Extracting Query Modifications from Nonlinear SVMs

Gary William Flake
NEC Research Institute
4 Independence Way
Princeton, NJ 08540
flake@research.nj.nec.com

Eric J. Glover
NEC Research Institute
4 Independence Way
Princeton, NJ 08540
compuman@research.nj.nec.com

Steve Lawrence
NEC Research Institute
4 Independence Way
Princeton, NJ 08540
lawrence@necmail.com

ABSTRACT

When searching the WWW, users often desire results restricted to a particular document category. Ideally, a user would be able to filter results with a text classifier to minimize false positive results; however, current search engines allow only simple query modifications. To automate the process of generating effective query modifications, we introduce a sensitivity analysis-based method for extracting rules from nonlinear support vector machines. The proposed method allows the user to specify a desired precision while attempting to maximize the recall. Our method performs several levels of dimensionality reduction and is vastly faster than searching the combination feature space; moreover, it is very effective on real-world data.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*query formulation*; I.5.1 [**Pattern Recognition**]: Models—*statistical*

General Terms

Algorithms, Experimentation

Keywords

query modification, rule extraction, support vector machine, sensitivity analysis

1. INTRODUCTION

When searching the WWW, users often desire results from a specific category, such as personal home pages or conference announcements. Unfortunately, many search engines return results from multiple categories, thus forcing the user to manually filter the results. One solution is to use an automated classifier that identifies whether a given result is close to the user's desired category. However, since search engines often have low precision for a given category, it may be necessary to retrieve a large number of documents from the engines, which may be expensive or impossible (search engines typically only allow retrieving a certain maximum number of hits for a query). A single query modification (such as `+"home page"`) can improve precision, but often at the expense of recall. To allow for both high precision and high recall, we introduce a method to generate a collection of query modifications that satisfies the user's desired precision and attempts to maximize recall.

We use a nonlinear support vector machine (SVM) to initially classify all documents. The problem of building classifiers that generalize-well is especially important in the domain of text classification because typical problem instances have high-dimensional input spaces (mapping to a word vector) with a relatively small number of positive examples. The scarcity of positive examples is related to the cost of having humans hand-classify examples. SVMs are specifically designed to generalize-well in high-dimensional spaces with few examples. Moreover, they have been shown to be extremely accurate and robust on text classification problems [8, 9], which is why we use SVMs in favor of other classification methods.

We also know of no other work that extracts symbolic information from SVMs; hence, we are also motivated by the desire to use the high accuracy of SVMs to extract valuable symbolic features, similar to earlier works that did the same for multilayer perceptrons [14].

Our feature extraction process uses sensitivity analysis on the underlying SVM to identify a linear model and a single query modification that explains a subset of the desired documents. We extract additional query modifications by iterating the procedure with a new dataset composed of the false negatives of the linear model and all negative examples. The procedure repeats, generating additional query modifications, until no further progress is made. In this way, our feature extraction method is similar to RIPPER [1] in that we iteratively extract rules; however, our method differs in that we use the SVM to guide our choice in rule selection.

In the end, we can identify a set of query modifications whose combination "covers" a large portion of the positive documents, but greatly reduces the number of false positives. Our method yields precise search results and can be built on top of existing search engines in the form of a metasearch engine. Moreover, by using SVMs to guide the rule search, our extracted rules are predisposed to have many of the same generalization qualities that the originating SVM possesses.

This approach differs from our other work on learning query modifications [6] by producing a set of query modifications that work together to improve recall, as opposed to a set of individually effective modifications, that may or may not work well together.

This paper is divided into five sections. In Section 2, we discuss WWW search engines and metasearch engines, with an emphasis on how our results can be used to improve

WWW2002, May 7–11, 2002, Honolulu, Hawaii, USA.
ACM 1-58113-449-5/02/0005.

metasearch engines. In Section 3, we describe our method for producing effective query modifications, which uses text preprocessing, SVM classification, sensitivity analysis, and a dataset deflation procedure. Section 4 gives experimental results for identifying personal home pages and conference pages and shows that our method gives both high precision and improved recall. Finally, Section 5 summarizes our work with a discussion on how web page patterns are exploited by our method's dimensionality reduction properties.

