Kernel Methods for Document Filtering

Nicola Cancedda[‡], Nicolò Cesa-Bianchi[*], Alex Conconi[*], Claudio Gentile[§], Cyril Goutte[‡], Thore Graepel[†], Yaoyong Li[†], Jean-Michel Renders[‡], John Shawe-Taylor[†], Alexei Vinokourov[†]

[§]CRII
Università dell'Insubria, Italy
<FamilyName>@dsi.unimi.it

[†]Department of Computer Science
Royal Holloway, University of London
<FirstName>@cs.rhul.ac.uk

[*]Dipartimento di Tecnologie dell'informazione
Università degli Studi di Milano
<FamilyName>@dti.unimi.it

[‡]Xerox Research Centre Europe
<FirstName>.<FamilyName>@xrce.xerox.com

4th February 2003

Abstract

This paper describes the algorithms implemented by the KerMIT consortium for its participation in the TREC 2002 Filtering track. The consortium submitted runs for the routing task using a linear SVM, for the batch task using the same SVM in combination with an innovative threshold-selection mechanism, and for the adaptive task using both a *second-order perceptron* and a combination of SVM and *perceptron with uneven margin*. Results seem to indicate that these algorithm performed relatively well on the extensive TREC benchmark.

1. Introduction

The KerMIT IST European project is concerned with the investigation of kernel methods for applications related to the categorization, retrieval, clustering and ranking of text documents and of images[1]. The KerMIT consortium participated in the TREC 2002 Filtering track as a means of evaluating the methods developed within the project on a large-scale benchmark. Six runs were submitted, out of which four for the adaptive task and one each for the batch and for the routing task. As the objective of our participation was the comparison of different techniques, submitted runs are actually issued from several different systems:

- Runs 'af1' and 'af2' were obtained using a variant of the "second-order perceptron" (Cesa-Bianchi et al. 2002)(Section 3);
- Runs 'af3' and 'af4' were obtained using a combination of the SVM algorithm and the Perceptron Algorithm with Uneven Margin (Li et al. 2002)(Section 4);
- Runs 'bf2' and 'rr2' are the output of SVMs, the former in combination with an improved threshold-selection mechanism (Section 5).

The paper is organised as follows. Section 2 sketches the data preparation process. Sections 3 to 5 describe the systems listed above in turn, together with the performance achieved on the respective tasks. Section 6 presents some additional experiments on intersection topics. Section 7 contains some concluding remarks.

2. Data preprocessing

Before turning to the description of the individual systems, we will outline the data preparation process

[1]More information on the KerMIT IST project is available from the project website: http://www.euro-kermit.org.

that we followed. The original NewsML files are preprocessed in the following way:

- The "title" and the "text" portions of the files are extracted and cleaned from tags;
- Each file is tokenised into words using a finite-state based tokeniser;
- All digit characters are replaced with a single special character;
- Stopwords are removed;
- A dictionary file is built associating a numeric code with each token occurring at least three times in the training set. Terms occurring only once or twice are ignored;
- For every document body and every title a sparse term vector is built;
- The title and the body vector for each document are combined giving double weight to the title;
- All document vectors are finally modified according to a tf*idf weighting scheme and normalised to unit norm.

In the case of the batch and the routing runs, idf is computed for every term based on the training set only. Idf weights for all other terms are set to zero. In other words, only terms in the test documents which also occur in the training set are considered.

In the case of the adaptive filtering runs, idf weights are initialized on the training corpus in the same way as for the batch and the routing runs. However, as more and more test documents come in, idf weights are updated. Similarly for the lexicon, new lexical items are added to the dictionary as soon as the number of occurrences in the training set combined with the test set up to the document itself reaches a threshold of three.

The adopted tf*idf weighting is the usual log-log one:

$$w_{i,j} = (1 + \log(tf_{i,j}))\ \log\left(\frac{N}{df_i}\right)$$

where $tf_{i,j}$ is the number of occurrence of term i in document j, N is the total number of documents in the collection, and df_i is the number of documents containing the term i.

The very limited availability of positive examples called for ways to take advantage of the topic descriptions as well. We considered two alternative approaches, one consisting in building an additional positive training example from the descriptions and another consisting in building a vector from the description and then adding to each document, as a new feature, the value of its (cosine) similarity with the description. The second alternative proved superior and was thus retained for the batch and the routing tasks, whereas time constraints did not allow its adoption in the adaptive case.

3. The second-order perceptron algorithm for adaptive filtering

In this section we describe the first of the two algorithms used in the adaptive filtering track. This first algorithm is defined by a pair $(\boldsymbol{w}, \tau)$, where $\boldsymbol{w} \in \mathbb{R}^N$ is the profile vector and $\tau \in \mathbb{R}$ is the relevance threshold. A document $\boldsymbol{x} = (x_1, \ldots, x_N) \in \mathbb{R}^N$ is judged relevant if and only if the *margin* $\boldsymbol{w}^\top \boldsymbol{x}$ (i.e., the inner product between $\boldsymbol{w}$ and $\boldsymbol{x}$) is not smaller than the threshold τ.

