World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:2, No:9, 2008

Meta-Classification using SVM Classifiers for Text Documents

Daniel I. Morariu, Lucian N. Vintan, and Volker Tresp

***Abstract*—**Text categorization is the problem of classifying text documents into a set of predefined classes. In this paper, we investigated three approaches to build a meta-classifier in order to increase the classification accuracy. The basic idea is to learn a meta-classifier to optimally select the best component classifier for each data point. The experimental results show that combining classifiers can significantly improve the accuracy of classification and that our meta-classification strategy gives better results than each individual classifier. For 7083 Reuters text documents we obtained a classification accuracies up to 92.04%.

***Keywords*—**Meta-classification, Learning with Kernels, Support Vector Machine, and Performance Evaluation.

I. Introduction

WHILE more and more textual information is available online, effective retrieval is difficult without good indexing and summarization of document content. Document categorization is one solution to this problem. The task of document categorization is to assign a user defined categorical label to a given document. In recent years a growing number of categorization methods and machine learning techniques have been developed and applied in different contexts.

Documents are typically represented as vectors in a features space. Each word in the vocabulary is represented as a separate dimension. The number of occurrences of a word in a document represents the value of the corresponding component in the document's vector.

In this paper we investigate some strategies for combining classifiers in order to improve the classification accuracy. We used classifiers based on Support Vector Machine (SVM) techniques. They are less vulnerable to degrade with an increasing dimensionality of the feature space, and have been shown effective in many classification tasks. The SVM is actually based on learning with kernels and support vectors.

We combine multiple classifiers hoping that the classification accuracy can be improved without a significant increase in response time. Instead of building only one highly accurate specialized classifier with much time and effort, we build and combine several simpler classifiers.

Several combination schemes have been described in the literature [1]. A usually approach is to build individual classifiers and later combine their judgments to make the final decision. Another approach, which is not so commonly used because it suffers from the "curse of dimensionality" [3], is to concatenate features from each classifier to make a longer feature vector and use it for the final decision. Anyway, meta-classification is effective only if it classifier synergies can be exploited.

In previous studies combination strategies were usually ad hoc and are strategies like majority vote, linear combination, winner-take-all [1], or Bagging and Adaboost [2]. Also, some rather complex strategies have been suggested; for example in [3] a meta-classification strategies using SVM [4] is presented and compared with probability based strategies.

Section 2 contains prerequisites for the work that we present in this paper. In sections 3 and 4 we present the methodology used for our experiments. Section 5 presents the experimental framework and section 6 presents the main results of our experiments. The last section debates and concludes on the most important obtained results and proposes some further work.

II. Support Vector Machine

The Support Vector Machine (SVM) is a classification technique based on statistical learning theory [5], [6] that was applied with great success in many challenging non-linear classification problems and on large data sets.

The SVM algorithm finds a hyperplane that optimally splits the training set. The optimal hyperplane can be distinguished by the maximum margin of separation between all training points and the hyperplane. Looking at a two-dimensional problem we actually want to find a line that "best" separates points in the positive class from points in the negative class. The hyperplane is characterized by a decision function like:

$$f(x) = \mathrm{sgn}\left(\left\langle \mathbf{w}, \mathbf{\Phi}(x) \right\rangle + b\right) \quad (1)$$

where **w** is the weight vector, orthogonal to the hyperplane, "*b*" is a scalar that represents the margin of the hyperplane, "*x*" is the current sample tested, "*Φ(x)*" is a function that transforms the input data into a higher dimensional feature space and $\langle \cdot, \cdot \rangle$ representing the dot product. *Sgn* is the sign function. If **w** has unit length, then <**w,** Φ*(x)*> is the length of Φ*(x)* along the direction of **w**. Generally **w** will be scaled by ||**w**||. In the training part the algorithm needs to find the normal vector "**w**" that leads to the largest "*b*" of the hyperplane.

Manuscript received September 27, 2006.

D. Morariu is with the Faculty of Engineering, "Lucian Blaga" University of Sibiu, Computer Science Department, E. Cioran Street, No. 4, 550025 Sibiu, Romania (phone: 40/0740/092202; e-mail: daniel.morariu@ulbsibiu.ro).

L. Vintan is with the Faculty of Engineering, "Lucian Blaga" University of Sibiu, Computer Science Department, E. Cioran Street, No. 4, 550025 Sibiu, Romania (e-mail: lucian.vintan@ulbsibiu.ro).

