Machine Learned Sentence Selection Strategies for Query-Biased Summarization

Donald Metzler
metzler@yahoo-inc.com
Yahoo! Research
2821 Mission College Blvd.
Santa Clara, CA 95054

Tapas Kanungo
kanungo@yahoo-inc.com
Yahoo! Labs
2821 Mission College Blvd.
Santa Clara, CA 95054

ABSTRACT

It has become standard for search engines to augment result lists with document summaries. Each document summary consists of a title, abstract, and a URL. In this work, we focus on the task of selecting relevant sentences for inclusion in the abstract. In particular, we investigate how machine learning-based approaches can effectively be applied to the problem. We analyze and evaluate several learning to rank approaches, such as ranking support vector machines (SVMs), support vector regression (SVR), and gradient boosted decision trees (GBDTs). Our work is the first to evaluate SVR and GBDTs for the sentence selection task. Using standard TREC test collections, we rigorously evaluate various aspects of the sentence selection problem. Our results show that the effectiveness of the machine learning approaches varies across collections with different characteristics. Furthermore, the results show that GBDTs provide a robust and powerful framework for the sentence selection task and significantly outperform SVR and ranking SVMs on several data sets.

Categories and Subject Descriptors

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

General Terms

Algorithms, Experimentation, Theory

Keywords

sentence selection, learning to rank, gradient boosted decision trees

1. INTRODUCTION

Search engines have become popular and are widely used for many different tasks, such as web search, desktop search, enterprise search, and various domain-specific verticals. It has become almost standard for search engines to augment result lists with document summaries. It is important to produce high quality summaries, since the summaries can bias the perceived relevance of a document. For example, if the summary for a highly relevant document is poorly constructed, the user may perceive the document as non-relevant and may never view the document. Since users implicitly infer relevance from these summaries, it is important to construct high quality summaries that align the user's perceived relevance of the document with the actual relevance of the document.

Document summaries can either be query independent [8, 15] or query dependent [26, 29]. A query independent summary conveys general information about the document, and typically includes a title, static abstract, and URL, if applicable. Here, titles and static abstracts can either be extracted from the document, manually constructed, or automatically generated. These types of summaries can be computed offline and cached for fast access. The main problem with query independent summaries is that the summary for a document never changes across queries. This is one of the problems that query dependent summarization algorithms attempt to address, by biasing the summary towards the query. These summaries typically consist of a title, dynamic abstract, and URL. Since these summaries are dynamically generated, they are typically constructed at query time.

In this paper, we focus on the task of automatically generating abstracts for query dependent summarization. Given a query and a document, a query dependent abstract is generated as follows. First, relevant (with respect to the query) sentences or passages within the document must be identified. This is referred to as the *sentence selection* problem. After the relevant sentences have been identified, the *composition* phase begins. When composing an abstract, it is important to take into account how many sentences to include, and how to compress the sentences to fit within a fixed bounding box [14]. Furthermore, notions of readability and novelty also play a role during composition. Since the composition process can quickly become overly complex due factors involving presentation, user interaction, and novelty, we focus on the sentence selection problem in the remainder of this paper and leave composition as future work.

We propose using machine learning techniques to solve the sentence selection problem. There are several benefits to using such techniques. First, they provide an easy means of incorporating a wide range of features. It is often difficult

SIGIR 2008 LR4IR Workshop, Singapore.

to incorporate arbitrary features into standard information retrieval models, such as language modeling and BM25. Second, depending on the machine learning technique used, the model can be learned over a rich function space. Manually constructing such a function would require a great deal of effort. Finally, machine learning techniques provide a mechanism for learning from explicit (e.g., human judgments) or implicit (e.g., click data) training data. Of course, this may also be one of the biggest disadvantages to using machine learning, as well, as it is often difficult or expensive to obtain training data. All of the techniques we explore are fully supervised, and therefore require some form of training data.

A recent study has shown the feasibility of using machine learning approaches for sentence selection [29]. Wang *et al.* showed that ranking support vector machines (SVMs) outperform SVM classifiers and BM25 on a very small test collection of only 10 queries. In this paper, we augment the observations presented by Wang *et al.* and undertake a more comprehensive view of the problem from multiple perspectives.

Our work has four key contributions. First, we propose using regression-based models, such as support vector regression (SVR) and gradient boosted decision trees (GBDTs) for the sentence selection problem. Regression models, and GBDTs, in particular, have recently been shown to be highly effective for learning ranking functions for web search [17, 32]. We hypothesize the same will be true for sentence selection. Second, we carry out a rigorous set of experiments over three TREC data sets that, combined, have 200 queries associated with them. The results of these experiments provide unique insights into the applicability of the various learning techniques to data sets with different characteristics. Third, we show that performance is quite sensitive to how sentences are actually selected by comparing and contrasting the effectiveness of retrieving a fixed number of sentences per query/document pair versus using a global score threshold. While this topic is often ignored, it is important when using these algorithms in practice. Finally, we plan to release our data set for possible inclusion in the LETOR benchmark suite. This would allow researchers to explore various aspects of learning to rank in the context of an interesting application that has many unique characteristics that differentiates it from *ad hoc* retrieval and web search.

