Effective and Efficient Multilabel Classification in Domains with Large Number of Labels

Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas

Department of Informatics,
Aristotle University of Thessaloniki,
54124 Thessaloniki, Greece
{greg,katak,vlahavas}@csd.auth.gr

Abstract. This paper contributes a novel algorithm for *effective* and *computationally efficient* multilabel classification in domains with large label sets L. The HOMER algorithm constructs a Hierarchy Of Multilabel classifiERs, each one dealing with a much smaller set of labels compared to L and a more balanced example distribution. This leads to improved predictive performance along with linear training and logarithmic testing complexities with respect to $|L|$. Label distribution from parent to children nodes is achieved via a new balanced clustering algorithm, called balanced k means.

1 Introduction

Traditional *single-label* classification is concerned with learning from a set of examples that are associated with a single label λ from a set of disjoint labels L, $|L| > 1$. If $|L| = 2$, then the learning task is called *binary* classification (or *filtering* in the case of textual and web data), while if $|L| > 2$, then it is called *multi-class* classification. In *multilabel* classification [1], the examples are associated with a set of labels $Y \subseteq L$.

This work considers the task of multilabel classification in domains with large number of labels (hundreds or more). Text categorization [2, 3], protein function classification [4, 5] and semantic annotation of multimedia [6] are examples of such domains.

The high dimensionality of the label space may challenge a multilabel classification algorithm in many ways. Firstly, the number of training examples annotated with each particular label will be significantly less than the total number of examples. This is similar to the class imbalance problem in single-label classification [7]. Secondly, the computational cost of training a multilabel classifier may be strongly affected by the number of labels. There are simple algorithms (e.g. binary relevance) with linear complexity with respect to $|L|$, but there are also more advanced methods [8] whose complexity is worse. Finally, although the complexity of using a multilabel classifier for prediction is linear with respect to $|L|$ in the best case, this may still be inefficient for applications requiring fast response times.

The main contribution of this work is a novel algorithm for *effective* and *computationally efficient* multilabel classification in domains with many labels, called HOMER[1] (Hierarchy Of Multilabel classifiERs). HOMER constructs a hierarchy of multilabel classifiers, each one dealing with a much smaller set of labels compared to L and a more balanced example distribution. This leads to improved predictive performance along with linear training and logarithmic testing complexities with respect to $|L|$.

One of the main processes within HOMER is the even distribution of a set of labels into k disjoint subsets so that similar labels are placed together and dissimilar apart. Such a task has been considered in the past in the literature under the name *balanced clustering* [9]. A secondary contribution of this paper is a new algorithm for this task, called *balanced k means*.

The rest of the paper is structured as follows. Sections 2 and 3 describe HOMER and balanced k means respectively. Sections 4 and 5 present the setup and results respectively of the experimental work comparing HOMER to binary relevance, which is the most popular and computationally efficient multilabel classification method. Finally, Section 6 concludes this paper and points to potential extensions.

2 The HOMER Algorithm

HOMER follows the divide-and-conquer paradigm of algorithm design. The main idea is the transformation of a multilabel classification task with a large set of labels L into a tree-shaped hierarchy of simpler multilabel classification tasks, each one dealing with a small number $k << |L|$ of labels.

Each node n of this tree contains a set of labels $L_n \subseteq L$. There are $|L|$ leaves, each one containing a singleton (single element set) $\{\lambda_j\}$ with a different label λ_j of L. Each internal node n contains the union of the label sets of its children, $L_n = \bigcup L_c, c \in \text{children(n)}$. The root contains all labels, $L_{\text{root}} = L$.

We define the concept of *meta-label* of a node n, μ_n, as the disjunction of the labels contained in that node, $\mu_n \equiv \bigvee \lambda_j, \lambda_j \in L_n$. Meta-labels have the following semantics: a training example can be considered annotated with meta-label μ_n, if it is annotated with at least one of the labels in L_n.