We model the filtering problem under the assumption that relevance judgments are generated using an unknown probabilistic linear function. Assuming all documents $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots$ are normalized such that $\|\boldsymbol{x}_t\| = 1$ for all $t \geq 1$, the relevance of $\boldsymbol{x}_t$ is given by a $\{-1, 1\}$-valued random variable Y_t (where $Y_t = 1$ means "relevant") such that there exists a fixed and unknown "target" profile vector $\boldsymbol{u} \in \mathbb{R}^N$, $\|\boldsymbol{u}\| = 1$, for which $\mathbb{E} Y_t = \boldsymbol{u}^\top \boldsymbol{x}_t$ for all $t = 1, 2, \ldots, n$, where $\mathbb{E}$ indicates the expected value. Hence $\boldsymbol{x}_t$ is relevant with probability $(1 + \boldsymbol{u}^\top \boldsymbol{x}_t)/2 \in [0, 1]$. The random variables $Y_1, Y_2, \ldots$ are assumed to be independent, whereas we do not make any assumption on the way the sequence $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots$ of documents is generated.

The profile vector of our filtering rule is a (biased) estimator of the target profile $\boldsymbol{u}$ constructed as follows. Let S_t be the matrix whose columns are the forwarded documents after the first t time steps and let $\boldsymbol{Y}_t$ be the vector of corresponding observed relevance labels. Note that $\mathbb{E} \boldsymbol{Y}_t = S_t^\top \boldsymbol{u}$ holds. Drop the index t for clarity and consider the least squares estimator $(S\,S^\top)^\dagger S \boldsymbol{Y}$ of $\boldsymbol{u}$, where $(S\,S^\top)^\dagger$ is the pseudo-inverse of $S\,S^\top$. For all $\boldsymbol{u}$ belonging to the column space of S, this is an unbiased estimator of $\boldsymbol{u}$, that is

$$\mathbb{E}\left[(S\,S^\top)^\dagger S \boldsymbol{Y}\right] = (S\,S^\top)^\dagger S\, \mathbb{E}\boldsymbol{Y} = (S\,S^\top)^\dagger S\,S^\top \boldsymbol{u} = \boldsymbol{u}\ .$$

To remove the assumption on $\boldsymbol{u}$, we make $S\,S^\top$ full rank by adding the identity matrix I. This also allows us to replace the pseudo-inverse with the standard inverse, obtaining the biased estimator

$$(I + S\,S^\top)^{-1} S\,\boldsymbol{Y} \tag{3.1}$$

with expectation $\mathbb{E}\big[(I + S\,S^\top)^{-1} S\,\boldsymbol{Y}\big] = \boldsymbol{u} - (I + S\,S^\top)^{-1}\boldsymbol{u}$ (this immediately follows from the matrix identity $(I + S\,S^\top)^{-1}\,S\,S^\top = I - (I + S\,S^\top)^{-1}$). Estimator (3.1) is a "sparse" variant of the ridge regression estimator (Hoerl and Kennard 1970), where the sparseness is due to the fact that we only store in S the documents for which we have a relevance label (i.e., those that were forwarded).

We use a variant of (3.1) that tries to estimate directly the margin $\boldsymbol{u}^\top\boldsymbol{x}$ rather than estimating $\boldsymbol{u}$. More precisely, we estimate $\boldsymbol{u}^\top\boldsymbol{x}$ with the quantity $\boldsymbol{W}^\top\boldsymbol{x}$, where the profile vector $\boldsymbol{W}$ is defined by

$$\boldsymbol{W} = (I + S\,S^\top + \boldsymbol{x}\,\boldsymbol{x}^\top)^{-1} S\,\boldsymbol{Y}\ . \tag{3.2}$$

Using the Sherman-Morrison formula, we can then write out the expectation of $\boldsymbol{W}^\top\boldsymbol{x}$ as

$$\mathbb{E}\Big[\boldsymbol{W}^\top\boldsymbol{x}\Big] = \frac{\boldsymbol{u}^\top\boldsymbol{x} - \boldsymbol{u}^\top(I + S\,S^\top)^{-1}\boldsymbol{x}}{1 + \boldsymbol{x}^\top(I + S\,S^\top)^{-1}\boldsymbol{x}}$$

which holds for all $\boldsymbol{u}$, $\boldsymbol{x}$, and all matrices S. Comparing the bias of $\boldsymbol{W}$ to the bias of (3.1) in estimating the margin, we may observe that $\boldsymbol{W}$ introduces a multiplicative bias whose effect is to shrink the expectation of the margin $\boldsymbol{W}^\top\boldsymbol{x}$. In fact, the term $\boldsymbol{x}^\top(I + S\,S^\top)^{-1}\boldsymbol{x}$ is always nonnegative due to the positive definiteness of $(I + S\,S^\top)^{-1}$. In the experiments $\boldsymbol{W}$ turns out to perform better than (3.1), though at present we do not have a convincing theoretical explanation of this fact. This algorithm can be turned into an equivalent dual form, which is needed when we use the feature expansion facility provided by the kernel functions. As a matter of fact, since the document vectors $\boldsymbol{x}$ in the dataset at hand have a large number of components, we found it convenient to run the dual form even without kernels. As a final remark, we note that $\boldsymbol{W}$ is strongly related to the second-order Perceptron algorithm for binary classification introduced in (Cesa-Bianchi et al. 2002).