The remainder of this paper is laid out as follows. First, in Section 2, we detail related work in both summarization and machine learning approaches to ranking. Then, in Section 3, we describe the three machine learning models used, the features used with the models, and the different strategies for choosing the number of sentences to select. In Section 4 we describe our experimental evaluation. Section 5 discusses miscellaneous sentence selection issues. Finally, in Section 6, we conclude and describe possible areas of future work.

2. RELATED WORK

Automatic text summarization has been explored in many research areas including artificial intelligence, natural language processing, and information retrieval [21, 19]. While research in AI and NLP has focused on analyzing well-written text and generating large summaries (5-10 sentences), web search and information retrieval has focused on generating very small summaries. In fact, in web search, the role of the summary is to give an idea to the user whether or not the destination page is relevant for the user's query. Since a search result page typically has 10 or more URLs, the summary associated with an individual URL can not exceed more than 2-3 lines.

Kupiec, Pedersen and Chen [15] first addressed the sentence selection problem by using a binary Naïve Bayes classifier that learned if a given sentence should be part of the summary or not. The features used within the model were query independent, and therefore the goal of the model was to generate static abstracts. The model was trained using a corpus where human judges selected sentences that they thought should be part of the summary.

Tombros and Sanderson [26] conducted user studies using query-biased summaries. Their experiments suggest that query-biased summaries improve the ability of users to accurately judge relevance. Clarke *et al.* [3] studied the correlation of various attributes of summaries with click behavior. Goldstein *et al.* proposed various features and scored sentences in newspaper articles according to a specific scoring function. The function itself was not learned, however. In addition, Turpin *et al.* recently described techniques for efficiently compressing summaries [27]. However, this work does not take quality/relevance of the summary into account, which is the primary focus of our work.

More recently, initiatives, such as the Document Understanding Conference (DUC) and the Text Retrieval Conference (TREC) have conducted quantitative evaluations of various summarization algorithms and sentence retrieval tasks. In particular, the TREC Novelty Track, which ran from 2002 to 2004 included a sentence retrieval sub-task that required participants to retrieve relevant sentences, rather than relevant documents. A majority of the groups participating used standard information retrieval models for the task, such as language modeling and BM25. It is important to note that sentence selection is very closely related to sentence retrieval. The primary difference is that in sentence retrieval, a ranked list of sentences is returned for a *set* of documents, whereas the sentence selection task only returns a ranked list of sentences for a single document. Since the two tasks are so similar, it is likely that techniques developed for sentence retrieval will also work well for sentence selection, and vice versa.

The problem of learning to rank for information retrieval has become a topic of great interest in recent years. Many different techniques have been proposed, including logistic regression [7], SVMs [2, 12, 20], neural networks [1], and perceptrons [6]. These techniques have been adapted to optimize information retrieval-specific metrics, such as precision at K [13], mean average precision [30], nDCG [16], among others. While benchmark data sets exist for *ad hoc* and web retrieval [18], none currently exist for sentence selection.

Regression-based models, and in particular, gradient boosted decision trees, have recently been explored for learning to rank and and have been shown to be highly effective for web search [32, 17]. In this work, we apply regression-based techniques to the sentence selection problem, which has very different characteristics than web search, both in terms of features and in terms of context.

The work done by Wang *et al.* [29] is the most closely related to ours. The authors propose using SVMs and ranking SVMs to model the relevance of sentences to queries. Their results show that ranking SVMs outperformed standard SVMs on a small test collection of 10 queries. In their experiments, they do not have a methodology for selecting the number of sentences. Instead, they always retrieve three sentences per document. In our work, we use ranking SVMs as a baseline against which we compare regression-based models, such as SVR and GBDTs. In addition, we analyze two different strategies for choosing the number of sentences to retrieve and carry out experiments out on three different test collections with over 200 queries, which allows us to draw inferences about the influence of various data set characteristics on effectiveness.

3. SENTENCE SELECTION USING MACHINE LEARNING

In this section, we describe the three machine learning techniques that we use for sentence selection, the set of features that we consider, and strategies for automatically choosing the number of sentences to return.

3.1 Models

There are numerous approaches to estimating the value of a categorical or continuous *response* variable (the human judgments) from measurements of *explanatory* variables (the extracted features). This problem has been studied under the names of statistical inference [28], pattern recognition [11] and more recently statistical machine learning [10]. Logistic regression, support vector machines, neural networks, and decision trees are some of the popular techniques. In this section, we briefly describe the three machine learning algorithms that we use for sentence selection.

3.1.1 Ranking SVMs

Ranking SVMs are a generalization of the classical SVM formulation that learns over *pairwise preferences*, rather than binary labeled data [12]. The motivation behind ranking SVMs is that for ranking problems, it is inappropriate to learn a classification model, since it does not take the structure of the problem into account. Instead, pairwise preferences can implicitly encode the structure of ranking problems, and therefore learning an SVM over such pairwise preferences is typically more effective when used for ranking since its objective function tends to be more in line with standard information retrieval metrics, such as precision, mean average precision, and F1.