2. WWW SEARCH AND METASEARCH

The primary tool for accessing data on the Web is a search engine such as AltaVista, Northern Light, Google, Excite, and others. All search engines accept keyword queries and return a list of relevance ranked results, where relevance is usually a topical measure. As every search engine user has experienced, typical search queries often return thousands of URLs, placing a burden on the user to manually filter the results.

One common trick for combating this problem is to "spike" a search query with a modification that is designed to narrow the results to a specific context or category. For example, if searching for the personal home pages of machine learning researchers, adding the term `"home page"` can increase the density of valuable results by encouraging a search engine to rank personal home pages higher than non-home pages. However, using any single query modification increases the search precision at the expense of recall, i.e., many home pages (e.g., those that do not contain "home page") may be missed.

The coverage of a search query can be improved by using a metasearch engine, such as DogPile, SavvySearch [7], MetaCrawler [13], or Profusion [3]. Each submits the user's query to multiple search engines and fuses the results, allowing for potentially higher recall. Most simple metasearch engines fuse results by considering only the titles, URLs and short summaries returned from the underlying search engines. As a result, it is more difficult for them to assess topical relevance or to determine if a particular result is of the type desired by the user. In addition, since metasearch engines combine many search engine results, there is a risk of one search engine causing the total precision to drop.

Unlike typical metasearch engines, content-based metasearch engines, such as Inquirus [10], download all possible results and consider the full HTML of each page when ranking them. Although this strategy may improve the accuracy of relevance ranking, it does not allow users to control the desired category of results and, at best, only organizes existing results. Just as with any metasearch engine, Inquirus can have low precision if the underlying search engines also have low precision. As a result, the improved coverage does not guarantee improved recall.

Some search and metasearch engines, such as Northern Light and SavvySearch, allow users to specify a desired category from a limited set of categories. Northern Light constrains user searches to results known to fall into the chosen cluster, while SavvySearch only submits the query to search engines known to be of the desired category, such as MP3 or news sites. Each approach offers the user improved control, but both are still limited.

A problem arises when a user's desired document category does not exactly match the provided choices, or the user wishes to submit his query to a general purpose search engine (to improve recall). There is no way to provide custom categories to these search tools, neither of which have a category for "personal home pages" or "conference pages." Nevertheless, users often desire documents in a well-defined category that is not easily localized like those covered by specialized metasearch engines.

Our work is motivated by the desire to build a metasearch engine that can adaptively specialize to many different document categories. Our prototype system, Inquirus 2 [5, 4], currently allows users to focus a search on categories such as "personal home pages," "research papers," and "product reviews." Previous versions of Inquirus 2 used hand-coded query modifications to improve the search performed by the search engines. This work gives a method by which effective query modifications that have both high precision and high recall can be generated automatically.

3. FINDING QUERY MODIFICATIONS

Our method for automatically identifying effective query modifications is a five-step process that uses labeled examples. The first stage is to preprocess the textual content of an HTML document into n-gram features that appear to have discriminatory power. With the document features and labels, we then train a nonlinear SVM to classify the documents. Sensitivity analysis on the SVM is used to estimate the importance of document features to the SVM classifier. The most important features are exhaustively analyzed to find the combination of query modifications that yields the highest recall for a desired level of precision.

Documents that are labeled as true positives by the best query modification are removed from the dataset. The process then repeats with a new SVM classifier until no new query modifications are found. The entire method is described in greater detail in the next four subsections.

3.1 Preprocessing

Our preprocessing extracts the full text and title text of a document and converts it into features that consist of up to three consecutive words. All non-letter characters are converted to whitespace and all capital letters are converted to lower case.

After every page in the training set is converted, two feature histograms are constructed for positive and negative exemplars. Since the number of features per document can be on the order of thousands, we reduce the number of features by eliminating those features that are too rare (such as proper names). Common features, such as stop words, are removed by the feature scoring process as well. However, since we distinguish between title text and the full document text, a particular stop word such as "s" in the title (from apostrophe "s"), could be a strong feature, even though it is common in the text.

