on the learned model, both in terms of the features selected and the training set mean average precision.

Next, we study the effect of retraining on the generalization properties of the model. As the test set results in Table 4 show, there is very little difference in mean average precision for no retraining versus retraining. The results, again, are statistically indistinguishable for every data set. Hence, retraining does not significantly affect how well the model generalizes to unseen data.

Therefore, given that retraining requires more computational power, has no effect on either the learned model or the generalization properties of the model, we conclude that there is no need to retrain the model each iteration.

5.3 Number of Features

We now analyze how sensitive the models are to the number of parameters, both in terms of potential overfitting, and in terms of test set effectiveness.

Figure 2 plots the training and test set mean average precision versus the number of features that have been added to the model. As the figure indicates, there appears to be little, if any overfitting happening. The test set mean average precision never significantly drops as more features are added to the model.

5.4 Analysis

The greedy nature of our feature selection algorithm provides us with a mechanism for analyzing the importance of different types of features across tasks and data sets. By looking at the order in which features are selected, and the weight assigned to each, we can develop deeper insights into the role that features play for a given task and/or data set.

For example, for the WT10G data set, with no retraining, the features are selected in the following order:

(FI, single term, BM25) :	0.8138
(FD, unordered terms, LM-U-8) :	0.0001
(SD, unordered terms, BM25-U-unlimited) :	0.0090
(FD, unordered terms, BM25-U-8) :	0.1575
(SD, ordered terms, BM25-O-8) :	0.0196

where the numbers after the colons are the weights assigned to each feature in the final model.

As Figure 2 shows, there is a large increase in both training and test set mean average precision after the second feature is added to model. This large increase, which is also exhibited for the GOV2 data set, reiterates the importance of term proximity models for large web collections [16]. We see that simply adding a single proximity feature increases mean average precision substantially. However, there is a much smaller effect observed after further term proximity / dependence model features are added to the model.

To provide a different example, we consider the order in which features were selected for the WSJ collection. The features selected, in order, are:

(FI, single term, BM25) :	0.5864
(SD, unordered terms, BM25-U-1) :	0.3746
(FD, unordered terms, BM25-U-32) :	0.0193
(FI, single term, LM) :	0.0196
(FD, unordered terms, BM25-U-unlimited) :	0.0001

As with the WT10G model, the first feature selected is the full independence, single term, BM25 feature. In fact, this feature was the first selected for every data set. This is not surprising, however, since the overwhelming importance of single term features has long been understood. In fact, some have argued that it is not possible to do much better than simply using single term identifiers [24].

However, no other strong regularities were observed across data sets. This indicates that the each data set has unique characteristics that make certain features more discriminative than others. Such characteristics may include things like query length, noise, document length distribution, and properties of the underlying vocabulary. This suggests that no single model, with a fixed feature set and fixed feature weights, can be applied to every possible task and data set. Instead, adaptive models and techniques, such as the one presented in this paper, can provide a means for automatically and robustly learning the best set of features to use on a task-by-task basis.

5.5 Summary of Results

Finally, we compare the retrieval effectiveness of the models automatically learned using our feature selection algorithm (MRF-FS) with several other retrieval models, including language modeling (Dirichlet smoothing), BM25, and two MRF models with hand selected features (MRF-LM and MRF-BM25) that have been shown to be highly effective in the past. For each model, parameters are tuned on the training set to maximize mean average precision. Therefore, every model is properly trained in accordance with the same evaluation metric.