Formally, the ranking SVM is formulated as a quadratic programming problem that has the following form:

$$\begin{array}{lll} \min & \frac{1}{2}\|w\|^2 + C\sum_{i,j}\xi_{i,j} & \\ s.t. & (w\cdot x_i - w\cdot x_j) \geq 1 - \xi_{i,j} & \forall (i,j)\in\mathcal{P} \\ & \xi_{i,j} \geq 0 & \forall (i,j)\in\mathcal{P} \end{array} \quad (1)$$

where w is the weight vector being fit, $\mathcal{P}$ is the set of pairwise preferences used for training, and C is a tunable parameter that penalizes misclassified input pairs. Once a weight vector w is learned, we can score unseen sentences by computing $w \cdot x_s$, where x_S is the feature vector for the sentence. These scores can then be used to rank sentences.

Ranking SVMs have been shown to significantly outperform standard SVMs for the sentence selection task and are currently the state of the art [29].

3.1.2 Support Vector Regression

Another generalization of the classical SVM formulation is support vector regression, which attempts to learn a regression model, rather than a classification or pairwise preference classification model. In our work, we fit a regression model directly to the human judgments, which typically corresponds to a target of $+1$ for relevant documents and -1 for non-relevant documents.

Support vector regression is formulated as follows:

$$\begin{array}{ll} \min & \frac{1}{2}\|w\|^2 + C_+\sum_{i:y_i=1}(\xi_i+\xi_i^*) + C_-\sum_{i:y_i=-1}(\xi_i+\xi_i^*) \\ s.t. & \\ & y_i - w\cdot x_i - b \leq \epsilon + \xi_i \\ & w\cdot x_i + b - y_i \leq \epsilon + \xi_i^* \\ & \xi_i, \xi_i^* \geq 0 \end{array} \quad (2)$$

where w is the weight vector being fit, C_- controls the cost associated with errors on non-relevant documents, C_+ controls the cost associated with errors on relevant documents, and ϵ is a free parameter controlling the amount of error tolerated for each input. In our experiments, we use $\epsilon = 0.1$.

Notice that the formulation we use allows for different costs for the relevant ($+1$ target) and non-relevant (-1 target) inputs. This is very important, since there are typically many more non-relevant sentences than there are relevant sentences. Therefore, it typically makes sense to ensure that the ratio of C_+ to C_- is greater than 1 in order to learn an effective model in the presence of such an imbalance.

3.1.3 Gradient Boosted Decision Trees

Gradient boosted decision trees are another technique that can be used for estimating a regression model [4]. Here, we use the stochastic variant of GBDTs [5]. GBDTs are a promising new machine learning approach that computes a function approximation by performing a numerical optimization in the function space instead of the parameter space. We provide a brief overview of the the GBDT algorithm and the parameters that influence the algorithm.

A basic regression tree $f(x)$, $x \in R^N$, partitions the space of explanatory variable values into disjoint regions R_j, $j = 1, 2, \ldots, J$ associated with the terminal nodes of the tree. Each region is assigned a value ϕ_j such that $f(x) = \phi_j$ if $x \in R_j$. Thus the complete tree is represented as:

$$T(x;\Theta) = \sum_{j=1}^{J} \phi_j I(x \in R_j), \quad (3)$$

where $\Theta = \{R_j, \phi_j\}_1^J$, and I is the indicator function. For a given loss function $L(y_i, \phi_j)$ the parameters are estimated by minimizing the the total loss:

$$\hat{\Theta} = \arg\min_{\Theta} \sum_{j=1}^{J} \sum_{x_i \in R_j} L(y_i, \phi_j). \quad (4)$$

Numerous heuristics are used to solve the above minimization problem.

A boosted tree is an aggregate of such trees, each of which is computed in a sequence of stages. That is,

$$f_M(x) = \sum_{m=1}^{M} T(x; \Theta_m), \quad (5)$$

where at each stage m, Θ_m is estimated to fit the *residuals* from the $m-1$th stage:

$$\hat{\Theta}_m = \arg\min_{\Theta_m} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \phi_{j_m}). \quad (6)$$

In practice, instead of adding $f_m(x)$ at the mth stage, one adds $\rho f_m(x)$ where ρ is the *learning rate*. This is similar to a "line search" where one moves in the direction of the gradient, but the step size need not be equal to the gradient. In the *stochastic* version of GBDT, instead of using the entire data set to compute to loss function, one sub-samples the data and then finds the function values ϕ_j such that the loss on the test set is minimized. The stochastic variant minimizes overfitting issues.

The depth of the trees in each stage is another algorithm parameter of importance. Interestingly, making the trees in each stage very shallow while increasing the number of boosted trees tends to yield good function approximations. In fact, even with depth 1 trees, often called stubs, it possible to achieve good results. Interaction amongst explanatory variables is modeled by trees of depth greater than 1.

Finally, the GBDT algorithm also provides what is called *feature importance* [4]. The importance is computed by keeping track of the reduction in the loss function at each feature variable split and then computing the total reduction of loss function along each explanatory feature variable. The importance is useful for analyzing which features contribute most to the model.

3.2 Features

Features play an important role in any machine learning algorithm. Since it is not the goal of this paper to undertake a comprehensive exploration of features for sentence selection, we use a relatively simple, yet representative set of features in our models. The features that we consider can be divided into those that are query dependent and those that are query independent. We now briefly describe how each is computed.