This dimensionality reduction considers the relative ability of any given feature to distinguish between positive and negative exemplars by assigning a score to each feature. The score is generated via the following four sets:

$$\begin{aligned} \mathcal{P} &= \{ \text{ all positive examples } \} \\ \mathcal{N} &= \{ \text{ all negative examples } \} \\ \mathcal{P}_f &= \{p \in \mathcal{P} : p \text{ contains feature } f\} \\ \mathcal{N}_f &= \{n \in \mathcal{N} : n \text{ contains feature } f\} \end{aligned}$$

Ignoring higher order correlations, the best features for

classifying are those which occur only in one set and the worst features are those which occur in an equal percentage of each set, or occur very frequently in both sets. A simple scoring function, score(f), that captures this notion is:

$$\max\left(\frac{|\mathcal{P}_f|/|\mathcal{P}|}{|\mathcal{P}_f|/|\mathcal{P}|+|\mathcal{N}_f|/|\mathcal{N}|}, \frac{|\mathcal{N}_f|/|\mathcal{N}|}{|\mathcal{P}_f|/|\mathcal{P}|+|\mathcal{N}_f|/|\mathcal{N}|}\right)$$

which is the probability (given equal sizes for $\mathcal{P}$ and $\mathcal{N}$) that you know which set a document containing feature f came from. The worst one can do is random or 0.5, and the best is absolute certainty, or 1. Unfortunately, this equation would predict all features which occur only once as being perfect classifiers. To remedy this problem, we add a requirement that a feature occur in at least some threshold percentage of documents from either $\mathcal{P}$ or $\mathcal{N}$. Any feature occurring less than the threshold (7.5% in our experiments) is removed from consideration.

After each feature is scored the top N are taken. For our experiments we used N equal to 100 for the personal home pages, and 300 for conference pages. After the top N features are determined, each page is converted to a binary vector, where each feature is assigned $\{-1, 1\}$, with -1 indicating that a feature is absent.

3.2 SVM Classification

Consider a set of data, $\{(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_N, y_N)\}$, such that $\boldsymbol{x}_i$ is an input and $y_i \in \{-1, 1\}$ is a target output. A support vector machine is a model that is calculated as a weighted sum of kernel function outputs. The kernel function of an SVM is written as $K(\boldsymbol{x}_a, \boldsymbol{x}_b)$ and it can be an inner product, Gaussian, polynomial, or any other function that obeys Mercer's condition [15].

In the simplest case, where $K(\boldsymbol{x}_a, \boldsymbol{x}_b) = \boldsymbol{x}_a \cdot \boldsymbol{x}_b$ and the training data is linearly separable, computing an SVM for the data corresponds to minimizing $||\boldsymbol{w}||$ such that

$$y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i - w_0) - 1 \geq 0, \forall i$$

Thus, an SVM yields the lowest complexity linear classifier that correctly classifies all data. The solution for $\boldsymbol{w}$ is found by solving the quadratic programming problem defined by the objective function and the constraints. In the linear case, the solution is

$$\boldsymbol{w} = \sum_{i=1}^{N} y_i \lambda_i \boldsymbol{x}_i$$

with $\sum_i y_i \lambda_i = 0$. The λ terms are the Lagrange multipliers found in the primal and dual Lagrangians. In many cases, most of the Lagrange multipliers will be zero. The only nonzero multipliers correspond to data points that lie closest to the decision boundary.

The formalism behind SVMs has been generalized to accommodate nonlinear kernel functions and slack variables for miss-classifications. We write the output of a nonlinear SVM as:

$$\begin{aligned} f(\boldsymbol{x}, \boldsymbol{\lambda}) &= \sum_{i=1}^{N} y_i \lambda_i \boldsymbol{\Phi}(\boldsymbol{x}_i) \cdot \boldsymbol{\Phi}(\boldsymbol{x}) + \lambda_0 \\ &= \sum_{i=1}^{N} y_i \lambda_i K(\boldsymbol{x}_i, \boldsymbol{x}) + \lambda_0. \end{aligned} \quad (1)$$

Thus, $K(\cdot)$ is a dot product in a nonlinear feature space $\boldsymbol{\Phi}(\boldsymbol{x})$. The objective function (which should be minimized) for Equation 1 is:

$$E(\boldsymbol{\lambda}) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \lambda_i \lambda_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{i=1}^{N} \lambda_i, \quad (2)$$

subject to the box constraint $\forall_i, 0 \leq \lambda_i \leq C$ and the linear constraint $\sum_i y_i \lambda_i = 0$. C is a user-defined constant that represents a balance between the model complexity and the approximation error; in the Lagrangian, C is multiplied by the sum of the magnitude of the slack variables used for absorbing miss-classifications.