The MRF-LM model uses three manually chosen features. These features are:

(FI, single term, LM)
(FD, ordered terms, LM-O-1)
(FD, unordered terms, LM-U-4)

These are the same features that are used in [16] under the full dependence model variant. The MRF-BM25 model uses the same exact set of features, except BM25 weighting is used in place of language modeling weighting. These hand selected features have been shown to be highly effective for the *ad hoc* retrieval task in the past, consistently producing

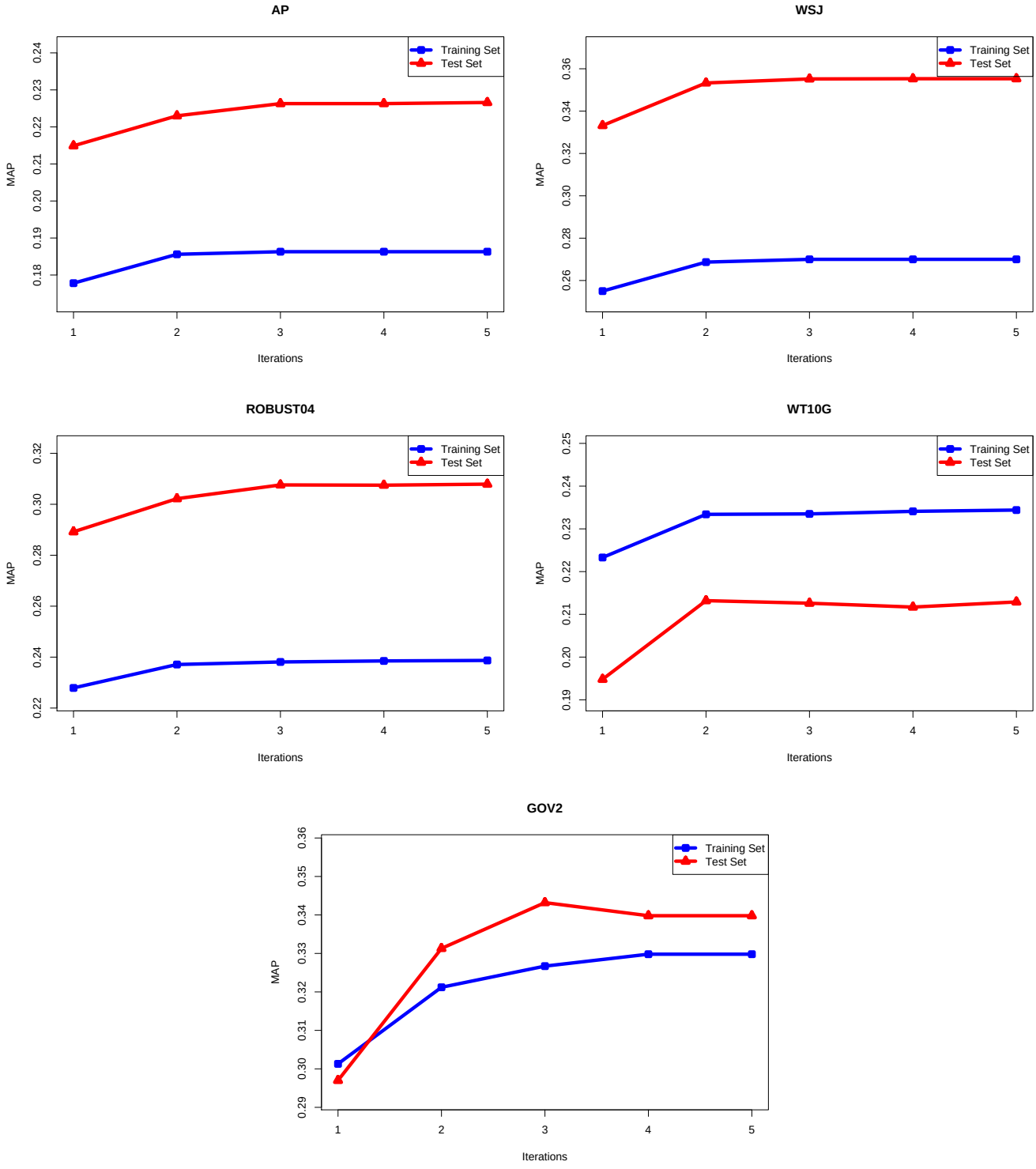


Figure 2: Mean average precision versus number of iterations for the training and test sets of the AP, WSJ, ROBUST, WT10G, and GOV2 data sets.