3.2.1 Query Dependent Features

Query dependent features attempt to capture how relevant a given sentence S is to the query Q. We use four different query dependent features that model relevance at different levels of granularity and expressiveness.

The first feature is *exact match*. It is a binary feature that returns 1 if there is an exact lexical match of the query string within the sentence. It is computed as:

$$f_{EXACT}(Q,S) = I(Q \text{ substring of } S) \tag{7}$$

where I is the indicator function that returns 1 if its argument is satisfied.

The next feature is *overlap*, which is simply the fraction of query terms that occur, after stopping and stemming, in the sentence. Mathematically, it is computed as:

$$f_{OVERLAP}(Q,S) = \frac{\sum_{w \in Q} I(w \in S)}{|Q|} \tag{8}$$

where $|Q|$ is the number of non-stopword terms that occur in Q.

The next feature, *overlap-syn*, generalizes the overlap feature by also considering synonyms of query terms. It is computed as the fraction of query terms that either match Q or have a synonym that matches Q. It is computed as:

$$f_{OVERLAP-SYN}(Q,S) = \frac{\sum_{w \in Q} I(SYN(w) \in S)}{|Q|} \tag{9}$$

where $SYN(w)$ denotes the set of synonyms of w. Note that $SYN(w)$ also includes w itself.

The last query dependent feature, *LM*, is based on the language modeling approach to information retrieval [25]. It is computed as the log likelihood of the query being generated from the sentence. The sentence language model is smoothed using Dirichlet smoothing [31]. The feature is computed as:

$$f_{LM}(Q,S) = \sum_{w \in Q} tf_{w,Q} \log \frac{tf_{w,S} + \mu P(w|C)}{|S| + \mu} \tag{10}$$

where $tf_{w,Q}$ is the number of times that w occurs in the query, $tf_{w,S}$ is the number of times w occurs in the sentence, $|S|$ is the number of terms in the sentence, $P(w|C)$ is the background language model, and μ is a tunable smoothing parameter.

Although approaches such as language modeling and BM25 are well known to be highly effective text retrieval models, we include all of the simpler query dependent features because they may provide additional useful information to the classifier when learning a sentence selection model. In fact, the features do end up playing an important role, as we will show in Section 5.

3.2.2 Query Independent Features

The goal of query independent features is to encode any *a prior* knowledge we have about individual sentences. Here, we use two very simple query independent features.

We expect that very short sentences and, possibly, very long sentences are less likely to be relevant, therefore our first query independent feature is *length*, which is the total number of terms in the sentence after stopping. It is computed as:

$$f_{LENGTH}(S) = |S| \tag{11}$$

The other query independent feature we consider is *location*, which is the relative location of the sentence within the document. The feature is computed according to:

$$f_{LOCATION}(S,D) = \frac{sentnum_D(S)}{\max_{S'} sentnum_D(S')} \tag{12}$$

where $sentnum_D(S)$ is the sentence number for S in D and $\max_{S'} sentnum_D(S')$ is the total number of sentences in D.

Although not explored here, other query independent features are possible, such as readability, formatting, among others.

3.3 Result Set Filtering

Each of the machine learning methods described produce a real-valued score for every query/sentence pair. For a given document, these scores can be used to produce a ranked list of the sentences within the document. However, when constructing a summary, we only want to consider the most relevant sentences in the document. This requires using a decision mechanism that filters the ranked list of sentences, eliminating the least relevant sentences, and keeping the most relevant ones. We now briefly describe two solutions to this problem that have been used in the past. In our evaluation, we compare the effectiveness of the two approaches.

3.3.1 Fixed Depth

Perhaps the most simple and straightforward way of filtering the result set is to only return the top k ranked sentences for every document. Filtering in this way is useful if the un-

	N2002	N2003	N2004
Query / Doc. Pairs	597	1187	1214
Avg. Sentences per Pair	52.1	31.9	30.5
Avg. Relevant Sentences per Pair	2.3	13.1	6.9

Table 1: Overview of the TREC Novelty Track data sets used in the experimental evaluation.

derlying summary construction algorithm requires a fixed number of sentences as input.

However, fixed depth filtering has several disadvantages. For example, if $k = 5$, but the document only contains a single relevant sentence, then we would end up returning four non-relevant sentences. At the opposite end of the spectrum, if the document contained ten relevant sentences, then we would miss out on returning five of them. Therefore, the fixed depth filtering scheme is very rigid and fails to adapt to documents with very few or very many relevant sentences.

3.3.2 *Global Score Threshold*

One way to overcome the rigid nature of fixed depth filtering is to filter based on the *scores* of the sentences. If we assume that the scores returned by the machine learning algorithm are reasonable, then it is fair to believe that sentences with higher scores will be more likely to be relevant than those with lower scores. Therefore, in order to filter, we can set a global score threshold, where sentences with scores above the threshold are returned, and sentences with scores below the threshold are not returned.

Of course, there are also issues concerned with global thresholding, such as the fact that scores may not be comparable across queries. However, our evaluation shows that this may actually not be an issue for the machine learning techniques used here. In fact, we will show that the optimal global score threshold, particularly for SVR and GBDTs, is not only comparable across queries, but also across data sets, meaning that it is very easy to choose such a threshold.