Equation 2 will always have a single minimum with respect to the Lagrange multipliers, $\boldsymbol{\lambda}$. The minimum to Equation 2 can be found with any of a family of algorithms, all of which are based on constrained quadratic programming. We used a faster variation [2] of the Sequential Minimal Optimization algorithm [11, 12] in all of our experiments.

When Equation 2 is minimal, Equation 1 will have a classification margin that is maximized for the training set. For the case of a linear kernel function ($K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i \cdot \boldsymbol{x}_j$), an SVM finds a decision boundary that is balanced between the class boundaries of the two classes. In the nonlinear case, the margin of the classifier is maximized in the nonlinear feature space, which results in a nonlinear classification boundary.

In our experiments we used a Gaussian kernel function of the form

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-||\boldsymbol{x}_i - \boldsymbol{x}_j||^2/\sigma^2\right). \quad (3)$$

The choice of σ is usually made to reflect the smoothness of the feature space and the density of the training data. As there is no general method for selecting σ, our choice is admittedly *ad hoc*, which we heuristically set to 15. However, one may have success with using a cross validation procedure choosing σ.

3.3 Sensitivity Analysis

After training a nonlinear SVM to classify our labeled training data, we use a form of sensitivity analysis to identify components of the document feature space that are important to the classifier. Suppose that our classifier is linear in the input space, i.e.,

$$f(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x} - w_0.$$

In the linear case, input features that are important are those with the largest coefficient magnitudes, $|w_i|$. In the nonlinear case, we can linearize the model about a point in input space with a Taylor expansion to obtain:

$$f(\boldsymbol{x}) \approx \hat{f}(\boldsymbol{x}) = f(\boldsymbol{v}) + (\boldsymbol{x} - \boldsymbol{v}) \left.\frac{\partial f}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\boldsymbol{v}}.$$

Thus, $\hat{f}(\boldsymbol{x})$, is a linear approximation to $f(\boldsymbol{x})$ that is locally accurate in the vicinity of $\boldsymbol{v}$. The largest components of $\partial f/\partial \boldsymbol{x}|_{\boldsymbol{x}=\boldsymbol{v}}$ are those features that are important to the linear approximation. Thus, if we want to know which inputs are most critical to computing $f(\boldsymbol{x})$ for choices of $\boldsymbol{x}$ near $\boldsymbol{v}$, we can restrict our attention to the inputs that have relatively large values of $|\partial f/\partial \boldsymbol{x}|$ evaluated at $\boldsymbol{x} = \boldsymbol{v}$, which happen to be the inputs that when changed, produce the largest change in $f(\boldsymbol{x})$.

Table 1: Procedure to find effective query modification given an SVM and a working dataset.

- **for** all $i : \lambda_i > 0 \wedge y_i = 1$
  1. calculate sensitivity, $df/d\boldsymbol{x}|\boldsymbol{x}=\boldsymbol{x}_i$
  2. find largest d magnitude components, $\boldsymbol{c}$, of sensitivity
  3. **for** all $2^d - 1$ combinations of $\boldsymbol{c}$
     (a) test query modification and note statistics
     (b) **if** precision rate is above desired value and recall is greater than best found so far, **then** save query modification.
- **return** best found query modification

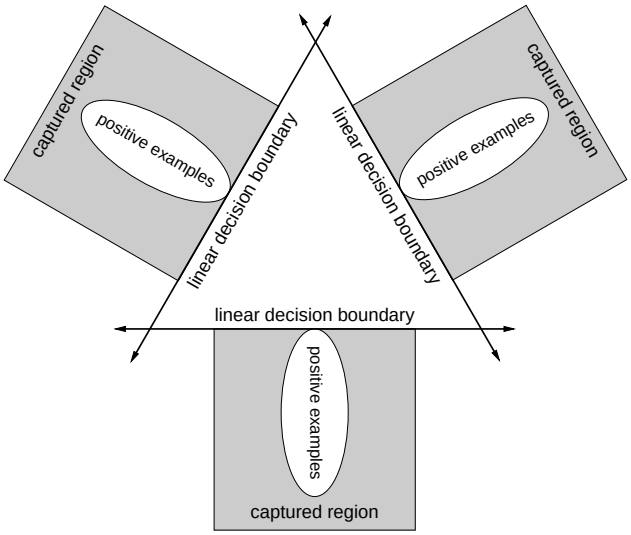

Figure 1: Rules for finding multiple regions of positive documents can be extracted by dataset deflation. Subsequent iterations find regions in input space that are explained by simple linear rules. Each identified rule/region will be mostly orthogonal (and, therefore, complementary) to previous rules/regions. In the high-dimensional input spaces typical of text classification problems, there are many such orthogonal regions (unlike this 2-dimensional example).