V. Tresp is with the Siemens AG, Information and Communications, 81739 Munchen, Germany (e-mail: volker.tresp@siemens.com).

For extending the SVM algorithm from two-class classification to multi-class classification typically one of two methods is used: "One versus the rest", where each topic is separated from the remaining topics, and "One versus the one", where a separate classifier is trained for each class pair. We selected the first method for two reasons: First, preliminary experiments shows that the first method gives better performance, which might be explained by the fact that the Reuter's database contains strongly overlapping classes and assigns almost all samples in more than one class. Second overall training time is much shorter for the first method.

III. Selecting Individual Classifiers

Our previous work [8], [10] showed that the correct classification of some "difficult" documents strongly depended on selecting the optimal SVM architecture. Following this observation we first generate a pool of different SVM classifiers. These classifiers use different kernel types, different kernel degrees and different input data representation. The latter was showed in [8] that have a great influence on classification accuracy. After analyzing test results for each studied classifier [8, 9, and 10] using the same training and testing data set we selected 8 different classifiers as components of the developed meta-classification system.

Table I. Selected Clasifiers

Nr. Crt.	Kernel type	Kernel degree	Data representation	Accuracy obtained
1	Polynomial	1	Nomianl	86.69
2	Polynomial	2	Binary	86.64
3	Polynomial	2	Cornell Smart	87.11
4	Polynomial	3	Cornell Smart	86.51
5	Gaussian	C1.8	Cornell Smart	84.30
6	Gaussian	C2.1	Cornell Smart	83.83
7	Gaussian	C2.8	Cornell Smart	83.66
8	Gaussian	C3.0	Cornell Smart	83.41

As we showed in [8] and [10] the best results were obtained for this data set for an optimal dimension of the feature vector of 1309 features. Table I present those 8 selected classifiers, each of them with optimized parameter using a cross validation set.

In all presented results we used Support Vector Machine as features selection method (SVM_FS) as presented in [8]. We showed that this method obtained best results if compared with Information Gain [8] or Genetic Algorithm [9].

An interesting question is if there are some input documents, which are incorrectly classified by all selected classifiers. In all comparisons we take as a reference the Reuters's classification. To analyze this question, we take all selected classifiers for the Reuters' data and count documents that are incorrectly classified by all classifiers. We found 136 test set documents from 2351 that are incorrectly classified by all classifiers. Thus the maximum limit of our meta-classifier containing these selected classifiers is 94.21.

IV. Meta-Classifier Models

In order to design the meta-classifier we are using three models. First of them is a simple approach based on the voting principle. The other two approaches are implementing adaptive methods.

A. Majority Vote

This first model for meta-classification was tested due to its simplicity. It is a maladjusted model that obtains the same results in time. The idea is to use all selected classifiers to classify the current document. Each classifier proposes a specified class for this document incrementing the corresponding class-counter. The meta-classifier will select the class with the greatest count. If we obtain two or more classes with identical count we classify the current document in all proposed classes. The percentage of documents correct classified with this meta-classifier is 86.38%. This result is with 0.73% worse than the maxim value obtained with the best selected classifier, but is greater than their average accuracy.

B. Selection based on Euclidean Distance (SBED)

Since the previous meta-classifier didn't obtain good results we build a meta-classifier that adapts its behavior depending on the input data. To do this, we build a meta-classifier that selects a classifier based on the current input data. It will learn only on data that is incorrectly classified by the selected classifier, because we are expecting a smaller number of incorrectly classified input data if compared to correctly classified input data. Thus we create for each classifier a buffer which contains all incorrectly classified documents. Therefore, our meta-classifier consists in the 8 queues attached to the component classifiers.

When we have an input document (current sample) that needs to be classified, first we randomly chose one classifier. We compute the Euclidean distance (equation 2) between the current sample and all samples that are in that self queue of the selected classifier. If we obtain at slightly one pattern with a distance smaller than a predefined threshold we will reject this classifier. Instead we will randomly select another classifier, except for the already rejected one. If all component classifiers are rejected, however, we'll choose the classifier with the greatest Euclidian distance.

$$Eucl(\mathbf{x},\mathbf{x}') = \sqrt{\sum_{i=1}^{n}([x]_i - [x']_i)^2} \qquad (2)$$

Where $[x]_i$ represent the value from entry *i* of the vector **x**, and **x** and **x'** represent the input vectors.

