Hierarchical Classifiers with Shortcuts and their Application to Text Classification

Seung Hoan Jeon

Department of Computer Science
Brown University

Submitted in partial fulfillment of the requirements for the Degree of Master of Science in the Department of Computer Science at Brown University

May 2001

Professor Thomas Hofmann
Advisor

ABSTRACT

This paper explores hierarchical classification of web content by using a collection of probabilistic multiclassifiers implementing Error Correcting Output Encoding (ECOC) and Probabilistic Support Vector Machines.

Error correcting output coding is a technique where ensembles of binary classifiers are used for the multi-way classification task. Using probabilistic support vector machines as binary classifiers, which are known for their effectiveness in text classification, we applied a probabilistic ECOC technique in categorizing real world datasets for the multi-class learning problem.

In hierarchical classification, the initial classification is critical since subsequent level of multi classifiers cannot recover from an error that occurred at a higher level.

We tried to approach this problem by constructing another category structure called a shortcut hierarchy, which skips the intermediate level in the category structure.

By observing prediction disagreements between the direct and shortcut hierarchical classifiers, misclassifications could be detected which could aid in error recovery.

Classification performances of the two hierarchy structures were analyzed, and several recovery heuristics by combining the two multiclassifier hierarchies were explored.

TABLE OF CONTENTS

1. Introduction

As information on the Internet grows exponentially, organizing relevant materials became increasingly difficult. As a result, various algorithms for automatic text classification have been developed. In this paper, we use probabilistic multi-way classifiers, which incorporate Error Correcting Output Codes with Support Vector Machines to classify web pages in a hierarchical manner.

SVMs are well known for their performance and efficiency and have been used effectively for text classification with large input features [1,2].   In order take the uncertainty of a SVM prediction into account; we used a generalized version of SVM that computed the probability of a label.

Multi-class learning using ECOC has been explored for a variety of problems, including text classification [3, 4, 5]. By using ensembles of probabilistic SVMs and ECOC, a probability distribution for all possible terminal categories could be calculated.

Two different classifier hierarchies using probabilistic multi-classifiers were built on the identical dataset to detect misclassification at a higher hierarchy level. With a single hierarchy of classifiers, an error at a higher category node could propagate to the terminal node without being detected. However, if a second classifier group voted for a different class, an error was transparent.

The advantage of using a hierarchical structure over a flat approach depended on the specific task. For certain applications, the hierarchical approach did not provide any significant advantage over the flat approach [7], whereas minor improvements were shown with specific text classification problems [8].

We compared the hierarchical and flat approach for classifying web pages using the Open Directory Repository as a reference, and explored possible improvements by combining the two category structures.

2. Probabilistic Multiclassifier

The probabilistic multi classifier outputs a probability distribution over each possible label for a test sample $\vec{x}$. Since a standard multiclassifier only returns the most probable label, obtaining a probability distribution for all the possible labels for $\vec{x}$ gives more insight of the prediction. Although the final result for simply choosing X for $\vec{x}$ is identical with assigning the highest probability value for X among all possible labels, we can get a better understanding with the second approach. It is possible to identify if the label X was assigned with a high confidence or simply a marginal preference over the second label, which provides more detail. In designing the probabilistic multiclassifier, probabilistic versions of the Support Vector Machines and Error Correcting Output Codes have been used.

2.1 Probabilistic Support Vector Machines

The output of the support vector machine $svm < \vec{x}, \vec{w} >$ represents the distance from point $\vec{x}$ to the decision boundary constructed with $\vec{w}$. The support vector machine is more confident with its prediction if the $\vec{x}$ is further away from the decision surface in either direction. We assume that training samples belonging to the same class as $\vec{x}$ has been used to train the SVM in order to retrieve $\vec{w}$.
With this information, a probabilistic version of the SVM can be used to measure the prediction confidence. Using a logistic transformation with a free parameter β, and a primal formulation with $\|\vec{w}\| = 1$, the SVM confidence in the predicted label +1 would be:

$$\Pr\{\ y(\vec{x}) = +1; \vec{w}, \beta\} = \frac{1}{1 + e^{-\beta \cdot svm < \vec{w}, \vec{x} >}}$$

Where $\Pr\{y(\vec{x}) = -1; \vec{w}, \beta\} = 1 - \Pr\{y(\vec{x}) = +1; \vec{w}, \beta\}$, and $y(\vec{x}) \in \{-1,1\}$ is the predicted label of the probabilistic SVM.
Notice that β controls how classification uncertainty relates to the distance from the classification boundary.