We now move on to the choice of the threshold τ. A possible route, which has been followed in (Cesa-Bianchi et al. 2003), is to approximately compute for each document $\boldsymbol{x}$ an interval centered on $\boldsymbol{W}^\top\boldsymbol{x}$, around which $\boldsymbol{u}^\top\boldsymbol{x}$ falls with high confidence. Then, whenever $\boldsymbol{x}$ is such that the left-hand border of the interval for $\boldsymbol{W}^\top\boldsymbol{x}$ is negative, $\boldsymbol{x}$ is judged relevant and forwarded. This approach corresponds to setting τ to a negative value chosen as a function of both $\boldsymbol{x}$ and the current profile. The primary effect of this approach is to boost recall at the expense of precision, resulting in an increased net performance when precision and recall are scored the same (see the results in (Cesa-Bianchi et al. 2003)). However, this goes exactly against the TREC evaluation measures which put an emphasis on precision. To reduce recall we then decided to set τ to a positive (instead of negative) value in the interval $[0, 1/10]$. This choice reduces dramatically the number of forwarded documents, thus pushing precision up, but it also slows down the convergence of the profile to the target $\boldsymbol{u}$, which results in a decrease of precision. Hence, unlike the one based on confidence intervals, this setting of τ, needs a reasonably good profile to start with.

3.1. TREC results

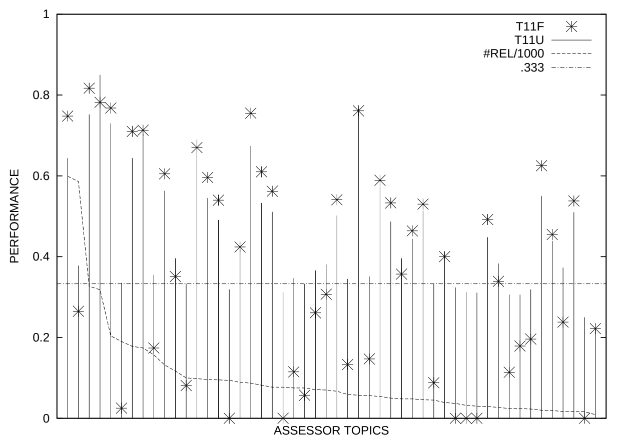
Based on the above discussion, we set the threshold τ to 0.1 for the first run (KerMITT11af1) and to 0.05 for the second run (KerMIT11af2). We then built an initial profile for each topic using a training set with 4 positive examples (three provided with the data plus one we built using the topic description). The table below shows the average results over TREC11 topics for KerMITT11af1 and KerMITT11af2 runs.

Topics range	KerMITT11af1		KerMITT11af2	
	T11U	T11F	T11U	T11F
Assessor	0.456	0.378	0.459	0.376
Intersection	0.323	0.049	0.310	0.047
All	0.389	0.213	0.385	0.211
Assessor ext. reljs	0.473	0.395	0.475	0.392

The last row contains results for assessor topics with the extended relevance judgments provided after the TREC conference. The positive threshold helped to control the number of false positives for the assessor topics (R101–R150), on which af1 and af2 obtained relatively good results. This did not happen on the intersection topics (R151–R200), where the average normalized linear utility measure (T11U) turns out to be slightly worse than the one of the trivial algorithm which retrieves nothing, and the $F_{0.5}$ measure (T11F) is remarkably low.

In the Reuters corpus with TREC11 topics only a small amount of positive examples is available for each topic. This leads to two problems:

- explorative predictions (i.e. when the algorithm judges a document as relevant because it is interested in obtaining the true label rather than scoring a good prediction) are not useful, because most of the times the obtained label is negative and conveys no information.



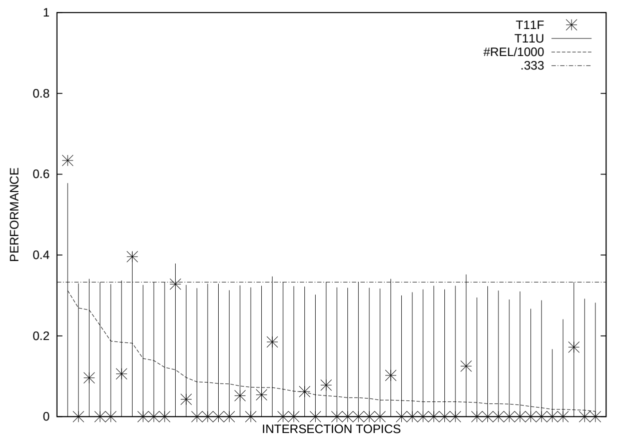


Figure 3.1. Performance for KerMITT11af1 on assessor (top) and intersection (bottom) topics.