But what value should $\boldsymbol{v}$ take? An interesting aspect of SVM learning is that the non-zero Lagrange multipliers correspond to data points that are crucial to determining the maximum margin classifier. Any data point with a zero Lagrange multiplier is redundant in the sense that its removal would not alter the solution. All other data points, with Lagrange multipliers at the solution taking non-zero values, are the so-called "support vectors" which are the only data points strictly needed in order to build the SVM model. For many real-world problems (text classification among them), the number of support vectors may be dramatically fewer than the number of data points. Thus, our search for useful values of $\boldsymbol{v}$ can be simplified by only searching those points in the input space that are also support vectors (i.e., have non-zero Lagrange multipliers). Moreover, since we are more interested in discovering rules that identify positive members of the document category, we restrict our attention to those positive support vectors that have y_i equal to 1. In this way, our search is restricted the few data points that are actually important to the SVM model.

The input sensitivity of the SVM from Equation 1 with the Gaussian kernel from Equation 3 is easily derived as:

$$\frac{\partial f}{\partial \boldsymbol{x}} = \frac{2}{\sigma^2} \sum_{i=1}^{N} (\boldsymbol{x}_i - \boldsymbol{x}) y_i \lambda_i K(\boldsymbol{x}_i, \boldsymbol{x}). \quad (4)$$

To find an effective query modification, we use the procedure described in Table 1. The procedure iterates over all positive training examples that have a non-zero Lagrange multiplier. A family of query modifications is found by identifying the largest d components of the sensitivity vector. Since no mainstream search engine allows the user to specify weights for individual terms (i.e., they only allow `+term` and `-term`), we cannot use all of the information in the coefficients of the sensitivity vector.

Instead, we treat positive coefficients as having a weight of 1, negative with -1, and all others 0. In this way, a query modification can be thought of as a simple threshold classifier with weights that are all in $\{-1, 0, 1\}$. At line 3 in Table 1, we examine all possible variations of a query modification that optionally zero out one or more terms. This restricted search is necessary because of the implicit thresholding that we are performing on the weights. However, the search is not as expensive as the bounds for the second for loop suggest because the sensitivity analysis may produce duplicate suggestions. In this case, we can hash the test query modifications, and only evaluate those that have not been evaluated thus far.

In the end, we find a query modification that (almost always) satisfies a pre-specified desired precision, but tends to maximize recall. If no query modification is found by the procedure, the search for effective query modifications is finished.

3.4 Rinse and Repeat

After finding the first effective query modification, we find additional query modifications by altering the training dataset for the SVM. All true positive training exemplars are removed from the dataset, leaving all negative exemplars and false negatives.

By deflating the dataset in this manner, we can find multiple local linear rules that are all effective, especially when fused together. Figure 1 shows a stylized example of how the method can work. In the example, we have three regions that are mostly disjoint from one another. For each region, the sensitivity analysis discovers a linear rule with the following properties. First, the location of the decision boundary will always be on the edge of the region that is closest to the other regions (which is guaranteed by only searching over support vectors). Second, the decision boundary will be parallel to the direction of greatest sensitivity of the region (which also implies that the decision boundary will be

Table 2: Extracted query modifications from personal home page data with desired precision set to 50%.

QM#	extracted query modification
1	`title:s title:page "about me"`
2	`"i am" "my favorite" "sign my"`
3	`"is my" "my page" "interests"`
4	`title:home "s home"`
5	`interests "my page" "me at"`
6	`"sign my" title:home`
7	`"my page" interests "welcome to my"`

perpendicular to the region's direction of greatest variance). As a result of these two properties, each decision boundary can be viewed as the tightest "slice" that removes a region from the rest of the training examples.