4. EXPERIMENTAL RESULTS

In this section, we evaluate the effectiveness of ranking SVMs, SVR, and GBDTs. We also analyze the effectiveness of two result set filtering techniques just described. All of our experiments are carried out on the 2002, 2003, and 2004 TREC Novelty Track data sets. These data sets include human relevance judgments for query / sentence pairs, and therefore can be used to evaluate sentence selection algorithms. Note that we do not use any of the novelty-related judgments associated with these data sets, only the relevance judgments. For more information on the details of these data sets, please refer to the TREC Novelty Track overview papers [9, 24, 23]. For our purposes, we throw out all query / document pairs that have no relevant sentences associated with them, since these are uninteresting from a learning and evaluation perspective. Summary statistics for the data sets are provided in Table 1. Notice that the characteristics of the data sets are quite varied, both in terms of average number of sentences per pair, as well as the proportion of relevant sentences per document. This allows us to analyze how the various learning algorithms perform over a range of data sets.

Algorithm 1 Evaluation Algorithm

```
for $i = 1$ to 5 do
  $(TRAIN, VALIDATE) \leftarrow split(TRAIN_i, p)$
  $utility_{max} \leftarrow -\infty$
  for $\theta \in \Theta$ do
    $model \leftarrow train(TRAIN; \theta)$
    $utility \leftarrow eval(model, VALIDATE)$
    if $utility > utility_{max}$ then
      $utility_{max} \leftarrow utility$
      $model_{max} \leftarrow model$
    end if
  end for
  output $rank(TEST_i, model_{max})$
end for
```

We use 5-folds cross validation for evaluation. All of the learning techniques have a number of hyperparameters that control various aspects of the learning algorithm. In order to properly tune the models, we must consider all reasonable settings of these hyperparameters. Therefore, we use a slightly modified version of 5-folds cross validation. The details of our evaluation algorithm are provided in Algorithm 1. In the algorithm, Θ is the set of hyperparameters that we will train over and *eval* is some evaluation measure that we are trying to maximize, such as precision, recall, or F1. As we see, during each training fold, the algorithm attempts to find the setting of the hyperparameters that maximizes the metric of interest by doing a brute force sweep over all reasonable settings. In order to control for overfitting, the effectiveness is measured on a held-out validation set.

For ranking SVMs, our algorithm sweeps over values for C (misclassification cost) and γ (RBF kernel variance). For SVR, we try various values for C_- and C_+ (misclassification costs), as well as γ. Finally, for GBDTs, we sweep over various weight values for the positive instances and tree depths (1, 2, 3). Additionally, for ranking SVMs and SVR, we only report results using the radial basis kernel, which provided the best results. Results for other kernels are omitted due to space constraints.

We use the SVMlight[1] implementation of ranking SVMs and SVR. We construct $\mathcal{P}$, the set of pairwise preferences used for training the ranking SVM, as the cross product of the relevant sentences and the non-relevant sentences for each query / document pair. For GBDTs, we use the GBM package for R [22].

Although none of the algorithms considered here directly maximize the metrics of interest to us, such as precision, recall, or F1, by training in this way we are implicitly optimizing for these measures by choosing the setting of the hyperparameters that maximizes the final measure we are interested in.

4.1 Sentence Selection

We now evaluate the effectiveness of the various machine learning algorithms for the sentence selection task within our experimental framework. There are many different ways to measure retrieval effectiveness, but most of the standard measures commonly used are inappropriate for the sentence selection task. From our perspective, R-Precision, computed

[1]`http://svmlight.joachims.org/`

	N2002	N2003	N2004
LM	.2602	.5566	.3944
Ranking SVM	$.3792^{\alpha}$	$.6904^{\alpha}$	$.4771^{\alpha}$
SVR	$.3587^{\alpha}$	$.7005^{\alpha\beta}$	$.4757^{\alpha}$
GBDT	$.4047^{\alpha\beta\delta}$	$.7060^{\alpha\beta}$	$.4806^{\alpha}$

Table 2: R-Precision for each data set and sentence selection approach. The α, β, and δ subscripts indicate a statistically significant improvement over language modeling, ranking SVMs, and SVR, respectively, according to a one-tailed pair t-test with $p < 0.05$.

over query/document pairs is one of the most meaningful measures. For a given query / document pair, R-Precision is computed as the precision at rank R, where R is the total number of relevant sentences in the document. This measure is appropriate because we ideally would like to return only the relevant sentences. Therefore, if a document has 10 relevant sentences, but we only retrieve 3 relevant sentences in the top 10, we have done a bad job for that document. Measures, such as precision at rank 10 do not take the number of relevant items per document into account, and therefore are not appropriate here.

Table 2 lists the R-Precision values of each learning method on each data set. For the sake of comparison, we also compare against language modeling (LM), which is a state of the art "bag of words" information retrieval model. The superscripts in the table indicate statistically significant improvements in R-Precision, as described in the caption.

The results indicate that all of the machine learned methods are better than language modeling, which is not surprising, since the language modeling score is a feature used by the learning algorithms that only considers term occurrences. This suggests that the other features we consider add considerable value.

