The Effect of Dimensionality Reduction on Large Scale Hierarchical Classification

Aris Kosmpoulos[1,2], Georgios Paliouras[1], and Ion Androutsopoulos[2]

[1] Institute of Informatics and Telecommunications,
National Center for Scientific Research "Demokritos", Athens, Greece
`https://www.iit.demokritos.gr/skel`
[2] Department of Informatics,
Athens University of Economics and Business, Greece
`http://nlp.cs.aueb.gr`

Abstract. Many classification problems are related to a hierarchy of classes, that can be exploited in order to perform hierarchical classification of test objects. The most basic way of hierarchical classification is that of cascade classification, which greedily traverses the hierarchy from root to the predicted leaf. In order to perform cascade classification, a classifier must be trained for each node of the hierarchy. In large scale problems, the number of features can be prohibitively large for the classifiers in the upper levels of the hierarchy. It is therefore desirable to reduce the dimensionality of the feature space at these levels. In this paper we examine the computational feasibility of the most common dimensionality reduction method (Principal Component Analysis) for this problem, as well as the computational benefits that it provides for cascade classification and its effect on classification accuracy. Our experiments on two benchmark datasets with a large hierarchy show that it is possible to perform a certain version of PCA efficiently in such large hierarchies, with a slight decrease in the accuracy of the classifiers. Furthermore, we show that PCA can be used selectively at the top levels of the hierarchy in order to decrease the loss in accuracy. Finally, the reduced feature space, provided by the PCA, facilitates the use of more costly and possibly more accurate classifiers, such as non-linear SVMs.

Keywords: Hierarchical Classification, Dimensionality Reduction, Principal Component Analysis

1 Introduction

In most classification problems the predefined categories are assumed to be independent. In hierarchical classification problems, a hierarchy is also given, which contains the relations between the categories. In the simplest case which we study in this paper, these relations are of is-a type and the hierarchy is a tree. We also assume that each instance belongs to only one category (single-label classification) and that this category is always a leaf.

Many researchers ignore the hierarchy and treat hierarchical classification of this type using flat classifiers, while others use mildly hierarchical approaches [1]. In most flat approaches, a binary classifier is trained for each category, using all the instances belonging to that category as positive examples and all or some of the other instances as negative examples (one-versus-all). The simplest form of hierarchical classification is that of cascade classification, where a binary classifier is trained for each node of the hierarchy, in order to separate it from its siblings. Then each test instance is guided through the hierarchy from root to leaf, choosing each time the most probable descendant.

In large scale hierarchical classification problems, the number of training instances and features can be very high (thousands or even millions). A flat classification approach can deal with the high dimensionality by performing instance and/or feature selection for each one-versus-all classifier. For hierarchical classification, however, feature selection is more complicated. Classifiers at the upper levels of the hierarchy need to deal with instances of all of their numerous descendant classes and any kind of intense feature selection could lead to a situation where many test instances cannot be represented adequately by the selected features. For example, in text classification with binary bag-of-word features (each indicating if a particular word is present in a text or not), there may be many test texts (belonging to very different descendant categories) that do not contain any of the words corresponding to the selected features of the upper level classifiers. These texts will have identical (all-zero) feature vectors and, hence, the upper level classifiers will be unable to distinguish them. Since intense feature selection is impossible in the upper levels of the hierarchy, classifier training can be very computationally expensive, because both the number of training instances and the number of features is high.

In order to facilitate hierarchical classification, we examine the use of a principal component (PCA) transformation, reducing the dimensionality at the top levels of the hierarchy. In this way, hierarchical classifiers can be trained on much fewer dimensions, leading to faster training and testing and to lower memory demands. At the same time, the use of a linear transformation of the initial feature set, instead of a discrete feature selection, reduces the risk that test instances will not be represented in the new space.

However, the use of PCA is not without difficulties. First, one needs to use a version of PCA that can handle the scale of the data. In this paper we select one such method and show that it can be used for large scale hierarchical classification. Additionally, we study the effect of dimensionality reduction on the computational performance of the classifier and its classification accuracy. In particular, we show that the combination of classifiers trained on a reduced feature space at the top levels of the hierarchy with classifiers trained on the original space at the lower levels provides the highest benefit for the lowest cost. We experiment with two popular hierarchical text classification datasets in this paper, but our approach should be useful in any hierarchical classification problem with many dimensions and sparse feature vectors.

In Section 2, we present the proposed approach and give advice on the selection of the most appropriate PCA method. Section 3 shows experimentally the effect of our approach on the computational cost and the accuracy of the hierarchical classifiers. Finally Section 4 concludes and points to future work.