Composing multiple "slices" in this way allows us to potentially separate large regions of positive examples with a small number of rules. Moreover, the procedure is structured so that each rule is a conjunction of terms while the composition of the rules is formed as a disjunction (i.e., disjunctive normal form); as a result, the rules can be submitted to a search engine as multiple queries.

4. EXPERIMENTAL RESULTS

We now describe how our method has been used to generate effective query modifications for identifying personal home pages and conference pages. For our results, we use the notation `title:term` to indicate that `term` should occur in the title of a document. Many search engines, such as AltaVista, support queries of this form. All other query modifications specify terms that should appear in the title or the full text of the document.

4.1 Personal Home Pages

The training set for this experiment consisted of 250 personal home pages and 999 random URLs that were given negative labels. All of the documents were preprocessed into vectors of 100 features. The negative exemplars contained approximately 1% false positives, based on human inspection of random samples. We set the desired precision to be at least 50%. The SVM was optimized with σ set to 7, C set to 5, and d from the extraction procedure was set to 5. The extracted query modifications are listed in Table 2

In the table, one line represents a single query modification. Notice that all of the query modifications contain less than four additional search terms even though our procedure searched for as many as five terms in a single modification.

To measure the effectiveness of the query modifications, we tested three queries without any modifications and with the first four generated query modifications. The results are presented in Table 4. Whenever a search engine returned more than 50 results, we only manually classified the first 50 pages; performing the statistics in this manner is consistent with our goal since we are using the query modifications to bias the search engine's ranking function.

As can be seen in Table 4, the fourth query modification for home pages gives better precision in all cases, but the merged results of all four query modifications gives the highest recall as indicated by the low overlap between returned results. Moreover, in each case, the combined precision is at least as high as the desired precision of 50%.

We also note that our test produced many pages that were listings of links to personal home pages, "about me" type pages, or CV and resume pages; thus, while these pages were labeled as false positives for our statistics, they are not too far from being desirable results.

Table 3: Extracted query modifications from conference page data with desired precision set to 50%.

QM#	extracted query modification
1	`"call for"`
2	`"program committee"`
3	`conference hotel workshops`
4	`"this workshop"`
5	`"fax email" "the conference" "sponsored by"`

4.2 Conference Pages

The conference web page data consisted of 529 negative examples and 150 positive examples. All of the documents were preprocessed into vectors of 300 features. As before, we set the desired precision to be 50%. The SVM was optimized with σ set to 15, C set to 10, and d from the extraction procedure was set to 5 as in the earlier test. The first five extracted query modifications are shown in Table 3.

As in the earlier test case, we used the top four query modifications to augment three test search queries: `"neural networks"`, `"learning"`, and `"linux"`. The test results are summarized in Table 5. For this experiment, we manually classified the first 20 web pages for all queries (classifying conference pages by hand is more difficult than home pages).

As can be seen, the merged results maintained a precision level that was close to the desired precision, while increasing recall compared to the individual query modifications. Moreover, the combined query modifications offer some insurance in case any single query modification fails.

5. DISCUSSION AND CONCLUSIONS

As stated in Section 2, our motivation for this work is to automate the process of generating effective query modifications for the Inquirus 2 metasearch engine. Previously, our query modifications were human-generated and selected based on the very unscientific notion that they seemed to "make sense." While some of the human-generated queries were supported by our experiments, our automated method discovered many novel query modifications, which, in turn, suggest some interesting facts about documents on the Web and about the method we propose in this paper.

5.1 Category Patterns

As part of this work, we performed many experiments on personal home pages that revealed interesting trends for the types of home pages that exist, and the different usages of language on the pages, which can be used to distinguish among them. For example, in one experiment we found `geocities` to be a very effective modifier because of the large number of free personal web pages hosted by GeoCi-

Table 4: Test results for using extracted home page query modifications: "Total Pages" refers to the number examined, which were always the top results, with a maximum of 50 pages examined at most.