Each internal node n of the hierarchy also contains a multilabel classifier h_n. The task of h_n is the prediction of one or more of the meta-labels of its children. Therefore, the set of labels for h_n is $M_n = \{\mu_c \mid c \in \text{children}(n)\}$. Figure 1 shows a sample hierarchy produced for a multilabel classification task with 8 labels $\{\lambda_1, \ldots, \lambda_8\}$.

For the multilabel classification of a new instance x, HOMER starts with h_{root} and follows a recursive process forwarding x to the multilabel classifier h_c of a child node c only if μ_c is among the predictions of $h_{\text{parent(c)}}$. Eventually, this process may lead to the prediction of one or more single-labels by the multi-label classifier(s) just above the corresponding leaf(ves). The union of these predicted

[1] Homer was an ancient Greek epic poet, alleged author of *Iliad* and *Odyssey*.

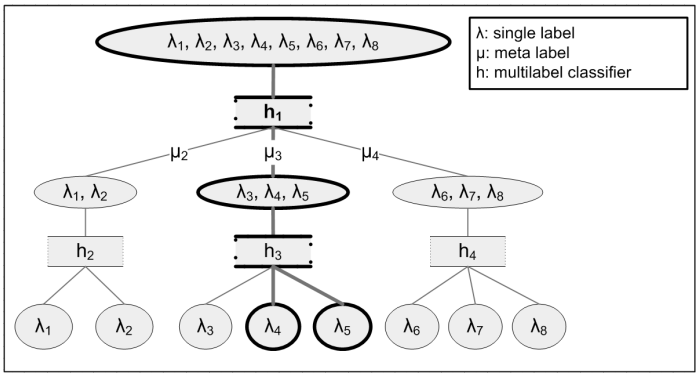


Fig. 1. Sample hierarchy for a multilabel classification task with 8 labels

single-labels is the output of the proposed approach in this case, while the empty set is returned otherwise.

For training HOMER, we assume the existence of a set $D = \{(\mathbf{x}_i, Y_i) \mid i = 1 \dots |D|\}$ of training examples, each one consisting of a feature vector $\mathbf{x}_i$ and a set of labels $Y_i \subseteq L$. HOMER creates the tree recursively in a top-down depth-first fashion starting with the root. At each node n, k children nodes are first created, unless $|L_n| < k$, in which case the number of children is $|L_n|$. Each such child n filters the data of its parent, keeping only the examples that are annotated with at least one of its own labels: $D_n = \{(\mathbf{x}_i, Y_i) \mid (\mathbf{x}_i, Y_i) \in D_{\text{parent}(n)}, Y_i \cap L_n \neq \emptyset\}$. The root uses the whole training set, $D_{\text{root}} = D$. Two main processes are then sequentially executed: a) the labels of the current node are distributed into k disjoint subsets, one for each child of the current node, and b) a multilabel classifier is trained for the prediction of the meta-labels of its children. The approach recurses into each child node that contains more than a single label.

In the latter process, each internal node n transforms its examples $(\mathbf{x}_i, Y_i) \in D_n$ into meta-examples $(\mathbf{x}_i, Z_i)$, where $Z_i = \{\mu_c \mid c \in \mathit{children}(n), Y_i \cap L_c \neq \emptyset\}$. These meta-examples are used for training h_n.

The main issue in the former process is how to distribute the labels of L_n to the k children. We argue that labels should be *evenly* distributed to k subsets in a way such that labels belonging to the same subset are as *similar* as possible. Such a task can be thought of as clustering with the additional constrain of equal cluster size. It has been considered in the past in the literature, under the name *balanced clustering* [9]. HOMER can use existing balanced clustering algorithms for this process (see Section 3.2). A new balanced clustering algorithm, called balanced k means, is presented in Section 3.