- wrong predictions are very likely to be false positives, which badly affect the TREC evaluation measures.

Figure 3.1 plots the performance for the run KerMITT11af1 on each of the assessor and intersection topics. The topics (x-axis) are ordered according to their decreasing frequency. The horizontal line at .333 marks the T11U measure of the trivial algorithm which retrieves nothing. As the plots clearly show, the T11F measure exhibits a dramatic drop on almost all of the intersection topics (except a few of the most frequent ones). The T11U measure does not drop so badly on the intersection topics, even though its average value remains slightly below the .333 threshold.

Our future plan is to use a separate self-tuning threshold for each topic, using the current number of false positives as an indicator of whether it is better to bias the threshold towards positive or negative predictions.

4. SVM plus the Perceptron Algorithm with Uneven Margin for adaptive filtering

This section describes the second system implemented for the adaptive filtering task. In this system we adapted the Support Vector Machine (SVM) for the adaptive filtering. The SVM is quite suitable and successful in batch filtering (Lewis 2001), which is essentially normal text categorization (Joachims 1997). However, the adaptive filtering task in TREC is different from batch filtering in several aspects.

1. Only a small number of positive examples and a great number of unjudged documents are provided to create an initial profile [2].

2. The profile can be updated based on the retrieved documents for adaptive filtering, whereas it is fixed when it is applied to the test documents in batch filtering.

3. There are three kinds of documents in adaptive filtering, i.e. the positive, the negative and the unjudged, which can be employed to update the profile. The system can achieve good performance if it takes into account the different contributions of the documents toward the profile.

Our system deals with these issues by the following strategies.

1. The Gram-Schmidt algorithm (Cristianini et al. 2002) was employed to choose the negative examples from the unjudged training documents, i.e. these unjudged documents which are the furthest away from the given positive examples. This reduces the likelihood of choosing an example that is actually a positive example.

2. Use a fast and effective on-line version of the SVM, the Perceptron Algorithm with Uneven Margins (PAUM) (Li et al. 2002), to update the profile by the latest retrieved document.

3. Introduce several so-called margin parameters into the SVM as well as the PAUM to balance the contributions of different kinds of documents towards the profile.

[2]It is noted that the so-called one-class SVM can learn only from positive examples. However, the experiments have shown that the normal SVM using both the positive and negative examples can achieve much better performance in text categorization than the one-class SVM.

In detail, the algorithm used in our system is as follows.

- **Require**: n_g — the number of negative examples chosen for training.
  Require: γ_p and γ_n — the marginal parameters in the SVM for the positive and negative examples, respectively.
  Require: τ_p, τ_n and τ_u — the marginal parameters in the PAUM for the positive, the negative and the unjudged documents, respectively.
  Require: t — the threshold to retrieve the test document.
- **Training**
  - Training set — As we were given some positive training examples, we choose the negative training examples from the unlabeled training documents by the Gram-Schmidt algorithm. In detail, we apply the Gram-Schmidt algorithm to the given positive examples and compute the residual norms of the unjudged training documents. We then chose the first n_g documents with the largest residual norms as the negative examples.
  - Training method — After obtaining the training set, solve the corresponding SVM with uneven marginal parameters γ_p and γ_n:

```
minimize$_{\mathbf{w},\xi}$ $\langle \mathbf{w}, \mathbf{w} \rangle + C \sum_{i=1}^{l} \xi_i$
subject to
$\langle \mathbf{w}, \mathbf{x}_i \rangle + \xi_i \geq \gamma_p$ for the positive examples
$\langle \mathbf{w}, \mathbf{x}_i \rangle - \xi_i \leq -\gamma_n$ for the negative examples
$\xi_i \geq 0$      for $i = 1, ..., l$
```

- **Test**
  1. Apply the profile $\mathbf{w}$ to the test documents sequentially. For the test document $\mathbf{d}_i$, apply the profile $\mathbf{w}$ to the $\mathbf{d}_i$. If $\langle \mathbf{w}, \mathbf{d}_i \rangle > t$ then the document $\mathbf{d}_i$ is retrieved and is used to update the profile as shown in 2.
  2. Update the profile $\mathbf{w}$ using the marginal perceptron algorithm:

```
Let $\tau = \tau_p$ and $y_i = +1$, if the $d_i$ is relevant.
Let $\tau = \tau_n$ and $y_i = -1$, if the $d_i$ is irrelevant.
Let $\tau = \tau_u$ and $y_i = -1$, if the $d_i$ is unjudged.
while $y_i \langle \mathbf{w}, \mathbf{d}_i \rangle \leq \tau$
    $\mathbf{w} \leftarrow \mathbf{w} + y_i \mathbf{d}_i$
endwhile
```

It is worth noting that, in the above algorithm, the profile is updated by using only the latest retrieved document d_i. We will discuss a variation of the algorithm later, which employs all the currently retrieved documents to update the profile.