2 Cascade Classification with PCA

In cascade classification a classifier must be trained for each node of the hierarchy. In this paper we focus on tree hierarchies, where instances belong only to the leaves of the hierarchy. An example of such a hierarchy is presented in Figure 1. In this example, a text classifier must be trained for each of the following nodes: *Arts*, *Health*, *Music*, *Dance*, *Fitness* and *Medicine*. A classifier of a node U is trained with all instances belonging to the leaf descendants of U as positive examples and all instances belonging to the leaf descendants of the siblings of U as negative ones. The classifier of node *Arts*, for example, would use instances of *Music* and *Dance* as positive examples and instances of *Fitness* and *Medicine* as negative ones.

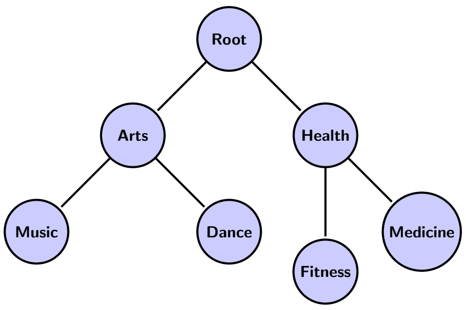

Fig. 1. Tree hierarchy example.

Assuming again bag-of-word features, if we do not perform any feature selection the classifier of node Arts would use features for all the words of its positive and negative instances. In large datasets, where the hierarchy is composed of thousands of categories, the number of initial features can be hundreds of thousands (or even millions in text classification, if stemming or other similar preprocessing techniques are not used). Training a classifier for each node of the hierarchy using all these features can be very computationally demanding. On the other hand, an intense feature selection at the upper levels of the hierarchy would lead to inaccurate classifiers, since the few selected features would be unlikely to represent the test instances adequately, as already discussed.

Instead of feature selection, we suggest dimensionality reduction with principal component analysis (PCA) applied to each set of siblings of the hierarchy. In Figure 1, for example, we would need to perform PCA three times:

– for the nodes *Arts* and *Health*
– for the nodes *Music* and *Dance*
– for the nodes *Fitness* and *Medicine*.

The two classifiers of nodes *Arts* and *Health* would use the same feature space, and similarly for the siblings *Music* and *Dance* and *Fitness* and *Medicine.* Hence, PCA needs to be performed only once for each set of siblings. We note that performing PCA on all the leaves (as if we had a flat classification problem) would require a very large number of principal components to distinguish the leaves, drastically reducing the benefit of applying PCA.

Even applying PCA to sets of siblings, however, is not trivial at the scale that we are considering. In regular PCA an eigen decomposition of the covariance matrix YY^T $(p \times p)$ must be performed, where Y is the $p \times n$ matrix of the observed data. Most of the times a Singular Value Decomposition (SVD) of Y is performed instead:

$$Y = U\Sigma V^T \tag{1}$$

where U is the square $(n \times n)$ matrix whose columns contain the left singular vectors of Y, Σ is a $n \times p$ rectangular diagonal matrix containing the singular values and V is the square $(p \times p)$ matrix whose columns contain the right singular vectors of Y.

The number of features (p) and instances (n) can by too large to perform a regular PCA. In [2] an Expectation Maximization (EM) algorithm for PCA is proposed, where the number k of principal components must be set from the beginning. The steps of the EM are the following:

$$\begin{aligned} &\text{E-step: } X = (C^TC)^{-1}C^TY \\ &\text{M-step: } C^{new} = YX^T(XX^T)^{-1} \end{aligned} \tag{2}$$

where X is a $k \times n$ matrix and C is $p \times k$ matrix. These quantities are much easier to compute than those of the regular PCA, since k can be set to a much smaller value than p. In order to compute the final eigenvectors and eigenvalues, we only need to project the observed data Y to the orthonormal basis for the range of matrix C $(orth(C)^TY)$ and perform a regular PCA in this k-dimensional subspace.

Adopting this approach we can perform PCA in very large datasets, where normal PCA would be very computationally demanding, especially in terms of memory. Even with this approach, PCA remains computationally expensive, but in practice it only needs to be performed once per dataset, greatly reducing the time needed to perform subsequent experiments with many different classifiers.

The only disadvantage is that we need to choose the value of k (number of principal components) prior to performing PCA. In Section 3 we present results which show that by using only a few hundreds of principal components one can achieve similar results as when thousands of initial features are used.