The justification in favor of similarity-based distribution is that if similar labels of a node n are placed in the same subset, then only a few (ideally just one) meta-labels of h_n will be predicted and thus the rest sub-trees will not be activated. This will lead to reduced cost during the operation and testing of

HOMER. Another expected benefit is that each child node will probably contain less training examples. The justification in favor of even distribution is that the multilabel classifiers at each node will deal with a more balanced distribution of positive examples for each meta-label. This is expected to lead to improved predictive performance.

2.1 Computational Complexity

To simplify the analysis, we will assume that $|L| = k^d$, for some integer d, which means that the hierarchy is a perfect k-ary tree of depth d (all internal nodes have k children and all leaves are at the same level). The number of internal nodes in such a tree is equal to $(|L| - 1)/(k - 1)$ [10].

The complexity of the balanced clustering process at each node n depends on the actual algorithm being used and can range from $O(|L_n|)$ to $O(|L_n|^3)$ (see Section 3.2). L_n is equal to L at the root, but subsequently decreases exponentially with the tree's depth. Therefore, if $f(|L_n|)$ is a function of the complexity of the balanced clustering algorithm with respect to L_n, then the overall complexity of HOMER with respect to this algorithm is $O(f(|L|)$. In other words HOMER retains the complexity of the balanced clustering algorithm.

Consider for example that $f(|L_n|) = |L_n|^2$. Then at the root we have a cost of $|L|^2$ while at the second level we have k additional costs of $(|L|/k)^2$, i.e. an additional cost of $|L|^2/k$. At the next level we have k^2 additional costs of $\left(|L|/k^2\right)^2$, i.e. an additional cost of $|L|^2/k^2$. This is a sum of a geometric series leading to a total cost of $2|L|^2$ when the depth of the tree approaches infinity.

At each node, there is also the cost of training the hierarchical multilabel classifier, which depends on the algorithm being used. If $g(|L|)$ is a function of the complexity of the multilabel classifier with respect to the number of labels $|L|$ then the complexity at node n is $O(g(k))$. Given that there are $(|L| - 1)/(k - 1)$ internal nodes, then the overall complexity of this process of HOMER is linear with respect to $|L|$ irrespectively of the multilabel classifier being used. This is an important benefit, especially if the complexity of the multilabel classifier used is worse than linear with respect to $|L|$.

Another interesting point is that the size of D_n at each node n also reduces with the depth of the tree. The rate of reduction depends on many factors (average number of labels per example, overlap of labels in examples, k) and cannot be explicitly represented.

To conclude, HOMER's overall training complexity is $O(f(|L|) + |L|)$, where $f(|L|)$ is the complexity of the balanced clustering algorithm with respect to a set of labels L.

During testing or operation, the cost depends on the average multilabel classifiers that are activated. Assuming that each example is annotated with a small number of labels compared to L, that HOMER outputs a small number of labels as well and that a different root-to-leaf path is followed for the prediction of each label, then we can consider the complexity as $O\left(log_k(|L|)\right)$. Compared to the typical $O(|L|)$ complexity, HOMER offers an important benefit, especially for domains where online performance is critical.

3 Balanced k Means

We developed a new balanced clustering algorithm, called balanced k means, which extends the well-known k means algorithm with an explicit constraint on the size of each cluster. Since the objects of clustering are labels, we only consider the label part, Y_i, of the data at the current node of the tree, $(\mathbf{x}_i, Y_i) \in D_n$, and in particular a subset, W_i, of Y_i with just the labels at the current node $W_i = Y_i \cap L_n$. The algorithm accepts as input a set of labels $L_n \subseteq L$, a set of label data W_i, the number of partitions k and a number of iterations it, and outputs k disjoint subsets of L_n with approximately equal size. Figure 2 shows the balanced k means algorithm in pseudo-code.

```
Input: number of clusters k, labels L_n, label data W_i, iterations it
Output: k balanced clusters of labels
for i ← 1 to k do
    // initialize clusters and cluster centers
    C_i ← ∅ ;
    c_i ← random member of L_n ;
while it > 0 do
    foreach λ ∈ L_n do
        for i ← 1 to k do
            d_λi ← distance(λ, c_i, W_i)
        finished ← false;
        ν ← λ ;
        while not finished do
            j ← arg min_i d_νi;
            Insert sort (ν, d_ν) to sorted list C_j;
            if |C_j| > ⌈|L_n|/k⌉ then
                ν ← remove last element of C_j;
                d_νj ← ∞ ;
            else
                finished ← true;
    recalculate centers;
    it ← it − 1
return C_1, ..., C_k;
```