4.1. TREC results

In order to actually apply the described algorithm to the TREC2002 dataset, the values of the parameters n_g, γ_p, γ_n, τ_p, τ_n, τ_u, and t need to be fixed. As part of previous experiments in the KerMIT project –before any actual work on the TREC2002 evaluation started– we had used part of the RCV1 corpus with the original reuters categories –as in the TREC2001 evaluation– to evaluate this same algorithm. More precisely, the training set consisted of the documents of the first 12 days, 20 through 31 August, 1996, and the test set was made of the documents of 30 days chosen from 1 October 1996 to 28 February, 1997. The topics considered in the dataset were those categories of the RCV1 classification scheme which had between 10 and 60 positive examples in the training set. Ten different parameter configurations were tried. The one which produced the best results was the following:

$$n_g = 80, \quad \gamma_p = 20, \quad \gamma_n = 5, \quad \tau_p = 10, \quad \tau_n = 2, \quad t = 5.0.$$

These values were retained for the KerMIT submissions based on this algorithm, namely KerMITT11af3 and KerMITT11af4, the assumption being that they were obtained through experiences in all comparable to a participation in the TREC2001 filtering track (see the TREC 2002 filtering track overview paper for a discussion of the issue of prior use of RCV1). It is indeed possible that results for those same runs would have been somewhat worse, had we used an entirely different document collection for chosing parameter values. Notice that a version of the algorithm without τ_u had been used in the previous experiments, given that the dataset we used did not contain any unjudged documents, and thus τ_u remained still to be assigned a value. It was decided to consider unjudged documents as "weekly negative" examples, and choose τ_u according to

$$\tau_u = -0.5 \frac{(pe_i + ne_i)}{(pe_i + ne_i + ue_i)},$$

where pe_i, ne_i and ue_i are the numbers of retrieved relevant, irrelevant and unjudged documents up to document i, respectively.

Finally, we adapted the threshold t in our system according to a simple rule whenever too few documents or too many unjudged documents were retrieved. The

run ID	Averaged over topics	**T11SU**	**T11F**	**Precision**	**Recall**
KerMITT11af3	Assessor topics	0.458	0.426	0.527	0.302
	Intersection topics	0.285	0.048	0.076	0.026
	Ass. top. (ext. reljs)	0.450	0.402	0.488	0.293
KerMITT11af4	Assessor topics	0.454	0.409	0.506	0.304
	Intersection topics	0.287	0.056	0.097	0.029
	Ass. top. (ext. reljs)	0.458	0.404	0.483	0.316

Table 1. The results of the two submitted runs. The last row for each run shows the result, on the assessor topics, with the extended relevance judgments provided after the TREC conference.

	Averaged over topics	**T11SU**	**T11F**	**Precision**	**Recall**
Run_1	Assessor topics	0.428	0.387	0.491	0.269
	Intersection topics	0.274	0.040	0.064	0.021
Run_2	Assessor topics	0.474	0.460	0.543	0.366
	Intersection topics	0.276	0.074	0.114	0.037

Table 2. The results of two additional runs, to be compared with the submitted run KerMITT11af3. Run_1 is obtained by choosing the negative training examples randomly. Run_2 is produced updating the profile by all the currently retrieved documents.

idea is that the system checks the threshold t everytime after a multiple of n_a test documents has been processed. In each check, if too few documents have been retrieved then the system decreases the threshold, otherwise, if too many unjudged documents have been retrieved, the system increases the threshold. We set $n_a = 10000$ in our system.

We applied the algorithm with the above settings to the dataset of TREC2002 adaptive filtering, and submitted two runs from this system, namely KerMITT11af3 and KerMITT11af4, with the initial values of the threshold t set to 5.0 and 4.5, respectively. The results for the two runs are listed in Table 1. We can see that the results are very different between the assessor topics (the first 50) and the intersection topics (the last 50).

4.2. Results of additional runs

In addition to the two submitted runs, we did several more experiments to test our algorithm. Table 2 lists the results of two additional runs.

One additional run was used to check the gain of the Gram-Schmidt algorithm in our system. This run had the same settings as those of KerMITT11af3, except that the negative examples for training were chosen randomly from the training documents exclusive of the three relevant documents. The results of this run, listed in the Run_1 of Table 2, show that selecting negative examples using the Gram-Schmidt algorithm yields significantly better results then selecting them at random.

In another additional run we obtained even better results than our two submitted runs from the system. In this run, we updated the profile by using all the currently retrieved documents, instead of only the latest retrieved document as we did in the two submitted runs. In detail, for each topic, when a new document was retrieved, we added it to the training set, and then apply the SVM with uneven margins to this updated training set to compute the new profile (i.e. the weight vector of SVM). The initial value of the threshold t in this run was set as 7.0, and other experimental settings are the same as those of KerMITT11af3. The result of this run is listed in Table 2, in the row marked as Run_2. It can be seen that, compared with the two submitted runs KerMITT11af3 and KerMITT11af4, we obtained better results in the Run_2 as we used more information to update the profile. On the other hand, Run_2 took much more processing time and needed more memory than the two submitted runs.