Various linear and nonlinear dimensionality reduction approaches exist [3]. In this paper we focus on linear approaches, because of the large scale factor. Another linear approach that we could use is that of Simple PCA [4], but we chose EM since the principal components in Simple PCA are calculated approximately.

The most similar one is the extension of the Stochastic Gradient Ascent (SGA) neural network, proposed in [5]. The disadvantage of this method compared to EM PCA is that it requires a rate parameter to be set and also converges less quickly. Another approach would be to use the implicitly restarted Arnoldi method to compute the SVD of the data matrix [6]. However the EM approach seems more straightforward. We also examined the idea of using Sparse Principal Component Analysis [7], which was not suitable, as our focus was more on solving computational issues, instead of computing more accurate eigenvectors. In [8], the Fisher vector could not be directly used in large scale (but not hierarchical) image classification; hence, three different compression techniques were proposed to reduce the dimensionality.

3 Experimental Results

3.1 Experimental Set-up

In order to assess the effect of combining PCA with cascade classification, we used both the dry-run and the large datasets form Task 1 of the first Large Scale Hierarchical Text Classification Challenge (LSHTC1).[3]

The dry-run dataset contains 6,323 instances (split into train and validation files), composed of 55,765 distinct features and belonging to 1,139 categories. An extra set of 1,858 test instances is also provided for evaluation. The large dataset contains 93,505 instances (split into train and validation files), composed of 381,581 distinct features and belonging to 12,294 categories. The test instances in this dataset are 34,880.

In both datasets, every instance has to be classified in a single leaf of the hierarchy, and the hierarchy is a tree. The systems are evaluated using the evaluation measures of the challenge, which are: Accuracy, Macro F-measure, Macro Precision, Macro Recall and Tree Induced Error [9].

As statistical significance tests, we used p-test ($p < 0.01$) for accuracy and S-test ($p < 0.01$) for macro F-measure. More information regarding these tests can be found in [10].

In the experiments we report, we used an L2 Regularized Logistic Regression [11], with the regularization parameter C set to 1 (usually the default value). We also conducted experiments with other regularization methods and other values of C, but the results were similar. We experimented with TF and TF-IDF bag-of-word features, but we report mostly experimental results with TF-IDF features, since led to better performance.

For each node of the hierarchy we trained two binary classifiers. One using all the initial features and one using k principal components. R-analysis requires that $k < n$ [12], but for computational reasons we set a much smaller value for k. In practice, we observed that in both datasets for values of k greater than a certain point the computational cost increased a lot, without significant gains in terms of classification accuracy. For the dry-run dataset, we set k equal to 390,

[3] http://lshtc.iit.demokritos.gr/node/1

i.e. 390 components. In cases where $390 > n$ we set k equal to $n - 1$ in order to satisfy $k < n$. In Figure 2 we present accuracy at the first top level of the hierarchy (children classes of the Root node) of the dry-run dataset, for various numbers of principal components. The figure illustrates the decreasing gains in accuracy as the number of components increases. Similarly for the large dataset we set k equal to 490.

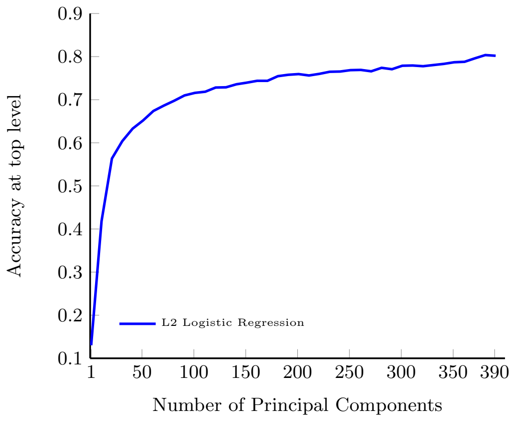

Fig. 2. Accuracy at top level of the hierarchy of the dry-run dataset, using k Principal Components for various values of k.

3.2 Feature Selection Results

In this section we present results showing that feature selection at the top levels of the hierarchy can heavily decrease the accuracy of the classifiers. In Tables 1 and 2 we present results in terms of accuracy and training time at the first level of the hierarchy of the dry-run and the large dataset, using feature selection. For each category the best features were selected according to the Chi-square statistic χ^2 [13]. These experiments were conducted using an i7 3.2 GHz CPU (single thread).