Fig. 2. Balanced k-Means Algorithm

The key element in the algorithm is that for each cluster i we maintain a list of labels, C_i, sorted in ascending order of distance to the cluster centroid c_i. When the insertion of a label into the appropriate position of the sorted list of a cluster, causes its size to exceed the maximum allowed number of labels (approximately equal to the number of items divided by the number of clusters), the last (furthest) element in the list of this cluster is inserted to the list of the

next most proximate cluster. This may lead to a cascade of $k-1$ additional insertions in the worst case. Another difference compared to k means, is that we limit the number of iterations using a user-specified parameter, it, as no investigation of convergence was attempted.

To calculate the centroids of clusters and the distance of labels to these centroids, we first expand the label data W_i into binary vectors $w_{ij}, j = 1 \dots |L_n|$, where $w_{ij} = 1$ if $\lambda_j \in W_i$ and $w_{ij} = 0$ otherwise. Then centroids are calculated as the mean vectors of the labels in each cluster, while the Euclidean metric is used to measure the distance between centroid and label vectors.

3.1 Computational Complexity

At each iteration of the balanced k means algorithm, we loop over all labels $|L_n|$, calculate their distance to the k cluster centers with an $O(|D_n|)$ complexity and insert them into a sorted list of max size $|L_n|/k$, which has complexity $O(|L_n|)$. This may result into a cascade of $k-1$ additional insertions into sorted lists in the worst case, but the complexity remains $O(|L_n|)$. So the total cost of the balanced k means algorithm is $O(|L_n||D_n| + |L_n|^2)$. As typically $|L_n| \ll |D_n|$, the algorithm can efficiently partition labels into balanced clusters based on very large datasets.

3.2 Related Work

Contrary to our scenario, typical applications of (balanced) clustering involve a very large number of objects (millions). Therefore, the scalability of balanced clustering algorithms is an important issue. A sampling-based algorithm with complexity $O(|L_n|log|L_n|)$ has been proposed in [9].

The frequency-sensitive k means algorithm [11] is a fast algorithm for balanced clustering (complexity of $O(|L_n|)$). It extends k-means with a mechanism that penalizes the distance to clusters proportionally to their current size, leading to fairly balance clusters in practice. However, it does not guarantee that every cluster will have at least a pre-specified number of elements.

Another approach to balanced clustering extends k means by considering the cluster assignment process at each iteration as a minimum cost-flow problem [12]. Such an approach has a complexity of $O(|L_n|^3)$, which is worse than the proposed algorithm.

Finally, according to [9], the balanced clustering problem can also be solved with efficient min-cut graph partitioning algorithms with the addition of soft balancing constraints. Such approaches have a complexity of $(O(|L_n|^2)$, similarly to the proposed algorithm.

4 Experimental Setup

4.1 Datasets

Two datasets are used in the experiments. The first one, called *delicious* was extracted from the `del.icio.us` social bookmarking site on the 1st of April

2007. It contains textual data of web pages along with their tags, and is used to train a multilabel classifier for automated tag suggestion. The second dataset, called *mediamill*, was introduced in a video annotation challenge [6]. Table 4.1 presents statistics of both datasets, including the number of examples, attributes, labels and the average number of labels in the examples (label cardinality) along with its normalized version (cardinality divided by the number of labels), called label density [1].