The linear kernel was used for the SVMs. Though other kernels provide more flexibility in constructing the decision surface in a higher dimensional space, text classification has been known to be a linearly separable problem [2] where a linear kernel would be sufficient. This approach also reduced the computational overhead with marginal precision loss.

2.2 Probabilistic Error Correcting Output Codes

Instead of using the hamming distance to find the best matching category in the ECOC [2], an ensemble of probabilistic support vector machines can be used to calculate the probability for each possible class.

With K classes {c_1, c_2 ... c_k}, an ensemble of M binary classifiers with weight vector $\vec{w}_m$ can be trained according to some error correcting code matrix. For Each class c_k, we will have a vector of target outputs $\vec{y}(c_k) \in \{-1,+1\}^M$. We can now compute the multi class probabilities by combining the binary classifiers linearly:

$$\Pr(y(\vec{x}) = c_k; \vec{w}_1, \ldots, \vec{w}_M, \beta\} = \frac{\sum_{m=1}^{M} \Pr\{y_m(\vec{x}) = y_m(c_k); \vec{w}_m, \beta\}}{\sum_{t=1}^{K} \sum_{m=1}^{M} \Pr\{y_m(\vec{x}) = y_m(c_t); \vec{w}_m, \beta\}}$$

Here, $y_m(\vec{x})$ is the label of $\vec{x}$ computed for the mth classifier.

As the β for the probabilistic SVM increases, the result becomes more similar to the original ECOC approach [2] using binary classifiers with only 0, 1 as the output.

3. Category Hierarchies

The probabilistic multi-way classifiers at each inner node are trained to discriminate data points between the multiple child categories. Computing the actual classification probabilities and interpreting those as transition probabilities to move from an inner node to one of its child nodes, it implicitly defined a probability distribution over the terminal nodes of the hierarchy. The probabilities can be computed efficiently with a single top down pass through the inner nodes of the hierarchy. Two different classifier hierarchies were used for the experiments.

3.1 Direct Hierarchy

Each non-terminal category has a corresponding probabilistic multiclassifier returning a probability distribution over its children. We can calculate the probability distribution of N terminal nodes over a two-level hierarchy as follows:

Step 1 $\Pr(N_i) = \Pr(Root(\vec{x}) = C_j) + \alpha \cdot \Pr(C_j(\vec{x}) = N_i)$

C_j : Parent of a terminal node N_i .
α : Discount factor controlling the effect of the second-level probability distribution on the terminal probability.

Subsequently, the following normalization step has to be applied to the probability distribution at the terminal level.

Step 2 $\Pr(N_i) = \dfrac{\Pr(N_i)}{\sum_{\forall j} \Pr(N_j)}$

The discount factor α controls the effect of the probability values of the inner nodes in calculating the terminal probability distribution. The second-level prediction dealing with a more specific category domain should have less influence in determining the terminal probabilities.
In general, α should be a sufficiently small value that does not affect the prediction sequence of the root classifier when propagated down to the terminal node.
Let's assume that the root classifier predicted the label for $\vec{x}$ in the following order.