	LM	BM25	MRF-LM	MRF-BM25	MRF-FS
AP	0.2077	0.2149	0.2147	0.2210	0.2266
WSJ	0.3258	0.3332	0.3425	0.3512	0.3553
ROBUST	0.2920	0.2892	0.3092	0.3101	0.3079
WT10G	0.1861	0.1948	0.2140	0.2129	0.2129
GOV2	0.2984	0.2970	0.3325	0.3476	0.3500

Table 5: Comparison of test set mean average precision for language modeling (LM), BM25, MRF model using language modeling weighting (MRF-LM), MRF model using BM25 weighting, and MRF learned using our proposed feature selection algorithm (MRF-FS).

statistically significant better effectiveness over bag of words models [16, 18]. This allows us to compare models that are automatically learned using our feature selection algorithm with models that use manually chosen features and have been proven to be highly effective.

Our results, which are summarized in Table 5, support previous observations that show that using MRF models with hand chosen features are generally more effective than bag of words models for *ad hoc* retrieval. However, we are interesting in how effective the automatically learned models are. For both the AP and WSJ data sets, the mean average precision of the automatically learned model is statistically significantly better than all of the other models, including the MRF models with manually chosen features. The improvement in mean average precision over BM25 for the AP data set is 5.4% and 6.6% on the WSJ data set.

On the ROBUST, WT10G, and GOV2 data sets, the automatically learned models are statistically significantly better than language modeling and BM25, but statistically indistinguishable from the two models with hand selected features. Despite the lack of a statistically significant improvement over the models with hand selected features, the results still provide evidence that the learned model is highly effective. Indeed, compared to BM25, the automatically learned model is 6.5% better for ROBUST, 9.3% better for WT10G, and 17.8% better for GOV2.

Therefore, our results show that our proposed feature selection algorithm produces very effective models that are competitive with, and often significantly better than models with hand selected features.

6. CONCLUSIONS AND FUTURE WORK

In this work, we presented an automatic feature selection algorithm for information retrieval models. While the algorithm was presented in terms of the MRF model for IR, it can also be applied to any linear feature-based model. The algorithm shifts the focus of information retrieval practitioners from manual feature selection/combination to feature engineering.

In addition, we presented a novel, robust scheme for representing textual features. Under the scheme, features are represented as 3-tuples that consist of dependence model type, clique set type, and a weighting function. This scheme can be used to represent a wide variety of useful information retrieval features. We used the scheme in conjunction with our feature selection algorithm to enumerate a large pool of candidate features.

We also evaluated the effectiveness of the automatically learned models. We showed that there is little benefit to retraining the entire model after each iteration and that there were few signs of overfitting when analyzing the learning curves. We also investigated the order in which features are added to the model and showed that single term features, as expected, are always added first, and that various term proximity features are then added, depending on various characteristics of the data set. Finally, we showed that the automatically learned models are always statistically significantly better than language modeling and BM25 and, for two data sets, statistically significantly better than a model with carefully hand selected features. Overall, our results prove the robustness and effectiveness of our proposed algorithm.

In terms of future work, it would be interesting to combine our feature selection approach with the genetic programming approaches described in Section 2. Genetic programming can be used to automatically induce new features that can then be automatically selected into a model using our algorithm. It would also be interesting to apply the proposed algorithm to other tasks, such as web search, blog search, or XML retrieval.

Acknowledgments

This work was supported in part by the Center for Intelligent Information Retrieval and in part by Microsoft Live Labs. Any opinions, findings and conclusions or recommendations expressed in this material are the author's and do not necessarily reflect those of the sponsor.

7. REFERENCES

[1] G. Amati and C. J. van Rijsbergen. Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Transactions on Information Systems*, 20(4):357–389, 2002.

[2] B. T. Bartell, G. W. Cottrell, and R. K. Belew. Automatic combination of multiple ranked retrieval systems. In *Proc. 17th Ann. Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 173–181, 1994.

[3] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proc. of the International Conf. on Machine Learning*, pages 89–96, 2005.