5. A new threshold-selection mechanism for the SVM for batch filtering

This section describes the method used for runs *bf2* and *rr2*, for the batch and the routing subtasks respectively.

Our choice for a basic classifier fell on the best scoring system of the TREC 2001 evaluation - the SVM^{light} package (Joachims 2002). We trained an SVM on the TREC preprocessed training corpus (see Section 2) and thus obtained predictions γ_i (distances from the separating hyperplane taken with the appropriate sign) of the labels for each document d_i from the training corpus. Note that training data were separable for all test queries and thus the sign of γ_i can be considered as the label.

The parameter C for SVM training was chosen based on machine learning theory (Vapnik 1998): $C_{opt} = ||R||^{-1}$ (where $R = \max ||x_i||$ is the radius of the ball containing training datapoints x_i and centered in the origin). As the data vectors were normalised C was set to 1.

Full cross-validation for determining the relative importance to be given to positive and negative examples in the training set (parameter j in SVM^{light}) was impractical given the size of the training set and the number of categories. We thus decided to use a fixed value, namely 7, which seemed to give results reasonably close to those obtained with cross-validation on some partial small-scale experiment using the training corpus only.

5.1. Finding threshold

We needed to find an appropriate threshold which would be optimal in some sense for one of the TREC 2002 evaluation measures - either `T11SU` or `T11F`. The approach undertaken was to approximate positive and negative datapoints distributions by two - "positive" and "negative" - Gaussians appropriately and then use a Monte-Carlo method to synthesize a new set of data with the same proportion of positive and negative examples as in the training corpus. Further, a threshold "optimal" in terms of the needed evaluation measure from the generated data was induced (Fig. 5.1). The Gaussians means $m_\pm$ and standard deviations $\sigma_\pm$ were estimated from appropriate sets of positive and negative γ_i's.

The pseudocode is given below:

```
Input: $\gamma_i$, $i = 1, .., M$
    Distances of training documents from the separating hyperplane.
Output: threshold $b$
$b$ = function Two_Gaussian_find_threshold($\{\gamma_i\}$);
    $N_+ = |\{\gamma_i : \gamma_i > 0\}|$;    $N_- = |\{\gamma_i : \gamma_i < 0\}|$;
    $(m_+, \sigma_+)$ = mean_and_std_dev($\{\gamma_i : \gamma_i > 0\}$);
    $(m_-, \sigma_-)$ = mean_and_std_dev($\{\gamma_i : \gamma_i < 0\}$);
    $\Gamma_+$ = sample_normal($m_+, \sigma_+, 5000 \frac{N_+}{N_-}$ points);
    $\Gamma_-$ = sample_normal($m_-, \sigma_-, 5000 (1 - \frac{N_+}{N_-})$ points);
    $b = \arg\max_{\gamma_i^+ \in \Gamma_+} T11SU(\gamma_i^+, \Gamma_+, \Gamma_-)$;
or $T11F$
end
```

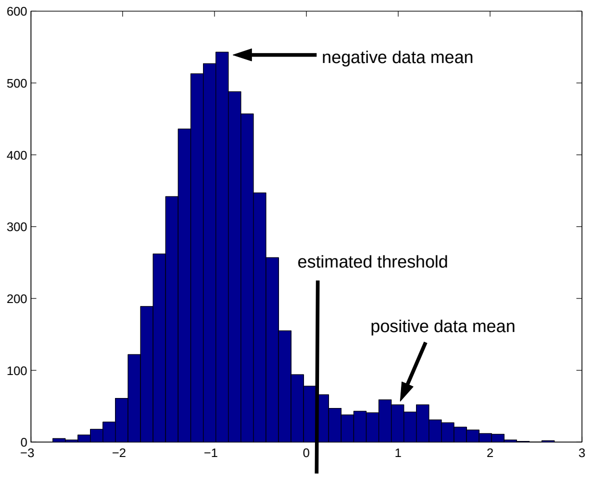


Figure 5.1. An illustration of the method that was chosen to find a threshold value.

Here function $T11SU(\gamma_i^+, \Gamma_+, \Gamma_-)$ computes the `T11U` score for the threshold γ_i^+ given predictions for the "relevant" Γ_+ and "irrelevant" Γ_- sets of documents.

5.2. TREC results

Results for the TREC batch and routing tasks are shown in Table 3.

The comparative analysis reveals that the method performs relatively well w.r.t. other submissions on the first 50 (assessor) topics whilst on the intersection topics its performance is rather uneven.

Table 4 displays the number of topics on which the method exhibited the best, the 2nd best, etc. result compared to the other 15 participants. Intersection topics turned out to be harder for the `Two_Gaussian` than for other methods.