After selecting the optimal classifier we'll used it to classify the current sample. If the selected classifier succeeds to correctly classify the current document, nothing is done. Otherwise we'll put the current document into the corresponding queue. To see if the document is correctly or incorrectly classified we compare our proposing class with the Reuters proposed class.

We have also implemented an alternative method that

compares the input data with all the samples stored in the queues. We are not presenting results using this method because it is slower, taking in average with 16 minutes more. The quality of the obtained results was similar with the presented SBED method.

The training is as follows: the meta-classifier analyzes the training set and each time when a document is incorrectly classified, the pattern is added to the selected classifier queue. In the second step, the validation step, we test the classification accuracy using the validation set. In the testing step the characteristics of the meta-classifier remains unchanged. Because after each training part the characteristics of meta-classifier might change, we repeated these two steps many times. After 14 steps we obtain good results and the classification accuracy have not substantially increasing after that.

C. Selection based on Cosine Angle (SBCOS)

The cosine angle is another possibility to compute the document similarity, often used to calculate text similarities. The formula to compute the cosine angle θ between two input vectors **x** and **x'** is:

$$\cos\theta = \frac{\langle \mathbf{x}, \mathbf{x}' \rangle}{|\mathbf{x}| \cdot |\mathbf{x}'|} = \frac{\sum_{i=1}^{n} [x]_i [x']_i}{\sqrt{\sum_{i=1}^{n} [x]_i^2} \cdot \sqrt{\sum_{i=1}^{n} [x']_i^2}}, \quad (3)$$

where $[x]_i$ represent the value from entry *i* of a vector **x**.

This method is like the method SBED with two modifications. The first modification is that the similarity between documents is computed using the cosine angle between input vectors. The second modification is that the classifier is not randomly selected. Instead we constantly take into consideration all available classifiers. We compute the cosine between the current sample and all incorrect classified samples that are in the self queues. We choose the classifier that obtains the minimum cosine value.

We have already implemented an alternative method that randomly chooses the first acceptable classifier, similar with the SBED presented method. We are not presenting this method because, despite its speed, the classification results are inferior (maximum 89.11% classification accuracy).

V. Experimental Framework

A. The Dataset

Our experiments are performed on the Reuters-2000 collection [11], which has 984Mb of newspapers articles in a compressed format. Collection includes a total of 806,791 documents, with news stories published by Reuters Press covering the period from 20.07.1996 through 19.07.1997. The articles have 9822391 paragraphs and contain 11522874 sentences and 310033 distinct root words. Documents are pre-classified according to 3 categories: by the *Region* (366 regions) the article refers to, by *Industry Codes* (870 industry codes) and by *Topics* proposed by Reuters (126 topics, 23 of them contain no articles). Due to the huge dimensionality of the database we will present here results obtained using a subset of data. From all documents we selected the documents for which the industry code value is equal to "System software". We obtained 7083 files that are represented using 19038 features and 68 topics. We represent a document as a vector of words, applying a stop-word filter (from a standard set of 510 stop-words) and extracting the word stem [12]. From these 68 topics we have eliminated those topics that are poorly or excessively represented. Thus we eliminated those topics that contain less than 1% documents from all 7083 documents in the entire set. We also eliminated topics that contain more than 99% samples from the entire set, as being excessively represented. After doing so we obtained 24 different topics and 7053 documents that were split randomly in training set (4702 samples) and testing set (2351 samples). In the feature extraction part we take into consideration both the article and the title of the article.

B. Kernel Types

The idea of the kernel trick is to compute the norm of the difference between two vectors in a higher dimensional feature space without representing those vectors in the new feature space. In practice we observed that by adding a constant bias to the kernel we obtained improved classifying results. In this work we present results using a new idea to correlate this bias with the dimension of the space in which the data will be represented. For more details please consult [7]. We consider that those two parameters (the degree and the bias) need to be correlated in order to improve the classification accuracy.