1. Chat	*0.40*
2. Email	*0.35*
3. Organizations	*0.25*

The three inner nodes "*Chat*", "*Email*" and "*Organizations*" will calculate their own probability distribution of $\vec{x}$ on their children. If the "*Chat*" node has four children, which are "*ICQ*", "*IRC*", "*Topic*s" and "*General*", the distribution might look like:

1. Chat.ICQ	*0.5*
2. Chat.IRC	*0.3*
3. Chat.Topics	*0.15*
4. Chat.General	*0.05*

If α is sufficiently small, it will guarantee that the terminal probability values to classes "*Chat.ICQ*", "*Chat.IRC*", "*Chat.Topics*", and "*Chat.General*" are higher than other terminal nodes whose parents are "*Email*" or "*Organizations*".
If α is too large, this condition would not hold which would invalidate the usage of a hierarchy structure. This greedy approach yielded better performance compared to other multiplicative approaches that will be explained later.

Empirical results have shown that the optimal value of α to be in the range between 0.05 and 0.1. This value range would generally avoid special cases where a terminal prediction in a second category, for example, "*Email.Client*" would have a higher value than one of the "*Chat*" terminal nodes in the previous scenario.

NOTE:

Probability Path Calculation

Initially, for the direct hierarchy structure, we applied the following formula to calculate the path probabilities before normalizing the terminal distribution.

$$\Pr(N_i) = \Pr(Root(\vec{x}) = C_j) \cdot \Pr(C_j(\vec{x}) = N_i)$$

C_j : parent of a terminal node N_i .

Path probability was computed multiplicatively instead of using linear addition. However, the accuracy with this terminal distribution was significantly lower than the greedy approach. It seemed that the probability distribution of the upper level nodes were too uniform which might have caused the problem. The following step was taken to spread out the probability in order to assign more extreme confidence values without changing the rankings.

$$\Pr(N_i) = \frac{\Pr^2(N_i)}{\sum_{\forall j} P^2 r(N_j)}$$

However, any kind of multiplicative approach on the direct hierarchical structure performed considerably worse than the greedy probability calculation.

A possible explanation to this problem could be that a probabilistic multiclassifier assumed that it has encountered the class of $\vec{x}$ at the training stage.
If a probabilistic multiclassifier encounters $\vec{x}$ not belonging in one of the N categories as it was trained initially, the ensembles of probabilistic SVMs could potentially output misleading confidence values. The SVM only calculates the distance of x from the decision surface and assumes that $\vec{x}$ belongs to either the positive or negative class. However, with $\vec{x}$ that belongs to neither the positive or negative class, the decision function might return a very long distance to the decision surface. This could be misinterpreted by the logistic function that $\vec{x}$ belongs to either one of the classes with high accuracy. Unfortunately, SVM only discriminates $\vec{x}$ as positive or negative example and does not detect if neither of the classes are correct.
For example, a probabilistic multiclassifier *A* has to classify $\vec{x}$ which belongs to *K*. *K* belongs to the left branch of the hierarchy. However, when calculating the terminal probability distribution, a multiclassifier on the right branch also has to classify $\vec{x}$. None of the classifiers of the right branch encountered examples of class *K* previously. Therefore, the multiclassifier could output a high confidence value that could be potentially higher than the prediction value of the *K* .

The only alternative to avoid this problem was to train every single multiclassifier with every terminal class. However, this was not a feasible solution, and therefore, we decided to use the greedy calculation for all the other experiments.

3.2 Shortcut Hierarchy

With the shortcut structure, we point straight from the root node to the grandchildren nodes skipping the intermediate level. At the training stage, the root node is directly trained with its grandchildren, so the probability distribution of $\vec{x}$ at the root is the terminal probability distribution.

For this hierarchy, there is only a single probabilistic multiclassifier at the root node that will discriminate all the terminal nodes. This approach contrasts the previous hierarchical structure employing an intermediate layer with several multiclassifiers.

3.3 Characteristics

Hierarchical classifications have shown performance improvements at certain problem domains [21] compared to flat classification. However, hierarchical classification performance was greatly dependant on the initial classification at the higher level.

If a test example was misclassified at an upper hierarchy level, the whole structure could not recover from the misclassification, and the error propagated to the terminal node.