Furthermore, we see that SVR is significantly better than ranking SVMs on the 2003 data set (1.5% improvement), and that GBDTs are significantly better than ranking SVMs on the 2002 (6.7% improvement) and 2003 (2.3% improvement) data sets. Therefore, the regression-based techniques are more effective than ranking SVMs, which is the current state of the art for sentence selection [29]. Lastly, we note that GBDTs are significantly better than SVR on the 2002 (12.8% improvement) data set. Thus, for the sentence selection problem, GBDTs are robust and highly effective across the different collections.

Interestingly, as the data set size grows, the effectiveness of ranking SVMs, SVR, and GBDTs seems to converge. This suggests that GBDTs, and SVR to a lesser extent, generalize better when the training data is sparse. It would be interesting to see if this behavior would persist if a larger feature set was used, as it would take more training examples to learn a good fit. This is an interesting area for future investigation.

4.2 Fixed Depth vs. Threshold Filtering

In our previous experiments, we always retrieved R sentences per query/document pair. While this was useful for comparing the effectiveness of the various techniques, it is not something that can be done in practice, since we do not know, *a priori*, how many sentences are relevant. Therefore, we must use one of the filtering techniques described earlier. When using these techniques, it is possible to retrieve a variable number of results per query/document pair. Therefore, R-Precision is no longer an appropriate measure. Instead, we use the F1 measure, which is the harmonic mean of precision and recall. This measure emphasizes the importance of both precision and recall and is comparable across query/document pairs that return different numbers of sentences.

In order to compare the effectiveness of fixed depth filtering and threshold filtering, we conduct an "upper bound" experiment. For each data set, we find the depth that results in the best F1, as well as the threshold setting that results in the best F1. We then can compare these two numbers to see which filtering technique, in the best case, would result in the best effectiveness. The results of this experiment are given in Table 3.

The results show that using fixed depth filtering is more effective on the 2002 data and threshold filtering yields better results on the 2003 and 2004 data sets. These results indicate that when there are very few relevant sentences per document, as is the case for the 2002 data set, a very shallow fixed depth filtering is better than using a global score threshold. Conversely, when there are many relevant sentences per document, as with the 2003 and 2004 data sets, fixed depth filtering is much worse than global thresholding.

In addition, the results show that GBDT have the most potential in terms of real world applicability, since the technique outperforms the others for both fixed depth and threshold filtering in a majority of cases. However, as we indicated, these results are upper bounds on the actual effectiveness that can be achieved using these filtering techniques.

In practice, one would have to either automatically learn the correct depth or threshold to use or use some robust "default" setting. Typically, it is difficult to define one setting that will work well across multiple data sets. However, as Figure 1 shows, the optimal threshold setting for GBDTs is very stable across the collections, more so than for ranking SVMs, and SVR. In fact, choosing -0.55 as a "default" threshold for GBDT yields an F1 that is within 2% of the optimal F1 for all three data sets.

Therefore, based on the sentence selection and filtering results, GBDTs appear to be the best choice of models to use out of the three that we explored. When using GBDTs, we recommend using fixed depth filtering for tasks with few relevant sentences per document, and that threshold filtering be used for tasks with many relevant sentences per document. Furthermore, if a threshold setting can not be reasonably estimated for the given task, then empirical evidence suggests that using -0.55 is a reasonable "default".

5. DISCUSSION

We now briefly discuss miscellaneous issues concerned with the approaches we explored in this paper.

5.1 Loss Functions

One theoretically interesting aspect of our work is the fact that regression-based models do not directly maximize the retrieval metric under consideration. Instead, they try to find a model that best fits the target labels. Ranking SVMs do not directly maximize general metrics, either, but they at least take the structure of the problem into account, more so, it seems, than simple regression models. However, as our results and the results of others indicate [17, 32], using

	N2002				N2003				N2004			
	Depth	F1	Thresh.	F1	Depth	F1	Thresh.	F1	Depth	F1	Thresh.	F1
Ranking SVM	2	.3411	-0.9	.2474	22	.5794	1.2	.6330	11	.4416	1.0	.4736
SVR	2	.3350	-0.9	.2880	22	.5791	-0.9	.6503	8	.4407	-0.2	.4637
GBDT	2	.3576	-0.55	.3302	20	.5771	-0.2	.6691	11	.4389	-0.5	.4745

Table 3: Comparison of result set filtering methods. For each data set, the optimal F1 measure for each technique is reported. The optimal depth and threshold settings are also reported.