Although the best features are selected for each binary classifier of the top level, many instances cannot be represented adequately by the selected features (empty feature vectors). As a result, in both datasets the accuracy falls significantly with the reduced feature sets. Since these errors at the top level will be carried to the leaves of the hierarchy, any form of intense feature selection will lead to inferior final results compared to keeping all the features. On the other hand, the training times seem to be almost proportional to the number of features. Therefore, we gain in terms of training times, as well as memory requirments, since the size of the training models is correlated to the number of features.

Number of Features	Accuracy	Training Time (sec)
55,765 (100%)	0.82	7
27,882 (50%)	0.76	5
5,576 (10%)	0.56	0.84
557 (1%)	0.29	0.18

Table 1. Accuracy and training time at the top level of the hierarchy of the dry-run dataset using χ^2 feature selection.

Number of Features	Accuracy	Training Time (sec)
381,580 (100%)	0.83	328
190,790 (50%)	0.78	182
38,158 (10%)	0.63	57
3,815 (1%)	0.33	7

Table 2. Accuracy and training time at the top level of the hierarchy of the large dataset using χ^2 feature selection.

3.3 Results on the Dry-run Dataset

Since feature selection is ineffective at the top levels of the hierarchy, we reduce the number of features using PCA. In Table 3, we present the results of four different systems on the dry-run dataset, using the five evaluation measures of Section 3.1. The first system (*Cascade*) uses all features (TF-IDF) in order to train each binary classifier. The second system (*PCA Cascade*) uses only classifiers trained with PCA features in all levels of the hierarchy (with PCA applied to sibling classes). In the system *Combo Cascade*, the classifiers at the top two levels of the hierarchy are trained using PCA features, while the ones at the lower levels are trained using the initial features. Finally we also provide results of flat classifiers (*Flat*) trained using all the initial features.

Evaluation Measure	Cascade	PCA Cascade	Combo Cascade	Flat
Accuracy	**0.444***	0.419*	0.436*	0.438*
Macro F-measure	**0.312***	0.276	0.302*	0.304*
Macro Precision	**0.284**	0.250	0.275	**0.284**
Macro Recall	**0.346**	0.307	0.336	0.326
Tree Induced Error	**3.588**	3.754	3.673	3.976

Table 3. Results using the dry-run dataset for each approach per evaluation measure, using TF-IDF features. The best performing approach per evaluation measure appears in bold. Since Tree Induced Error is an error rate, lower values are better. Results with no statistically significant difference are marked with a $\star$ symbol.

The first observation is that the *Cascade* system performs better than all the other systems, including the popular flat classifier. Flat classifiers perform well enough according to the four flat evaluation measures, but they have the worst performance according to the Tree Induced Error (lower number indicates better performance), which is the only hierarchical evaluation measure. This means that when the flat classifier fails to predict the exact category, its mistake is further from the correct category compared to the hierarchical systems. This is particularly important in hierarchical classification problems. Another disadvantage of flat classification is that on large scale problems, the use of traditional classifiers, such as SVMs or Logistic Regression, can be prohibitively expensive computationally [14], since a binary classifier must be trained for each leaf using all instances.

Comparing *PCA Cascade* and *Cascade* we see that the latter is more accurate according to all evaluation measures. However, the difference is relatively small and in *PCA Cascade* the classifier is trained with 305 features instead of a few tens of thousands. Furthermore, we can improve the performance of *PCA Cascade* by using PCA only at the top levels of the hierarchy, where training is most expensive. As can be seen in Table 3, *Combo Cascade* achieves classification performance that is less than one precedence point smaller than that of *Cascade*.

Therefore, the combination of PCA at the top levels with training on the original feature space at the lower levels, provides high classification accuracy, while making cascading scalable to large datasets. The choice of the level at which the method should stop reducing the dimensionality of the feature space is largely an issue of computational cost. However, it is important to assess the effect of dimensionality reduction at each level. The results of this experiment are presented in Table 4. At each level of the hierarchy, we assume that the preceding (higher-level) classifiers have predicted the correct category and we measure only the accuracy at the corresponding level. According to the results in Table 4, it seems safe to assume that PCA affects classification accuracy similarly in all the levels of the hierarchy.

Level of the Hierarchy	Original Features	Reduced Dimensions (PCA)
1	$0.820^\star$	$0.814^\star$
2	$0.819^\star$	$0.812^\star$
3	$0.820^\star$	$0.807^\star$
4	$0.856^\star$	$0.849^\star$
5	$0.840^\star$	$0.834^\star$

Table 4. Accuracy for cascade classification on the dry-run dataset, using the original and the reduced dimensions (PCA) per level of the hierarchy. Results with no statistically significant difference are marked with a $\star$ symbol.