6. Why are intersection topics so hard?

Intersection topics turned out to be extremely difficult, both in the adaptive and in the batch settings. In order to gain some insight on the reasons for this, some additional experiments were run after the TREC conference. For six of the fifty intersection topics, representative of different category sizes, the *intersecting* Reuters categories from which they were obtained were identified. These were the following:

run ID	Averaged over topics	**T11SU**	**T11F**	**AvgP**
KerMITT11bf2/rr2	Assessor topics	0.505	0.495	0.427
	Intersection topics	0.245	0.101	0.061
KerMITT11bf2	Assessor topics	0.362	0.121	-
std. SVM thresh. selection	Intersection topics	0.330	0.010	-

Table 3. Results for the batch and routing tasks. The bottom part contains the results achieved when the "standard" SVM threshold selection is adopted instead of the proposed one.

	best	2nd best	3rd best	2nd worst	worst
# Assessor topics	16	10	8	1	0
# Intersection topics	11	4	9	11	9

Table 4. Relative performance for assessor and for intersection topics.

Query	Categories		Query	Categories	
R151	C31	GSCI	R157	C331	GPOL
R153	C11	M143	R199	C31	E13
R156	M14	E513	R200	C41	GOBIT

SVMs were trained independently for each of the intersecting Reuters categories, using the threshold selection mechanism described in Section 5, and tested using the TREC split. We computed the performance of the "intersection classifier" obtained by combining the classifiers for the intersecting categories using a logical AND. Results are presented in Table 5. In all cases, the performance of the "intersection classifier" is largely inferior to what one would expect from the performance on the intersecting Reuters categories. This shows that the hindsight provided by the query composition does not help in designing a better classifier for the query, even though the classifiers for each components have relatively good performance. This suggest that the documents in the intersection are atypical in at least one of the intersecting categories.

In order to verify this, we investigated the distribution of the output of the SVM classifers, $f(\boldsymbol{x}) = \sum \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x})$. By analogy with the large margin argument, we call this the "margin" of an example. In Figure 6.1, we display, for each category, the distribution of the margins of the documents that are 1/ in the intersection (dashed) and 2/ in this category, but not in the intersection (solid). For a perfect classifiers, all margins should be on the right-hand side of the threshold (dotted)—margins on the left-hand side indicate misclassified examples. Figure 6.1 shows that in most cases, the distribution of margins for intersection documents is shifted towards the left, indicating that the intersection documents tend to be misclassified, or in other words, that these documents are either "atypical" for the category, or, at least, harder to learn for the SVM categoriser. A more speculative conjecture would be that intersections contain mostly annotation errors. For query R151, for example, 22 relevant documents represent only 1% of category GSCI, and about 0.06% of C31.

7. Conclusions

The algorithms developed in the context of the KerMIT project seem to perform relatively well on the TREC filtering benchmark. Nevertheless, at least a couple of points remain to be clarified. The first is the very uneven performance on the assessor topics and on the intersection topics. Despite falling short of providing an exhaustive explanation, our experiments show at least that documents relevant to intersection topics tend to be peripheral within the intersecting categories. The second point is the lack of improvement in performance when more complex kernels, such as polynomial kernels of degree higher than one or radial basis function kernels, are used. This is somewhat in contradiction with previous findings in document categorisation (Joachims 1997), which indicated that such kernels do indeed perform better than the basic inner product.

8. Acknowledgments

The authors would like to thank Eric Gaussier, Jaz Kandola, Craig Saunders, and Yoram Singer for the many useful suggestions. The work described in this paper has been supported by the European Commission through the IST Programme under Contract IST-2000-25431 (KerMIT).

References

Cesa-Bianchi, N., A. Conconi, and C. Gentile (2002). A second-order perceptron algorithm. In *Proceedings of the 5th Annual Conference on*

R151	C31		GSCI		C31 AND GSCI	
relevant	20015	16186	1265	923	0	22
irrelevant	11958	674982	225	720728	0	723119
T11F	0.6099		0.7763		0	
R153	C11		M143		C11 AND M143	
	positive	negative	positive	negative	positive	negative
relevant	9265	12292	18254	1376	0	37
irrelevant	8407	693177	2621	700890	33	723071
T11F	0.5022		0.8850		0	
R156	E513		M14		E513 AND M14	
relevant	1503	565	70382	5750	7	65
irrelevant	351	720722	4973	642036	7	723062
T11F	0.7924		0.9321		0.2734	
R157	C331		GPOL		C331 AND GPOL	
relevant	482	596	40856	9726	3	34
irrelevant	143	721920	20437	652122	10	723094
T11F	0.6736		0.6907		0.1685	
R199	C31		E13		C31 AND E13	
relevant	20015	16186	4680	922	35	80
irrelevant	11958	674982	1489	716050	185	722841
T11F	0.6099		0.7728		0.1759	
R200	C41		GOBIT		C41 AND GOBIT	
relevant	8308	1770	87	684	7	79
irrelevant	2559	710504	24	722346	19	723036
T11F	0.7758		0.3580		0.1842	

Table 5. Confusion matrices and T11F scores for six pairs of intersecting classifiers and their intersections.

Computational Learning Theory, LNAI 2375, pp. 121–137. Springer.

Cesa-Bianchi, N., A. Conconi, and C. Gentile (2003). Margin-based algorithms for information filtering. In *Advances in Neural Information Processing Systems 15*. MIT Press.