The shortcut structure without an intermediate level, did not suffer from this problem. However, learning a discriminator function for a larger number of classes with less available training data raised different concerns.

4. The Open Directory Project - Top.Computers.Internet

The *Top.Computers.Internet* subcategory from the Open Directory Project [6] was used for all the experiments. These Open Directory web pages have been manually classified into a hierarchy of categories by volunteering web editors. The project is continuously evolving and more web pages are constantly added to the hierarchy structure.

The collection for the experiment was sampled on November 2000 containing 11,624 documents. Only the children and grandchildren node levels were used in the study. Category nodes below the grandchildren level in the tree structure were merged to the appropriate grandchildren nodes. After merging, category nodes with insufficient training data that had less than twenty examples were removed prior to processing.

If the multi-way classifier had to learn to discriminate ten to twenty classes, less than twenty training examples for a unique class seemed to be insufficient to perform the task properly.

Not all the web pages from the Open Directory Structure were at the terminal nodes. Editors intentionally assigned certain pages to inner nodes, when it was difficult to assign the certain web page into a subcategory. For all the web pages stored at the inner nodes, we created a separate "*General*" category and moved the pages to this child node. After removing the sparse categories and creating "*General*" nodes, we ended up with 13 first-level and 50 second-level categories.

For the experiment, the direct hierarchy used the thirteen first and fifty second-level categories, whereas the shortcut hierarchy only used the fifty second-level for their structures.

5. Preprocessing

For this experiment, we used the vector space representation for the web pages.
Several preprocessing steps were applied before transforming the web pages into the corresponding document vector.
Stop words were removed using the Britannica stop word list [9] containing 559 case sensitive stop words. The list compiled and published by Britannica.com, was considered to be a relatively small stop word list and did not have a significant impact in reducing the feature dimensionality for this particular application.
Next, the Porter Stemming algorithm [10] was applied to the dataset in order to strip the suffixes. This step reduced 30% of similar features and resulted in 150,000 unique case sensitive features. The high dimensionality did not have a significant impact on computation due to the fact that individual documents are represented as sparse vectors. The number of features with non-zero values in the document vector was a more critical factor in determining computational cost.
Finally, the lengths of the document vectors were normalized prior to training.

6. Training

Using the hierarchy as a natural decision tree, we trained an ensemble of probabilistic multi-way classifiers at each inner node to discriminate between the multiple child nodes. A bottom up approach was taken where the training propagated from the lower level inner nodes to the root node.

Each terminal node passed its own data samples with its category label to its parent. The parent nodes receiving the training data from all the child nodes trained their own multi-way classifier accordingly.

After training at hierarchy level *k* was completed, the *k* level nodes merged their training data with the different labels, and propagated the new training set to its parent with its own unique label.

For example, the inner node *Top.Computers.Internet.Chat* received training data from its children, which included the *Top.Computers.Internet.Chat.General*, *Top.Computers.Internet.Chat.ICQ* and *Top.Computers.Internet.Chat.IRC* nodes. Once the *Top.Computers.Internet.Chat* node learned a discriminator function, it merged all the training data from nodes starting with the *Top.Computers.Internet.Chat.** prefix and passed the merged training set to the *Top.Computers.Internet* parent node. The merged training set replaced all of its data label to the more generic *Top.Computers.Internet* .

Training a node at a higher level increased the training time, as the number of training samples increased with merging more data. However, with an increased number of training samples and training a more generic discriminator function, classification performance increased.

This merge and train approach was applied to both the direct and shortcut hierarchy using the identical training set.

7. Testing - Classification Performance

The two different structures were trained with the identical training and test set. Therefore, a direct comparison between the two hierarchical approaches could be made. The Open Source Directory data set was split randomly into the training and test set with a 9:1 ratio. The training and testing process for the experiments were repeated five consecutive times on average for a fair evaluation. Most of the time, the results were almost identical which confirmed the validity of the randomized approach. Classification accuracies and the disagreement level between the two structures were measured independently prior to applying a combination of the hierarchies.
This step was taken to examine the validity of using a second hierarchical structure used to recover from misclassifications.