query	home pages	total pages	precision
+"information retrieval"	0	50	0%
+"information retrieval" +title:s +title:page +"about me"	3	3	100%
+"information retrieval" +"i am" +"my favorite" +"sign my"	0	1	0%
+"information retrieval" +"is my" +"my page" +"interests"	0	12	0%
+"information retrieval" +title:home +"s home"	46	50	96%
Merging all 4 query modifications and removing duplicates (No duplicates)	49	66	74%
+"beagles"	1	50	2%
+"beagles" +title:s +title:page +"about me"	5	7	71%
+"beagles" +"i am" +"my favorite" +"sign my"	7	32	22%
+"beagles" +"is my" +"my page" +"interests"	0	9	0%
+"beagles" +title:home +"s home"	39	50	78%
Merging all 4 query modifications and removing duplicates (4 positive duplicates)	47	94	50%
+"starcraft"	0	50	0%
+"starcraft" +title:s +title:page +"about me"	24	34	71%
+"starcraft" +"i am" +"my favorite" +"sign my"	27	50	54%
+"starcraft" +"is my" +"my page" +"interests"	15	49	31%
+"starcraft" +title:home +"s home"	44	50	88%
Merging all 4 query modifications and removing duplicates (9 positive duplicates)	101	174	58%

Table 5: Test results for using extracted conference page query modifications: "Total Pages" refers to the number examined, which were always the top results, with a maximum of 20 pages examined at most.

query	conference pages	total pages	precision
+"neural networks"	1	20	5%
+"neural networks" +"call for"	10	20	50%
+"neural networks" +"program committee"	12	20	60%
+"neural networks" +conference +hotel +workshops	17	20	85%
+"neural networks" +"this workshop"	16	20	80%
Merging all 4 query modifications and removing duplicates (4 positive duplicates, 1 negative duplicate)	51	75	68%
+"learning"	0	20	0%
+"learning" +"call for"	5	20	25%
+"learning" +"program committee"	9	20	45%
+"learning" +conference +hotel +workshops	15	20	75%
+"learning" +"this workshop"	6	20	30%
Merging all 4 query modifications and removing duplicates (No duplicates)	35	80	44%
+"linux"	3	20	15%
+"linux" +"call for"	5	20	25%
+"linux" +"program committee"	9	20	45%
+"linux" +conference +hotel +workshops	12	20	60%
+"linux" +"this workshop"	10	20	50%
Merging all 4 query modifications and removing duplicates (2 positive duplicates, 2 negative duplicates)	34	78	44%

ties. We also found that children's home pages often made reference to their `favorite` things, while academics often wrote of their research `interests`. Thus, our method for finding multiple query modifications appears to identify language trends that occur in subgroups of a category.

While it is unlikely to derive a set of query modifications that has 100% recall and 100% precision, we believe that our work supports the use of multiple query modifications for increasing recall while maintaining a desired precision.

5.2 "Eigenqueries" & Data Deflation

One key facet of our proposed method is the dataset deflation step, which eliminates all true positives from the training data, and retrains the SVM. The procedure in Table 1 performs a constrained and limited search for an effective query modification. To make an analogy, the query modification found in the step is akin to an "eigenquery" in that it spans a large portion of the training data.

By removing the true positives from the dataset, at the next iteration we force a procedure to find query modifications that complement the previously found query modifications. Continuing the analogy, data deflation is similar to factoring out an eigenvector from a matrix so that the next eigenvector can be found.

In this way, our proposed method attempts to span a large portion of document space by identifying as many "eigenqueries" as possible.

5.3 Dimensionality Reduction

Another benefit of our approach is that it performs dimensionality reduction in at least four different ways. During preprocessing, we eliminate n-grams that are either too common or too uncommon. While it is possible for this preprocessing to eliminate features that are key to identifying a subset of the positive examples, we believe that reducing the feature space via our preprocessing is warranted considering that without it the feature space would consist of hundreds of thousands of words or phrases.

Our search for sensitive positive exemplars is also assisted by the use of SVMs as a nonlinear model because the discovered support vectors (exemplars with positive Lagrange multipliers) are exactly the training points whose removal would alter the solution. Thus, by only searching sensitivities for positive support vectors, we reduce the number of documents which must be considered.

The sensitivity analysis also provides a ranking of features in the input space. By only considering the largest d components, we can efficiently search for the optimal d-dimensional query modification that uses those d input features.

Finally, and as previously mentioned, the dataset deflation step eliminates a significant portion of the training data at each step, which has the effect of speeding training time for the SVM and simplifying the document space for which we are searching.