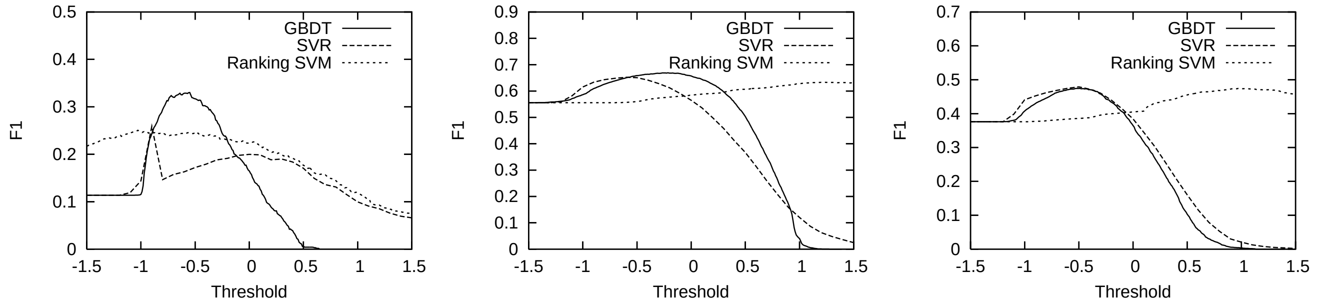

Figure 1: Effectiveness, measured in terms of F1, as a function of threshold value for the TREC 2002, 2003, and 2004 Novelty data sets (left to right).

these GBDT models prove to be highly effective. However, it is not yet clear as to exactly why this is the case. It would be interesting, as part of future work, to compare the effectiveness of regression-based techniques with those that directly optimize the metric of interest for this task.

5.2 Feature Importance

As discussed in Section 3, GBDTs provide a mechanism for reporting the relative importance of each feature. By analyzing the relative importances, we can gain insights into the importance of each feature for a given data set.

As an example, Figure 2 plots the relative feature importances of the features for the 2002 (top) and 2003 (bottom) data sets. It is interesting to note that the ordering of the importances is different for the two data sets. The two features with the highest importance for the 2002 data set are $overlap - syn$ (query/sentence overlap with synonyms) and $overlap$ (query/sentence overlap), whereas the two features with the highest importance for the 2003 data set are $length$ (sentence length) and lm (language modeling score). The ordering is also different for the 2004 data set, which indicates it may be difficult to manually construct a heuristic rule-based method that works well for all data sets. Although such rule-based methods may work well for a single task, such as web search, we are primarily interested in developing approaches that work well across a wide range of application domains.

5.3 Efficiency

Although our primary focus in this work is on effectiveness, we briefly describe our general observations on the efficiency of the three machine learning approaches explored here. First, the ranking SVM model was the least efficient of the techniques. This is due to the fact that the model is trained over pairwise preferences, which are inherently quadratic in nature. The SVR did not suffer from this problem, however. Second, SVR and ranking SVM models took even longer to train when the RBF kernel was used. Train-

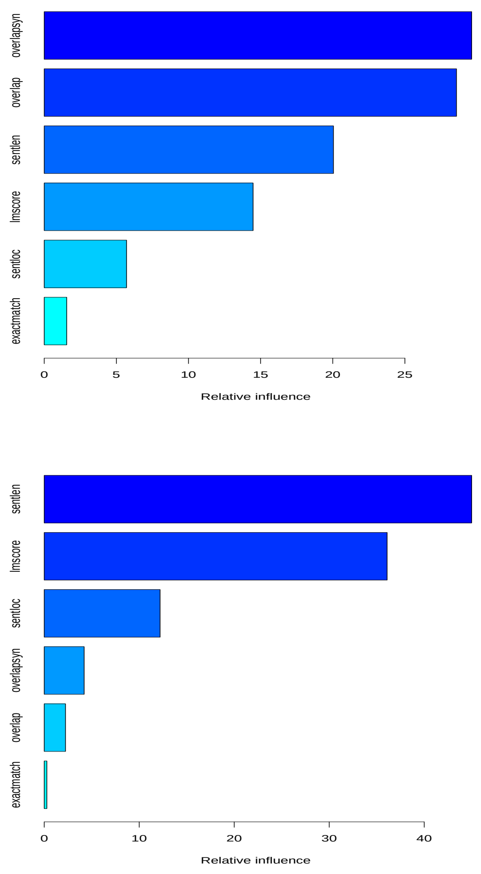

Figure 2: Relative feature importances, as computed by gradient boosted decision trees, for the Novelty 2002 (top) and 2003 (bottom) data sets.

ing time was significantly reduced when the "linear" kernel was used instead, but effectiveness was reduced. Finally, the GBDTs took significantly less time to train than the SVR and ranking SVMs (with and without kernels). The GBDTs were boosted for up to 1500 iterations. However, retrospective analysis shows that the optimal number of trees (iterations) for a model was always less than 200, which means that the training time could have been sped up even more. Therefore, in addition to the advantages GBDTs provide with respect to effectiveness, they also provide a number of benefits in terms of efficiency, as well.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed using regression-based machine learning techniques, such as support vector regression (SVR) and gradient boosted decision trees (GBDTs), for the sentence selection task, which is an important sub-task of constructing query-biased abstracts and summaries.

Our experimental results showed that SVR and GBDTs significantly outperform a simple language modeling baseline and ranking SVMs, which are considered to be the current state of the art. Our results also show that GBDTs are very robust and achieve strong effectiveness across three data sets of varying characteristics.

We also investigated two result set filtering techniques, including fixed depth and global score threshold filtering. Our results showed that fixed depth filtering is effective when there are few relevant sentences per document and that threshold filtering is more effective when there are many relevant sentences per document. Furthermore, our results indicated that threshold-based filtering for GBDTs is much more stable across data sets than ranking SVMs or SVR.