Table 1. Information and multilabel statistics for the data sets used in the experiments

	Examples		Attributes			Label	Label
Dataset	Train	Test	Numeric	Discrete	Labels	Cardinality	Density
delicious	12920	3185	0	500	983	19.020	0.019
mediamill	30993	12914	120	0	101	4.376	0.043

In the rest of this subsection we describe the extraction process for the *delicious* dataset. Initially, we retrieved the 140 most popular tags in `del.icio.us`[2]. Then, for each of these tags we retrieved the 1000 most recent bookmarks and selected the 200 most popular, based on the number of users that included them in their personal bookmarks. This resulted into 28000 bookmarks, including duplicate entries. After the removal of duplicates, 19740 bookmarks remained.

Each bookmark is usually annotated with more than one tag, either because the user who originally posted the bookmark annotated it with more than one tags or because other users assigned different tags to it. For each bookmark, we extracted the 25 most popular tags, as listed in the bookmark's web page. This resulted into a total of 22139 distinct tags for all bookmarks. To discharge the dataset from rarely used tags we removed tags used in less than 10 bookmarks. This led to a final set of 983 tags.

We then retrieved the Web page of each bookmark and extracted its content. The total number of distinct words from all pages was considerably large (808255). To discharge the dataset from rare words, those appearing in less than 10 bookmarks were removed. This led to a reduced vocabulary of 7234 words. The content of web pages was represented using the Boolean bag-of-words model.

In order to further reduce the computational cost of training, feature selection was applied in the following way. We used the χ^2 feature ranking method separately for each label in order to obtain a ranking of all features for that label. We then selected the top 500 features based on the their maximum rank over all labels.

After the aforementioned preprocessing, and the removal of empty examples (examples with no features or labels) the final version of the dataset included 16105 instances, 983 classes and 500 features. Various versions of the dataset are publicly available at `http://mlkd.csd.auth.gr/multilabel.html`.

[2] Extracted from the Web page at `http://del.icio.us/tag/`

4.2 Methods

We focus our experiments on the scalability of the binary relevance method (BR), since it is the most widely used multilabel classification method. Both the training and the testing phases of BR are already linear with respect to $|L|$. We compare BR against HOMER using BR as the multilabel classifier at each node. To reduce the computational cost of the experiments, we use naive Bayes as the base classifier for the decomposed binary tasks. We evaluate the performance of methods using a hold-out set.

HOMER is run with $k = 2 \dots 8$. In addition to the balanced k means algorithm we examine two different approaches to distributing the labels into subsets. The first variation, called HOMER-R, distributes evenly but randomly the labels into the k subsets. The motivation here is to examine the benefits of clustering on top of the even distribution of labels. The second variation called HOMER-K, distributes the labels using the k means algorithm without any constraint on cluster sizes. The motivation in this case is to examine the benefits of even distribution on top of the clustering. The default version of HOMER with balanced k means is called HOMER-B.

5 Results and Discussion

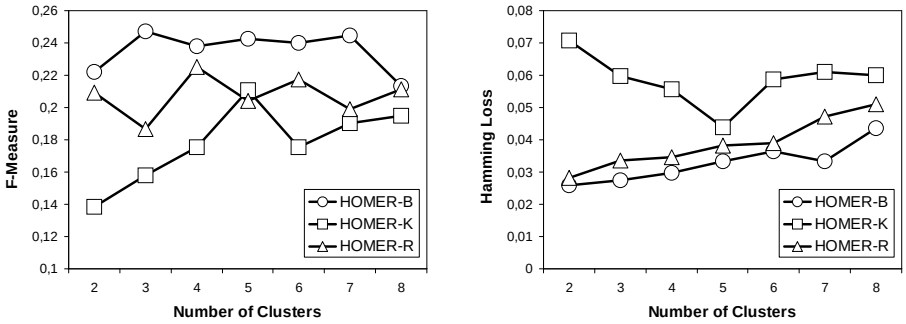
5.1 Quality of Prediction

Several metrics exist for the evaluation of the predictive performance of multilabel classifiers [8]. We present results based on two representative ones, the Hamming loss and the micro averaged F-measure.