[4] S. Büttcher, C. L. A. Clarke, and B. Lushman. Term proximity scoring for ad-hoc retrieval on very large text collections. In *Proc. 29th Ann. Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 621–622, 2006.

[5] N. Craswell, S. Robertson, H. Zaragoza, and M. Taylor. Relevance weighting for query independent evidence. In *Proc. 28th Ann. Intl. ACM SIGIR Conf.*

on Research and Development in Information Retrieval, pages 416–423, 2005.

[6] K. Eguchi. NTCIR-5 query expansion experiments using term dependence models. In *Proc. of the Fifth NTCIR Workshop Meeting on Evaluation of Information Access Technologies*, pages 494–501, 2005.

[7] W. Fan, M. D. Gordon, and P. Pathak. A generic ranking function discovery framework by genetic programming for information retrieval. *Inf. Process. Manage.*, 40(4):587–602, 2004.

[8] H. Fang and C. Zhai. An exploration of axiomatic approaches to information retrieval. In *Proc. 28th Ann. Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 480–487, 2005.

[9] H. Fang and C. Zhai. Semantic term matching in axiomatic approaches to information retrieval. In *Proc. 29th Ann. Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 115–122, 2006.

[10] E. A. Fox and J. A. Shaw. Combination of multiple searches. In *Proc. 2nd Text REtrieval Conference*, pages 243–252, 1993.

[11] J. Gao, J. Nie, G. Wu, and G. Cao. Dependence language model for information retrieval. In *Proc. 27th Ann. Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 170–177, 2004.

[12] J. Gao, H. Qi, X. Xia, and J. Nie. Linear discriminant model for information retrieval. In *Proc. 28th Ann. Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 290–297, 2005.

[13] T. Joachims. A support vector method for multivariate performance measures. In *Proc. of the International Conf. on Machine Learning*, pages 377–384, 2005.

[14] A. McCallum. Efficiently inducing features of conditional random fields. In *Proc. 19th Conf. on Uncertainty in Artificial Intelligence*, 2003.

[15] D. Metzler and W. B. Croft. Combining the language model and inference network approaches to retrieval. *Information Processing and Management*, 40(5):735–750, 2004.

[16] D. Metzler and W. B. Croft. A Markov random field model for term dependencies. In *Proc. 28th Ann. Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 472–479, 2005.

[17] D. Metzler and W. B. Croft. Linear feature based models for information retrieval. *Information Retrieval*, to appear, 2006.

[18] D. Metzler, T. Strohman, Y. Zhou, and W. B. Croft. Indri at terabyte track 2005. In *Online proceedings of the 2005 Text Retrieval Conf.*, 2005.

[19] P. Ogilvie and J. Callan. Combining document representations for known-item search. In *Proc. 26th Ann. Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 143–150, 2003.

[20] J. Peng, C. Macdonald, B. He, V. Plachouras, and I. Ounis. Incorporating term dependency in the dfr framework. In *Proc 30th Ann. Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, page To appear, 2007.

[21] S. D. Pietra, V. D. Pietra, and J. Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393, 1997.

[22] S. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proc. 17th Ann. Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 232–241, 1994.

[23] S. Robertson, H. Zaragoza, and M. Taylor. Simple bm25 extension to multiple weighted fields. In *Proc. 13th Intl. Conf. on Information and Knowledge Management*, pages 42–49, 2004.

[24] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[25] F. Song and W. B. Croft. A general language model for information retrieval. In *Proc. 8th Intl. Conf. on Information and Knowledge Management*, pages 316–321, 1999.

[26] T. Strohman, D. Metzler, H. Turtle, and W. B. Croft. Indri: A language model-based serach engine for complex queries. In *Proc. of the International Conf. on Intelligence Analysis*, 2004.

[27] C. J. van Rijsbergen. A theoretical basis for the use of cooccurrence data in information retrieval. *Journal of Documentation*, 33(2):106–119, 1977.

[28] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proc. 24th Ann. Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 334–342, 2001.