8. Hierarchy Combination

We explored the case when the direct hierarchical classifier misclassified $\vec{x}$, whereas the shortcut classifier assigned the correct label or vice versa. If both trained hierarchical classifiers agree or disagree on a prediction regardless of its correctness, it would be impossible to detect a possible misclassification. If both predictions were in agreement all the time, creating a shortcut hierarchy would be unnecessary.
However, a combination of the two hierarchical classifiers would be provide a significant advantage if one of the prediction is incorrect, but the other classifier correctly labels $\vec{x}$. A heuristic implementing a disagreement policy has to be trained at the validation stage or set manually. However, the difficulty at a prediction disagreement would be to detect the correct label between the two predictions. Without prior knowledge of the test data or a policy, choosing the correct label would be difficult. The special case where both prediction labels are different but none of them correct is disregarded because the likelihood of recovery is extremely low.
We experimented a linear combination approach by adjusting the coefficient weights of the two terminal probability distribution of $\vec{x}$. By assigning different weights on the distributions, an optimal policy when a disagreement occurred could be found.

9. Implementation

Most of the project components were implemented with Java 1.3 that ran under Solaris. Initially, the SVM-Light 3.5 library [11] was integrated using the Java Native Interface. Unfortunately, multithreading and calling a large number of native calls on the SVM library stalled the Virtual Machine before finishing execution. Several bug reports with the identical problem were filed, but no alternatives have been suggested by Sun at the time of implementation. The learning component, which extensively called the SVM learning functions, was rewritten in C++.

After obtaining the Open Directory Project repository XML file containing the URLs of the web pages of each category structure, the download component fetched the most recently updated URLs from the Internet, and stored them as RDF files on a local disk. A major concern with Java sockets occurred when an open connection blocked and did not return to the client call. Not receiving a response from a socket could have been due to a slow connection or a temporarily disconnection, situations which should have been taken into account at the socket layer.

However, the sockets blocked at certain URLs and did not return after a significant amount of time, which stopped the whole downloading process. This component was rewritten using a multithreaded timer, which closed the connection if no response was received after a specific period of time.

The content of the RDF files were preprocessed accordingly to the specification. This phase included stop word removal, stemming and normalization prior to transforming the web pages into a document vector.

The hierarchical training component was implemented in C++ using the SVM-Light 3.5 library due to difficulty with the Java Virtual Machine. Though the direct hierarchical classifier had fourteen multiclassifiers, whereas the shortcut hierarchy only a single one, the numbers of SVM were identical which lead to a similar training time. The average training time for a hierarchy was 170 minutes using fifty Support Vector Machines. Training time was proportional with the number of SVM employed in the ECOC.

10. Results

10.1 Individual Hierarchy Precision

Evaluation was carried out using standard precision and accuracy. We computed global performance estimates using micro and macro-averaging. For micro-averaging, the results are computed based on global sums over all decisions, whereas macro-averaging computes the results on a per-category basis and averages these results. This was relevant to this application as it turned out that different categories had different number of data available.

First, we calculated the precision of the individual hierarchical classifiers. The precision values of five randomized experiments using a 9:1 ratio per category for the training and test set were averaged. The standard deviation of the precision values was marginal, which reaffirmed the validity of the randomized approach.

Standard Precision

	Micro Precision	Macro Precision
Direct Hierarchy	0.65	0.33
Shortcut Hierarchy	0.67	0.34

As we can see from the outcome, micro-averaged scores seemed to dominate macro evaluation. Micro-averaged scored tend to be dominated by the most commonly used categories, whereas macro averaging was dominated by the performance of rarely used categories. For this application, categories with a higher number of training and test data seemed to perform significantly above average, and therefore enhanced precision. By observing individual precision values, we could find that categories with very small data sets performed very poor as it was expected. In addition, the *General* categories grouping web pages, which could not be assigned to a proper class, had the worst performance. Almost all *General* categories had precision values of 0. This poor performance could be explained because it collected web pages that were hard to categorize in the first place. Web editors assigned them to internal nodes and deferred specialized groupings. Poor performance on this data set was as expected.