In Section 3.1 we mentioned that TF-IDF features provided better results than TF features. In Table 5 we present the results for *Cascade*, *PCA Cascade* and *Flat* using TF features. Not only *Cascade* and *Flat* systems performed worse with TF features, but also the *PCA Cascade* was greatly affected. Therefore we advise those who may use the proposed hierarchical PCA approach to perform a TF/IDF transformation.

Evaluation Measure	Cascade	PCA Cascade	Flat
Accuracy	**0.388***	0.361*	0.385*
Macro F-measure	**0.254***	0.221	0.248*
Macro Precision	0.232	0.201	**0.235**
Macro Recall	**0.281**	0.246	0.262
Tree Induced Error	**4.065**	4.356	4.625

Table 5. Results for each approach per evaluation measure, using TF features. The best performing approach per evaluation measure appears in bold. Since Tree Induced Error is an error rate, lower values are better. Results with no statistically significant difference are marked with a $\star$ symbol.

3.4 Results on the Large Dataset

In order to examine the scalability of our approaches, in Table 6 we present results for the large dataset using TF-IDF features. As in the dry-run data set *Cascade* uses all features in order to train each binary classifier. In *Combo Cascade* the classifiers at the top level of the hierarchy are trained using PCA features, while the ones at the lower levels using the initial features. Since the initial features are much more than the dry-run dataset (381,581 compared to 55,765), we used 100 more principal components (490) in the *Combo Cascade* system. We also provide results of flat classifiers (*Flat*) trained using all the initial features. Finally, we present the training times of the classifiers in each case.

Evaluation Measure	Cascade	Combo Cascade	Flat
Accuracy	0.404*	0.385	**0.405***
Macro F-measure	**0.278***	0.259	0.256*
Macro Precision	**0.269**	0.249	0.254
Macro Recall	0.289	0.268	**0.302**
Tree Induced Error	**3.609**	3.845	3.874
Training Time	8.66 min	6.2 min	1017.63 min

Table 6. Results using the large dataset for each approach per evaluation measure and training time, using TF-IDF features. The best performing approach per evaluation measure appears in bold. Since Tree Induced Error is an error rate, lower values are better. Results with no statistically significant difference are marked with a $\star$ symbol.

According to Accuracy and Macro Recall, *Flat* is somewhat more accurate than *Cascade*, although the p-test detected no statistically significant difference between them in Macro Recall. On the other hand, as in the dry-run dataset, according to the hierarchical evaluation measure (tree induced error) *Cascade* performs better. *Cascade* also performs better in terms of Macro Precision and Macro F-measure, although the S-test for Macro F-measure detected no significant difference. Finally *Combo Cascade*, with PCA applied to the top level of the hierarchy, performs slightly worse. In terms of training times, *Flat* is very slow compared to *Cascade*. Between *Cascade* and *Combo Cascade*, we observe a speed up of about 30%. Performing PCA at lower levels would only mildly affect speed, since the initial feature vectors are already sparse enough.

Furthermore, computational cost could also affect the choice of the classifier used. For example, an RBF SVM [15] is very expensive at the top level of the hierarchy, if the original feature set is used. However, with PCA the use of such a costly classifier is made possible, as the number of features is reduced from 381,581 to a few hundreds. In Table 7 we present the time in minutes required to train L2 logistic regression and SVM with an RBF kernel using the original and the reduced (PCA) features at the top level of the hierarchy of the large dataset.

	Original Features	Reduced Dimensions (PCA)	Gain
L2 Logistic Regression	5.46 min	2.84 min	48%
SVM with RBF kernel	691.2 min	124.1 min	82%

Table 7. Training time for L2 logistic regression and RBF SVM using the original and the reduced dimensions (PCA) at the top level of the hierarchy of the large dataset.

Although in both cases the training times are reduced, the gain is much larger for the RBF SVM. In addition to the training time, complex classifiers require more parameter tuning, which is only made possible with the *Combo Cascade* method. In Table 8 we present results, using an RBF SVM classifier at the top level of the hierarchy for the *Combo Cascade* method. In one case the SVM is trained using the default parameters, while in the other the parameters have been (non-exhaustively) tuned (c=100, g=0.01). As we can observe the tuned SVM performs much better than the default one. The p-test and S-test detected no statistically significant difference between *Cascade* and *Combo Cascade* with a tuned SVM. Given the training time cost in the initial feature space, this tuning would be much harder without the use of PCA.