We'll use in our selected classifiers two types of kernels each of them with different parameters (see section III). For the polynomial kernel we vary the degree and for the Gaussian kernel we change the parameter *C* according to the following formulas (*x* and *x'* being the input vectors):

- Polynomial

$$k(x, x') = \left(2 \cdot d + \langle x \cdot x' \rangle\right)^d \quad (4)$$

  - *d* being the only parameter to be modified and represent the degree of the kernel,
- Gaussian (radial basis function RBF)

$$k(x, x') = \exp\left(-\|x - x'\|^2 / n \cdot C\right) \quad (5)$$

  - *C* being the classical parameter and *n* being the new parameter, introduced by us, representing the number of elements from the input vectors that are greater than 0.

For feature selection with Support Vector Machine method we use the polynomial kernel with degree 1 [8].

C. Correlating Parameters for the Kernel

Usually when learning with a polynomial kernel researchers use a kernel that can be expressed as like $\left(\langle \mathbf{x} \cdot \mathbf{x}' \rangle + b\right)^d$ where *d* and *b* are independent parameters. Parameter "*d*" is the kernel degree and it is used as a parameter that helps mapping

the input data into a higher dimensional space. Thus, this parameter is intuitive. The second parameter "*b"* (the bias), is not so easy to infer. In all previous work, the researchers used a nonzero *b*, but they didn't present a method for selection it. We notice that if this parameter was eliminated (i.e., chosen to be zero) the quality of the results can be poor. It is logically that there is a need to correlate the parameters *d* and *b* because the offset *b* needs to be modified as the dimension of the space modifies. Due to this, based on laborious classification simulations presented in [7], [10], we suggest the best correlation is "*b=2*d"*.

Also for the Gaussian kernel we modified the standard kernel used in the research community given by formula $k(x,x') = \exp(-\|x-x'\|^2 / C)$, where the parameter C is a number witch usually takes values between 1 and total numbers of features. We introduce the parameter *n* [7] that multiplies the usually parameter *C* with a value that represents the number of distinct features having weights greater than 0 that occur in the current two input vectors, decreasing substantially the value of *C* (see Equation 5). As far as we know, we are the first authors proposing a correlation between these two parameters for both polynomial and Gaussian kernels.

D. Representing the Input Data

Also in our selected classifier we will use different representation of the input data. After extensive experiments [8], [10] we see that different types of kernels work better with different types of data representation. We represent the input data in three different formats. In the following formulas *n(d, t)* is the number of times that term *t* occurs in document *d,* and $n(d,\tau)$ is the maximum frequency occurring in document *d*.

- **Binary representation** – in the input vector we store "0" if the word doesn't occur in the document and "1" if it occurs without considering the number of occurrences.
- **Nominal representation** – we compute the value of the weight using the formula:

$$TF(d,t) = \frac{n(d,t)}{\max_{\tau} n(d,\tau)} \tag{6}$$

- **Cornell SMART representation** –we compute the value of the weight using the formula:

$$TF(d,t) = \begin{cases} 0 & if\ n(d,t) = 0 \\ 1 + \log(1 + \log(n(d,t)) & otherwise \end{cases} \tag{7}$$

This are later called as BIN, NOM or CS.

VI. Experimental Results

In [8] and [10] we showed that the best results are obtained using a dimension of the feature space about 1309 relevant features. In this paper we present results obtained using only this feature dimension. For select relevant features we use feature selection method based on support vector machine technique, also with a linear kernel. This method was detailed in [8].

In all presented comparisons accuracies we take as a reference the Reuter's classification topics that was considered to be perfect. Also all results are presented for multi-class classification, taking into consideration all 24 selected classes.

In Fig. 1 we present results obtained using all three approaches. Also in this figure the upper limit that can be obtained with our selected classifiers is represented.

As we already mentioned, the last two methods, SBED and SBCOS, request some learning steps for training. We resume presenting here only first 14 steps because after those steps the substantial improvement of classification accuracy wasn't obtained. Sometime, after those steps we obtain small decreases followed by small increases, in average the values are closer to presented value. For example for SBCOS obtain 89.66% after 10 steps, decreasing after that to 89.58% and increasing to 89.74% in step 14th. In order to have a good view, we multiply in presented figure the value of upper limit (94.21%) that can be obtained.

With Majority Vote the accuracy of classification that was obtained with this meta-classifier is 86.38%. This result is with 0.73% smaller than the maxim individual value but it is greater than average over all classifiers.

For each of the last two methods (SBED and SBCOS) we are doing 14 learning steps. After each learning step we do a testing step. In Fig. 1 we present results obtained after each step as a percentage of correct classified documents.