Though the precision of the multi-way classifier does not seem to match binary classifiers, which perform at the 90% level, the results are still acceptable considering that the classifier is trying to discriminate among more than twenty classes. Another measure was performed to analyze the precision of the first subcategory only. If $\vec{x}$ belonged to *Internet.Chat.ICQ*, all predicted labels starting with the *Internet.Chat* prefix would be viewed as correct. Though a prediction assigning *Internet.Chat.AIM* is false, the root classifier correctly classified the first subcategory *Chat*. This relaxed prediction measurement would still reveal some insight regarding the performance of the probabilistic multiclassifier.

First Level Precision

	Micro Precision	Macro Precision
Direct Hierarchy	0.83	0.44
Shortcut Hierarchy	0.84	0.45

10.2 Prediction Disagreements

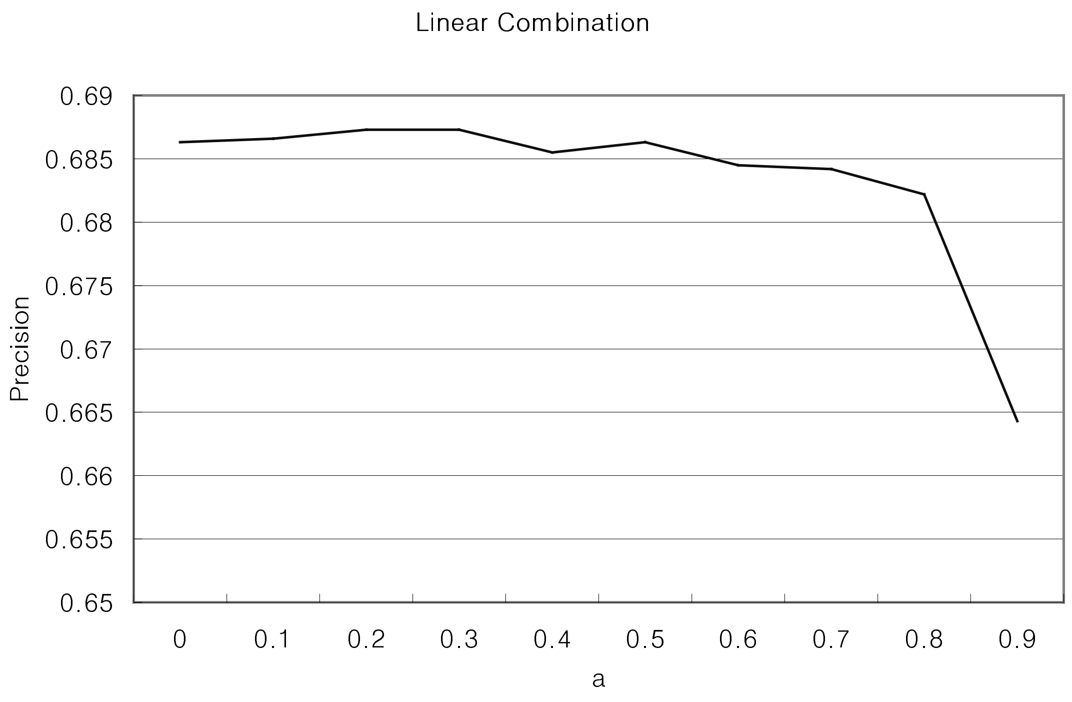
If both hierarchical classifiers were in complete agreement, the need for the second classifier would be redundant. We examined the disagreement between two classifiers prior to applying a linear combination to the terminal probabilities in order to integrate the two classifiers. Again, the data set was randomly split with a 9:1 ratio to create the training and testing set.