5.4 Summary & Conclusions

Two alternative extremes for identifying query modifications would use linear classifiers and an exhaustive search in the feature space. The former method fails because it cannot work effectively in the linearly inseparable case. The latter approach is problematic because of the exponential number of binary classifiers that would need to be considered. Our approach exploits the underlying regularity of the documents that make up the WWW. We can discover higher-order correlations in the form of query modifications that contain multiple terms—even with a linearly inseparable feature space—but we can also guarantee that our procedure will terminate in a reasonable amount of time.

It is also interesting to compare our approach to a "straw man" approach represented by a more aggressive procedure that attempts to construct a large set of ridiculously complicated query modifications, each of which identifies only one or two pages. Such a set essentially acts as a lookup table for the training data. In this case, one would have little hope of generalizing to real-world data. By enforcing dimensionality reduction at many steps, we find query modifications that appear to generalize to real-world data despite the relatively small size of our training sets.

This last issue is especially important for very restricted searches, say, one in which we are trying to find the personal home page of a particular person (with a very common name) instead of one about a particular topic. By restricting the complexity of our query modifications and striving for maximum recall, our approach of using multiple query modifications can further increase the likelihood of finding very narrow search results which might normally be ranked beyond the limit of a given search engine.

6. ACKNOWLEDGEMENT

We thank Frans Coetzee for insightful discussions, Sven Heinicke and Andrea Ples for help in performing experiments, and the anonymous reviewers for many helpful comments.

7. ADDITIONAL AUTHORS

Additional authors: C. Lee Giles (Penn State University, email: `giles@ist.psu.edu`).

8. REFERENCES

[1] W. Cohen. Fast effective rule induction. In *Proc. of the Twelfth Int. Conf. on Machine Learning*, pages 115–123. Morgan Kaufmann, 1995.

[2] G. W. Flake and S. Lawrence. Efficient SVM regression training with SMO. *Machine Learning*, 46(1/3):271–290, 2002.

[3] S. Gauch, G. Wang, and M. Gomez. ProFusion: Intelligent fusion from multiple, distributed search engines. *Journal of Universal Computer Science*, 2(9), 1996.

[4] E. J. Glover, S. Lawrence, W. P. Birmingham, and C. L. Giles. Architecture of a metasearch engine that supports user information needs. In *Eighth International Conference on Information Knowledge Management (CIKM'99)*, pages 210–216, Kansas City, MO, November 1999. ACM Press.

[5] E. J. Glover, S. Lawrence, M. D. Gordon, W. P. Birmingham, and C. L. Giles. Recommending web documents based on user preferences. In *ACM SIGIR 99 Workshop on Recommender Systems*, Berkeley, CA, August 1999. ACM Press.

[6] E. Glover, G. Flake, S. Lawrence, W. P. Birmingham, A. Kruger, C. L. Giles, and D. Pennock. Improving category specific web search by learning query modifications. In *Symposium on Applications and the Internet, SAINT*, San Diego, CA, January 8–12 2001.

[7] A. E. Howe and D. Dreilinger. SavvySearch: A meta-search engine that learns which search engines to query. *AI Magazine*, 18(2), 1997.

[8] T. Joachims. Text categorization with support vector machines: learning with many relevant features. In C. Nédellec and C. Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398, pages 137–142, Chemnitz, DE, 1998. Springer-Verlag, Heidelberg, DE.

[9] T. Joachims. Transductive inference for text classification using support vector machines. In *Proc. 16th International Conf. on Machine Learning*, pages 200–209. Morgan Kaufmann, San Francisco, CA, 1999.

[10] S. Lawrence and C. L. Giles. Context and page analysis for improved web search. *IEEE Internet Computing, July-August*, pages 38–46, 1998.

[11] J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1998.

[12] J. Platt. Using sparseness and analytic QP to speed training of support vector machines. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems 11*. MIT Press, 1999.

[13] E. Selberg and O. Etzioni. The MetaCrawler architecture for resource aggregation on the Web. *IEEE Expert*, (January–February):11–14, 1997.

[14] G. G. Towell and J. W. Shavlik. Extracting refined rules from knowledge-based neural networks. *Machine Learning*, 13:71, 1993.

[15] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, New York, 1995.

APPENDIX

All SVM experiments performed in this work were done with NODElib, which can be downloaded from `http://www.neci.nec.com/homepages/flake/nodelib/html/`.