Evaluation Measure	Cascade	Combo Cascade with Tuned SVM	Combo Cascade with Default SVM
Accuracy	**0.404***	0.402*	0.377
Macro F-measure	**0.278***	0.274*	0.252
Macro Precision	**0.269**	0.264	0.243
Macro Recall	**0.289**	0.283	0.262
Tree Induced Error	**3.609**	3.616	3.952

Table 8. Results using the large dataset for Cascade Combo with an SVM classifier at the top level, using TF/IDF features. Cascade results are repeated for ease of reference. The best performing approach, given each evaluation measure, appears in bold. Since Tree Induced Error is an error rate, lower values are better. Results with no statistically significant difference are marked with a $\star$ symbol.

4 Conclusion

In large scale hierarchical classification problems the number of features and instances to be used for training classifiers can be very large at the upper levels of the hierarchy. This can discourage the use of more complex, but more accurate classifiers and cause computational issues. In this paper we examined the use of dimensionality reduction (PCA) for hierarchical classification. This approach is independent of the classifier used and can be applied to all or some nodes of the hierarchy.

Even though performing PCA itself on such large scale datasets is not trivial, there are methods that can handle the complexity. We also showed experimentally that, although applying PCA to all levels of the hierarchy can decrease accuracy to some extent, this effect can be drastically limited, if we apply PCA only to the upper levels, which are the most computationally demanding. It would also be interesting to adaptively select the nodes were PCA should be used, instead of just applying it to the upper levels. We plan to examine such an adaptive selection method in the future.

The dimensionality reduction of the PCA procedure allows the use of more complex and possibly more accurate classifiers, such as non-linear SVMs. As future work, we also plan to compare the presented results with that of better tuned SVM classifiers. These classifiers are easier to train, using the reduced feature space provided by the PCA approach. We also plan to extend the presented PCA approach to Directed Acyclic Graphs (DAG) hierarchies, where it is unclear how the multiple inheritance of each node will affect the PCA for each set of siblings.

References

1. Kosmopoulos A., Gaussier É., Paliouras G., Aseervatham S.: The ECIR 2010 large scale hierarchical classification workshop. In: SIGIR Forum. vol. 44, pp. 23–32. (2010)
2. Roweis S.: EM Algorithms for PCA and SPCA. In: Advances in Neural Information Processing Systems. pp 626–632. (1998)
3. Van der Maaten L.J.P, Postma E.O., van den Herik H.J.: Dimensionality reduction: A comparative review. In: Journal of Machine Learning Research 10. pp 66–71. (2009)
4. Patridge M., Calvo R.: Fast dimensionality reduction and Simple PCA. In: Intelligent Data Analysis, 2. pp 292–298. (1997)
5. Oja E.: Principal components, minor components, and linear neural networks. In: Neural Networks. pp 927–935. (1992)
6. Lehoucq R. B., Sorensen D. C. , Yang C.: ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods. In: Software, Environments, and Tools 6. (1998)
7. Grbovic M., Dance R. C., Vucetic S.: Sparse Principal Component Analysis with Constraints. In: Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence. (2012)

8. Perronnin F., Liu Y., Sánchez J., Poirier H.: Large-scale image retrieval with compressed Fisher vectors. In: The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition. pp 3384–3391. (2010)
9. Dekel O., Keshet J., Singer Y.: Large margin hierarchical classification. In: ICML '04: Proceedings of the twenty first international conference on Machine learning, pp 27. (2004)
10. Yang Y., Liu X.: A re-examination of text categorization methods. In: ACM Press. pp 42–49. (1999)
11. Fan R.-E., Chang K.-W., Hsieh C.-J., Wang X.-R., Lin C.-J.: LIBLINEAR: A library for large linear classification Journal of Machine Learning Research 9. pp 1871–1874. (2008)
12. Venables K. V., Ripley B. D.: Modern Applied Statistics with S. Springer-Verlag (2002)
13. Setiono R., Liu H.: Chi2: Feature selection and discretization of numeric attributes. In: Proceedings of the Seventh IEEE International Conference on Tools with Articial Intelligence, (1995).
14. Liu T., Yang Y., Wan H., Zeng H., Chen Z., Ma W.: Support Vector Machines Classification with a Very Large-scale Taxonomy. In: SIGKDD Explor. Newsl. pp 36–43. (2005)
15. Chang C., Lin C.: LIBSVM : a library for support vector machines. In: ACM Transactions on Intelligent Systems and Technology. (2011)