Prediction disagreement occurred with 15 percent of the test examples. These were the instances where at least one of the predictions was correct. The circumstance where both predictions disagreed and both were false was ignored in the measure due to difficulty in recovery. The precision of the shortcut classifier was at around 70 percent whereas the direct classifier classified 30 percent of the examples correctly.

If a heuristic choose the prediction of the shortcut classifier at disagreement, we would correctly classify 70 percent of the unclear examples, which is ten percent of the total test data.

10.3 Classifier Combination

The terminal probability distribution of $\vec{x}$ from the direct and shortcut classifiers were combined linearly before choosing the category with the highest probability value. Different weights for the distribution, and their prediction performance were shown in the graph below.



a : [0,1], weight fraction of the direct hierarchical classifier probability distribution.

The horizontal axis represent the fraction of the direct hierarchy classifier. If *a* was zero, the classification of the shortcut classifier was disregarded. If *a* was assigned one, only the shortcut probability distribution was taken into account. A performance improvement between one and two percent could be achieved by applying a weighed combination of the probability distribution of the two classifiers. Optimal performance could be achieved by combining the distributions with a 7:3 ratio for the direct and shortcut classifiers.

10.4 ECOC and Support Vector Machines

For the experiment, the ECOC matrix was generated randomly. However, each column was tested if its values were identical. The columns that had identical values were regenerated due to their ineffectiveness in contributing to the ECOC classification. For most part of the experiment, the code word bit length was set to twice the number of the categories that the multiclassifer was trying to discriminate. To observe the effect of the SVM numbers, training and testing was rerun using different code word bit lengths. The shorcut hierarchical classifier with fifty categories was used in the experiment.

Precision relation with ECOC code word bit length

Code Word Length	N	1.5N	2N	2.5N
Micro Precision	0.67	0.69	0.70	0.70

N – Number of learned categories

Precision improved as more SVMs were employed for training. However, it seemed to reach a plateau after 2N bits were used in the ECOC matrix.

11. Conclusion

In this paper, we have presented a combination of hierarchical classifiers to organize web content from the Open Directory Project. The direct and shortcut hierarchical classifiers disagreed with approximately fifteen percent of the test examples, where one of the predictions included the correct label. By employing a linear combination of the probability distribution of the classifiers, overall performance increased by two percent. Though the upper bound of an improvement is limited at fifteen percent, it would be difficult to apply a more successful heuristic without any more details about the test data. Using a different ECOC matrix generation algorithm with a different code word bit size could have boosted performance, however, any different approach would have shown limited improvements due to the similarity of the categories in the training set. Although the usage of a second hierarchical classifier doubled the training time, it performed better than a single hierarchical classifier. With the Open Directory Dataset, the shortcut hierarchical classifier should be preferred if only one classifier can be used in a constraint setting.

12. References

1. Susan Dumais, Hao Chen
   Hierarchical Classification of Web Content
   Proceedings of SIGIR -00, 23rd ACM International Conference
   on Research and Development in Information Retrieval (SIGIR'00).

2. Thorsten Joachims
   Text Categorization with Support Vector Machines
   Learning with Many Relevant Features.
   Proceedings of the European Conference on Machine Learning, Springer, 1998.

3. Thomas G. Dietterich, Ghulum Bakiri
   Solving Multiclass Learning Problems via Error-Correcting Output Codes
   Journal of Artificial Intelligence Research 2(1995) 263-286.

4. Rayid Ghani
   Using Error-Correcting Codes For Text Classification
   Proc. 17th International Conf. on Machine Learning (2000)

5. Adam Berger
   Error-correcting output coding for text classification
   In IJCAI'99: Workshop on machine learning for information filtering, 1999.

6. http://www.dmoz.org

7. L. Larkey
   Some issues in the automatic classification of US patents.
   In Working Notes for the AAAI-98 Workshop on Learning for
   Text Classification (1998).