As part of future work, we plan to compare SVR and GBDTs to methods that directly maximize R-Precision or F1 to better understand the impact of the underlying loss function. We would also like to investigate set-based ranking algorithms in order to incorporate notions of novelty and sub-topic coverage. In addition, we would like to make our feature sets available as part of the growing LETOR benchmark [18] so that other researchers can develop and evaluate learning to rank techniques for the sentence selection task, which has very different characteristics than the typical *ad hoc* and web retrieval tasks.

7. REFERENCES

[1] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proc. 22nd Proc. Intl. Conference on Machine Learning*, pages 89–96, 2005.

[2] Y. Cao, J. Xu, T.-Y. Liu, H. Li, Y. Huang, and H.-W. Hon. Adapting ranking svm to document retrieval. In *Proc. 29th Ann. Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 186–193, 2006.

[3] C. Clarke, E. Agchtein, S. Dumais, and R. White. The influence of caption features on clickthrough patterns in web search. In *Proc. of SIGIR*, 2007.

[4] J. H. Friedman. Greedy function approximation: A graidient boosting machine. *Annals of Statistics*, 29:1189–1232, 2001.

[5] J. H. Friedman. Stochastic gradient boosting. *Computational Statistics and Data Analysis*, 38:367–378, 2001.

[6] 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.

[7] F. Gey. Inferring probability of relevance using the method of logistic regression. In *Proc. 17th Ann. Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 1994.

[8] J. Goldstein, M. Kantrowitz, V. Mittal, and J. Carbonell. Summarizing text documents: sentence selection and evaluation metrics. In *Proc. 22nd Ann. Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 121–128, 1999.

[9] D. Harman. Overview of the trec 2002 novelty track. In *Proc. 11th Text REtrieval Conference*, 2002.

[10] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Sringer-Verlag, New York, NY, 2001.

[11] A. K. Jain, R. P. W. Duin, and J. Mao. Statistical pattern recogntion: A review. *IEEE Transactions on Pattern Analysis and Machine Learning*, 22:4–37, 2000.

[12] T. Joachims. Optimizing search engines using clickthrough data. In *Proc. 8th Ann. Intl. ACM SIGKDD Conf. on Knowledge Discovery and Data Mining*, pages 133–142, 2002.

[13] T. Joachims. A support vector method for multivariate performance measures. In *Proc. 22nd Proc. Intl. Conference on Machine Learning*, pages 377–384, 2005.

[14] K. Knight and D. Marcu. Statistics-based summarization — step one: Sentence compression. In *Proc. of AAAI*, 2000.

[15] J. Kupiec, J. Pedersen, and F. Chen. A trainable document summarizer. In *Proc. 18th Ann. Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 68–73, 1995.

[16] Q. Le and A. Smola. Direct optimization of ranking measures. http://www.citebase.org/abstract?id=oai:arXiv.org:0704.3359, 2007.

[17] P. Li, C. J. Burges, and Q. Wu. Mcrank: Learning to rank using multiple classification and gradient boosting. In *Proc. 21st Proc. of Advances in Neural Information Processing Systems*, 2007.

[18] T.-Y. Liu, J. Xu, T. Qin, W. Xiong, and H. Li. LETOR: Benchmark dataset for research on learning to rank for information retrieval. In *SIGIR 2007 workshop: Learning to Rank for Information Retrieval*, 2007.

[19] I. Mani and M. T. Maybury. *Advances in Automatic Text Summarization*. MIT Press, Cambridge, MA, 1999.

[20] R. Nallapati. Discriminative models for information retrieval. In *Proc. 27th Ann. Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 64–71, 2004.

[21] D. R. Radev and E. Hovy. *Intelligent Text Summarization*. AAAI, 1998.

[22] G. Ridgeway. The state of boosting. *Computing Science and Statistics*, 31:172–181, 1999.

[23] I. Soboroff. Overview of the trec 2004 novelty track. In *Proc. 13th Text REtrieval Conference*, 2004.

[24] I. Soboroff and D. Harman. Overview of the trec 2003 novelty track. In *Proc. 12th Text REtrieval Conference*, 2003.

[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] A. Tombros and M. Sanderson. Advantages of query biased summaries in information retrieval. In *Proc. 21st Ann. Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 2–10, 1998.

[27] A. Turpin, Y. Tsegay, D. Hawking, and H. E. Williams. Fast generation of result snippets in web search. In *Proc. 30th Ann. Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 127–134, 2007.

[28] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Sringer-Verlag, New York, NY, 2002.

[29] C. Wang, F. Jing, L. Zhang, and H.-J. Zhang. Learning query-biased web page summarization. In *Proc. 16th Intl. Conf. on Information and Knowledge Management*, pages 555–562, 2007.

[30] Y. Yue, T. Finley, F. Radlinski, and T. Joachims. A support vector method for optimizing average precision. In *Proc. 30th Ann. Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, page To appear, 2007.

[31] 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.

[32] Z. Zheng, H. Zha, T. Zhang, O. Chapelle, K. Chen, and G. Sun. A general boosting method and its application to learning ranking functions for web search. In *Proc. 21st Proc. of Advances in Neural Information Processing Systems*, 2007.