Figure 3 shows the predictive performance of HOMER and its variations in delicious with respect to the number of clusters. BR has a Hamming loss of 0.282 and an F-measure of 0.081. Therefore, we first notice that even the worst results of both HOMER and its variations are much better compared to BR. This verifies that the skewness of the distribution of the examples for each label is an important problem of BR in domains with large number of classes and that HOMER manages to alleviate it at some extent.

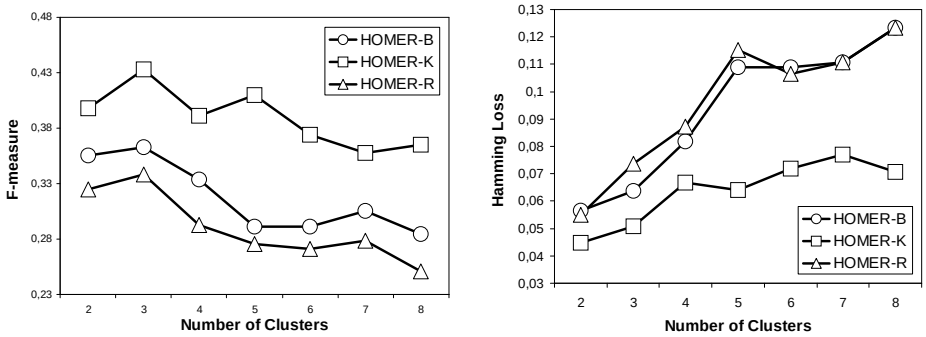
We then notice that for both metrics, HOMER-B has the best results, followed by HOMER-R and then HOMER-K. This shows that clustering improves on top of the even distribution, which is actually more important than simple clustering in this domain. HOMER-K seems to be improving with the number of clusters. Perhaps the balancing of the subsets is increased, due to the distribution of a large cluster into more smaller ones. The performance of HOMER-B and HOMER-R has a decreasing trend in terms of Hamming loss.

Figure 4 shows the predictive performance of HOMER and its variations in mediamill with respect to the number of clusters. BR has a Hamming loss of 0.331 and an F-measure of 0.157. Similarly to the previous dataset, we notice the clear benefits of HOMER independently of the label distribution approach.



(a) (b)

Fig. 3. Predictive performance of HOMER and variations in delicious



(a) (b)

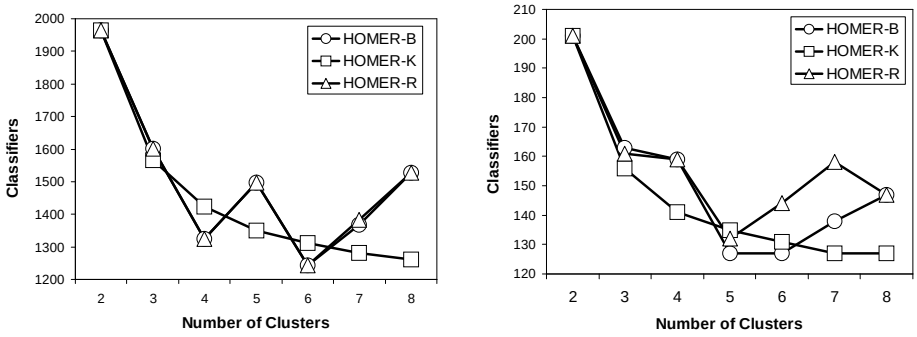
Fig. 4. Predictive performance of HOMER and variations in mediamill

However, in this case we notice that HOMER-K outperforms HOMER-B, which remains better than HOMER-R. This shows that in this domain, clustering seems to be more important than balancing the examples. A possible explanation is that balancing is more important as the number of labels increase. This will have to be examined in the future via controlled experiments. Another potential reason is that the labels of mediamill may be naturally grouped into existing balanced clusters. For example many of the 101 categories relate to people (people, people walking, people marching, etc), others to vehicles (car, bus, vehicle, etc) and so on [6]. The performance of both HOMER and its variations decreases with the number of clusters.