Cristianini, N., J. Shawe-Taylor, and H. Lodhi (2002). Latent semantic kernels. *Journal of Intelligent Information System 18*(2/3), 127–152.

Hoerl, A. and R. Kennard (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics 12*, 55–67.

Joachims, T. (1997). Text categorization with support vector machines: Learning with many relevant features. Technical report, Universitaet Dortmund.

Joachims, T. (2002). SVM^{light} - Support Vector Machine. http://svmlight.joachims.org.

Lewis, D. D. (2001). Applying support vector machines to the trec-2001 batch filtering and routing tasks. In *Proceedings of the Tenth Text REtrieval Conference (TREC 2001)*, pp. 286–292.

Li, Y., H. Zaragoza, R. Herbrich, J. Shawe-Taylor, and J. Kandola (2002). The perceptron algorithm with uneven margins. In *Proceedings of ICML 2002*, pp. 379–386.

Vapnik, V. (1998). *Statistical Learning Theory.* Chichester, UK: Wiley.

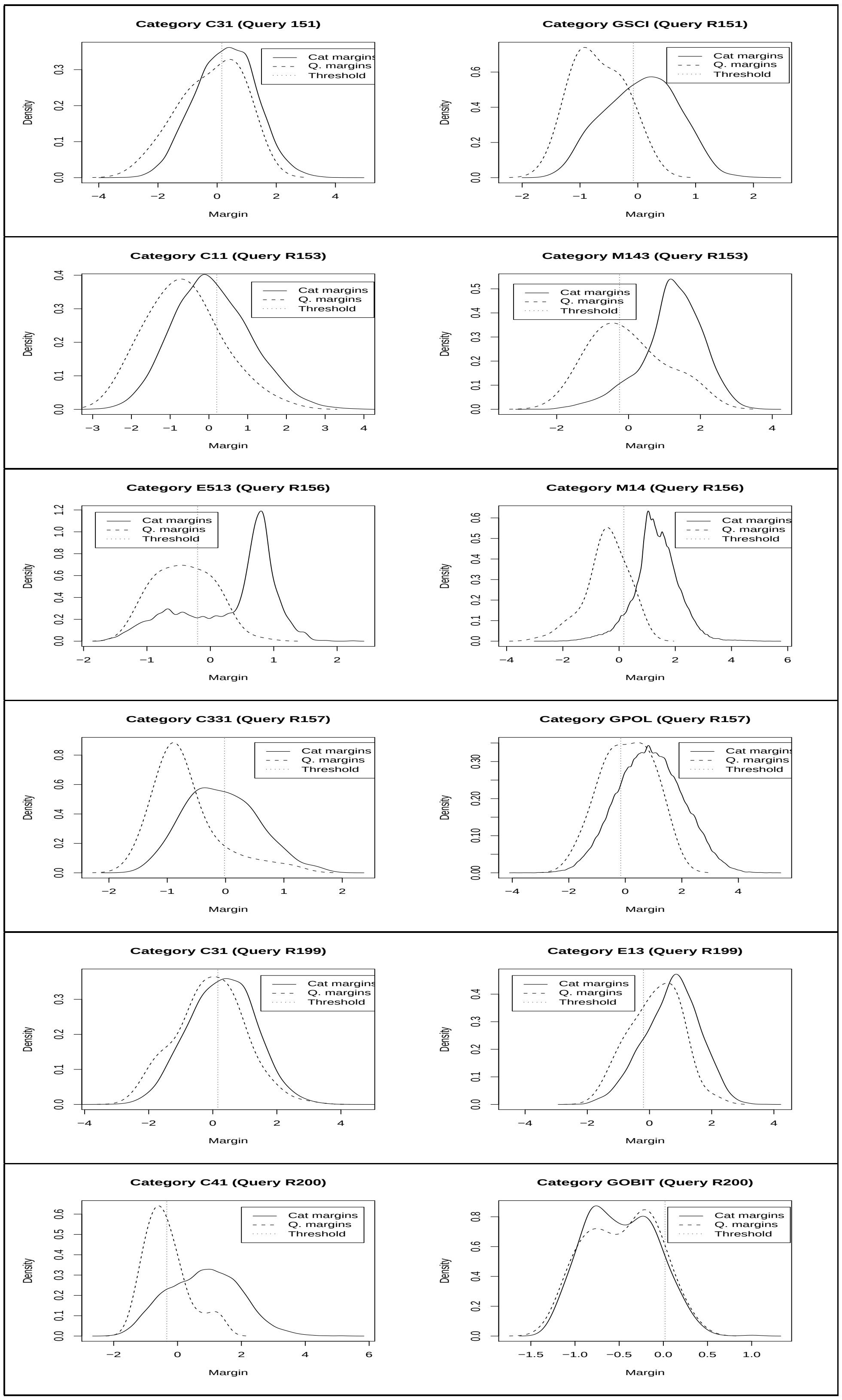


Figure 6.1. Distributions of the margins of the documents within the intersections (dashed lines) and within the category but out of the intersection (solid lines) with respect to the hyperplanes trained independently for the intersecting categories (dotted line).