For SBED the distance threshold was chosen during the firsts 7 steps equal to 2.5 and during the last 7 steps equal to 1.5. For the initial training phase we selected a greater threshold value in order to quickly obtain a "correct" classifier. Both these distance thresholds were chosen after laborious simulations.

At the beginning of training, when there are no documents in the queue yet, the classification accuracy is not so good (84.77%). But, as can be observed, after each step the accuracy improves growing up to 92.04% in 13th step. Comparatively with the upper limit that can be obtained with these selected classifiers, the obtained results after the 13th step can be considered a good result.

In SBCOS we chose during the first 7 steps a cosine threshold equal to 0.8 followed during the last 7 steps by a threshold equal to 0.9. If compared with SBED this method has a better starting point (85.33%). After 14 steps the accuracy increases only to 89.75%. Also the SBCOS method is slower than the SBED because it computes the distance between the current sample and all samples that are into the queues, in order to find the minimum value. In contrast, SBED randomly finds the first acceptable chose pattern and not the globally optimal one. The difference time between those two methods is in average of 21 minutes. If we compare those two methods using same modality to select classifier the method based on Euclidian distance is faster in average with 5 minutes than method based on cosine, whatever selected method we chose to be use.

In last two methods we kept in the queue of each classifier

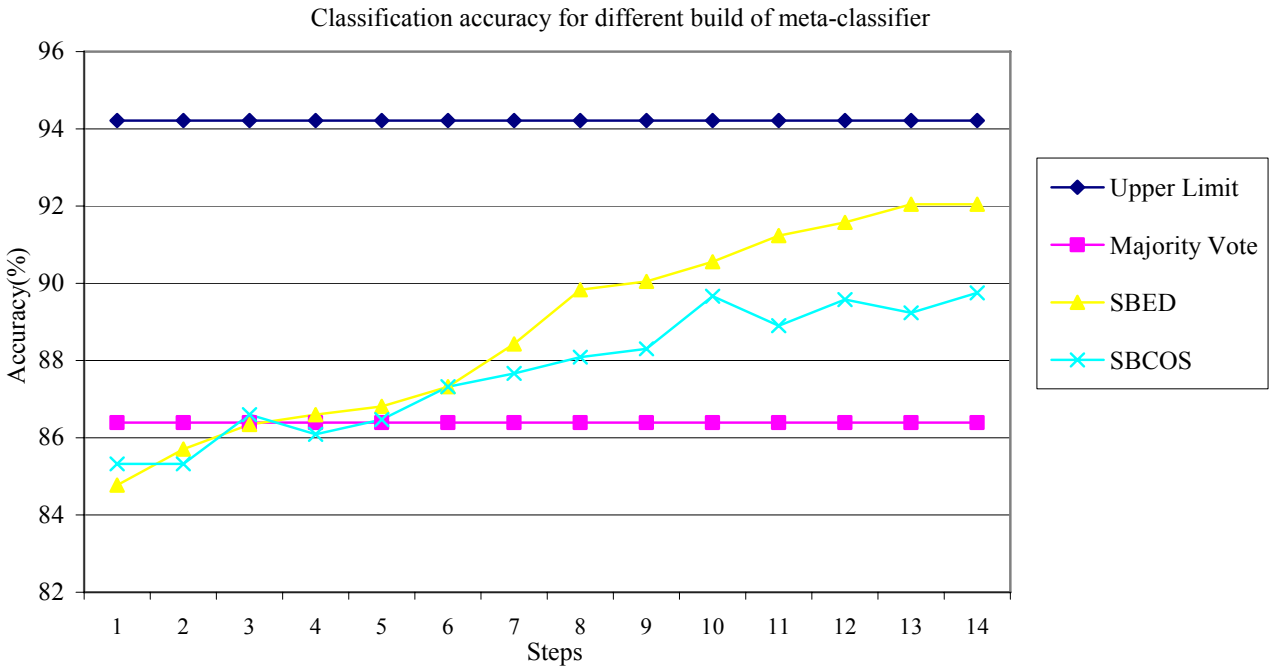

Fig. 1 Influence of approaches to build meta-classifier

the vector of documents that were incorrectly classified by that classifier. As an alternative to this, we also tried to reduce the queues' dimension by keeping only the average over all vectors that are needed to be kept. Thus each queue has now only a single vector, making the algorithm faster. Unfortunately the results are not so good, achieving only 87.11% accuracy, so that we will not discuss it further.