8. Ruiz, M.E. and Srinivasan, P.
   Hierarchical neural networks for text categorization
   Proceedings of the 22nd International ACM SIGIR
   Conference on Research and Development in Information Retrieval (SIGIR'99)

9. http://www.britannica.com

10. Porter, M.F.
    An algorithm for suffix stripping, *Program* (1980)

11. Thorsten Joachims
    SVM-Light Support Vector Machine 3.5 (1998)

Appendix

Open Directory Top.Computers.Internet Hierarchy (November 2000)

13 First-Levels
50 Second-Levels

First-Level

Top.Computers.Internet.Abuse
Top.Computers.Internet.Chat
Top.Computers.Internet.Commercial_Services
Top.Computers.Internet.Cyberspace
Top.Computers.Internet.Domain_Names
Top.Computers.Internet.E-mail
Top.Computers.Internet.Etiquette
Top.Computers.Internet.Mailing_Lists
Top.Computers.Internet.Organizations
Top.Computers.Internet.Protocols
Top.Computers.Internet.Publications
Top.Computers.Internet.Telephony
Top.Computers.Internet.WWW

Second-Level

Top.Computers.Internet.Abuse.Spam

Top.Computers.Internet.Chat.General
Top.Computers.Internet.Chat.ICQ
Top.Computers.Internet.Chat.IRC
Top.Computers.Internet.Chat.The_Palace
Top.Computers.Internet.Chat.Topics
Top.Computers.Internet.Chat.Transcripts
Top.Computers.Internet.Chat.Virtual_Places

Top.Computers.Internet.Commercial_Services.Access_Providers
Top.Computers.Internet.Commercial_Services.Consulting
Top.Computers.Internet.Commercial_Services.Cybercafes
Top.Computers.Internet.Commercial_Services.Domain_Names
Top.Computers.Internet.Commercial_Services.Forwarding
Top.Computers.Internet.Commercial_Services.Internet_Marketing
Top.Computers.Internet.Commercial_Services.Training
Top.Computers.Internet.Commercial_Services.Web_Design_and_Development
Top.Computers.Internet.Commercial_Services.Web_Hosting
Top.Computers.Internet.Commercial_Services.Web_Presence_Providers
Top.Computers.Internet.Commercial_Services.Web_Tools

Top.Computers.Internet.Cyberspace.Culture
Top.Computers.Internet.Cyberspace.Online_Communities

Top.Computers.Internet.Domain_Names.Disputed_Domain_Names
Top.Computers.Internet.Domain_Names.General
Top.Computers.Internet.Domain_Names.Name_Search

Top.Computers.Internet.E-mail.Electronic_Postcards
Top.Computers.Internet.E-mail.Free
Top.Computers.Internet.E-mail.Web-Based

Top.Computers.Internet.Etiquette.General

Top.Computers.Internet.Mailing_Lists.Directories
Top.Computers.Internet.Mailing_Lists.Hosting_Companies

Top.Computers.Internet.Organizations.General

Top.Computers.Internet.Protocols.DNS
Top.Computers.Internet.Protocols.FTP
Top.Computers.Internet.Protocols.SNMP
Top.Computers.Internet.Protocols.Transmission_Protocols

Top.Computers.Internet.Publications.E-zines
Top.Computers.Internet.Publications.Mailing_Lists

Top.Computers.Internet.Telephony.General

Top.Computers.Internet.WWW.Authoring
Top.Computers.Internet.WWW.Best_of_the_Web
Top.Computers.Internet.WWW.Free_Stuff
Top.Computers.Internet.WWW.Message_Boards
Top.Computers.Internet.WWW.Searching_the_Web
Top.Computers.Internet.WWW.Web_Applications
Top.Computers.Internet.WWW.Web_Counters
Top.Computers.Internet.WWW.Web_Logs
Top.Computers.Internet.WWW.Web_Portals
Top.Computers.Internet.WWW.Web_Usability
Top.Computers.Internet.WWW.Webcams
Top.Computers.Internet.WWW.Website_Promotion