The Majority Vote needs more than one hour to generate the result. This relatively long training time occurs because it is necessary to compute the result involved by each of the 8 component classifier. The SBCOS is also not as fast: the response time increase from 19 minutes for the first step to 53 minutes for the 14th step when the dimension of the queue is greater. The fastest method is SBED where the time increases from 18 minutes for the first step to 24 minutes at 14th step, when the queue is completed. The numbers are given for a Pentium IV at 3.4 GHz, with 1 GB memory, 10 GB HDD a (7200 rpm) and Windows XP.

VII. Conclusions and Further Work

In this paper, we investigated three approaches to build an efficient meta-classifier. Based on our previous work we select 8 different SVM classifiers. For each of the classifier we modified the kernel, the degree of the kernel and the input data representation. Based on these selected classifiers we calculate the upper limit of our meta-classifier that is 94.21%. We compare one simple static method based on Majority Vote with two adaptive methods.

With Majority Vote the classification accuracy was 86.38%. As we expected, the documents that are correctly classified by only one classifier can't be correctly classified by this method.

The SBED method obtains best results, growing up to 92.04% after 14 learning steps with 2.17% smaller than the upper limit. Also this method is the fastest one because it selects the first acceptable classifier and because the computation cost is lowers. The last method (SBCOS) is the most rigorous one because it finds the best component classifier. As a consequence, the training time for SBCOS is longer at an average of 21 minutes comparatively with SBED.

The goal of our ongoing work is to classify larger text data sets (the complete Reuters database). Also we want to develop a pre-classification of all documents, obtaining fewer samples (using simple algorithms like Linear Vector Quantization or Self Organizing Maps). After that we'll use the obtained samples as entry vectors for the already developed features selection and classification methods.

An interesting natural extension of our work might be an adaptation for Web mining applications, in order to extract and categorized online news.

Acknowledgements

The first two authors would like to express thanks to SIEMENS AG, CT IC MUNCHEN, Germany, especially to Dr. h. c. Hartmut RAFFLER, for his generous support, that he has provided in developing this work.

References

[1] N. Dimitrova, L. Agnihotri and G. Wei, *Video Classification Based on HMM Using Text and Face*, Proceedings of the European Conference on Signal Processing, Finland, 2000

[2] G. Siyang, L. Quingrui, M. Lin, *Meta-classifier in Text Classification*, http://www. comp.nus.edu.sg/~zhouyong/papers/cs5228project.pdf

[3] W.-H. Lin , A. Houptmann, *News Video Classification Using SVM-based Multimodal Classifier and Combination Strategies*, 2003
[4] W.-H. Lin , R. Jin, A. Houptmann, *A Meta-classification of Multimedia Classifiers*, International Workshop on Knowledge Discovery in Multimedia and Complex Data, Taiwan, 2002
[5] B. Schoelkopf, A. Smola, "Learning with Kernels, Support Vector Machines", MIT Press, London, 2002.
[6] C. Nello, J. Swawe-Taylor, "An introduction to Support Vector Machines", Cambridge University Press, 2000.
[7] D. Morariu, L. Vintan, "A Better Correlation of the SVM kernel's Parameters", Proceeding of the 5th RoEduNet International Conference, Sibiu, June 2006.
[8] D. Morariu, L. Vintan, V. Tresp, *Feature Selection Methods for an Improved SVM Classifier*, Proceedings of the 14th International Conference on Computational and Information Science, pp. 83-89, Prague, August 2006
[9] D. Morariu, L. Vintan, V. Tresp, *Evolutionary Feature Selection for Text Documents Using the SVM* , Submitted to The 3rd International Conference on Neural Computing and Patter Recognition, October 2006
[10] D. Morariu, "Classification and Clustering using Support Vector Machine", 2nd PhD Report, University „Lucian Blaga" of Sibiu, September, 2005, http://webspace.ulbsibiu.ro/ daniel.morariu/html/Docs /Report2.pdf.
[11] Reuters Corpus: http://about.reuters.com/researchandstandards/corpus/. Released in November 2000.
[12] S. Chakrabarti, "Mining the Web- Discovering Knowledge from hypertext data", Morgan Kaufmann Press, 2003.