5.2 Training Performance

Training performance is measured via the total wall time of training, but also through two additional indices: a) number of binary classifiers trained, and b) total number of examples that these classifiers have to process.

BR trains 983 classifiers in delicious and 101 in mediamill. Figure 5 shows that HOMER and variations need to train an additional number of binary classifiers for each internal node, apart from the root, which is generally reduced with the number of clusters.



(a) delicious (b) mediamill

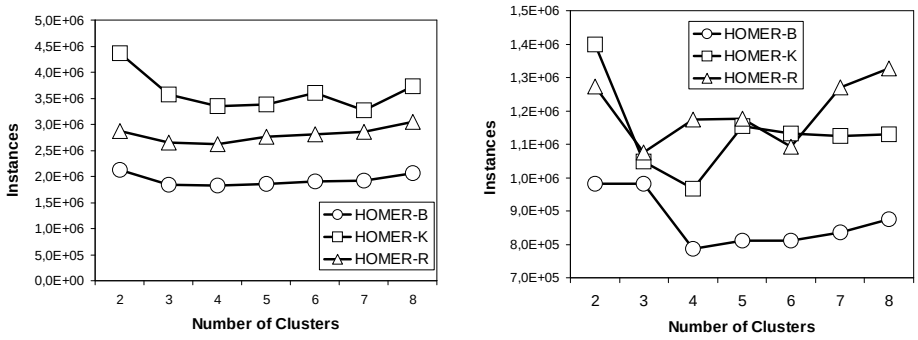
Fig. 5. Number of binary classifiers trained by HOMER and variations

However, we also notice in Figure 6 that the total number of examples processed by these methods are much less than the total number of examples processed by BR ($|L||D|$), which is more than 15 million in delicious and more than 3 million in mediamill. Notice that both balancing and clustering help reducing the examples propagated from parents to children, as HOMER-B leads to the fewest total examples in both datasets.

The total training time of BR is 24.6 minutes in delicious and 10.1 minutes in mediamill. Apart from the number of examples, the total training time of HOMER further depends on the process used to distribute the labels to subsets. We have seen that HOMER-B is quadratic with respect to $|L|$. Despite the fact, that k means is linear with respect to $|L|$, HOMER-K is also quadratic, because the computational benefits of balancing are lost. HOMER-R is linear with respect to $|L|$.

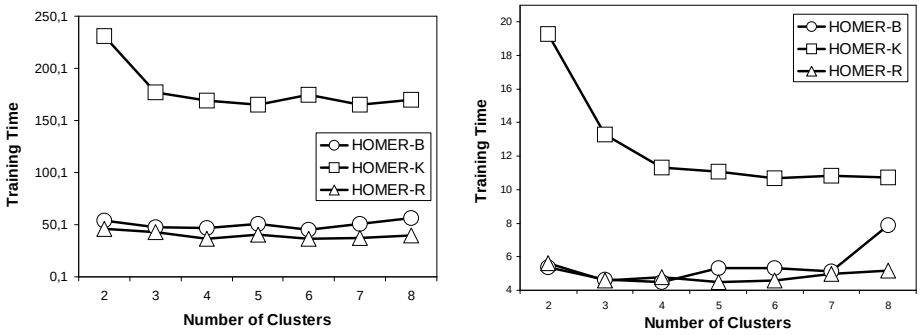
Figure 7 shows the training time of HOMER and variations with respect to the number of clusters. We see that actually HOMER-K requires more time, which can be explained by the fact that it requires a number of iterations to converge. However, we interestingly notice that HOMER-B has only a small overhead compared to HOMER-R.

This means that the total training time of HOMER and variations should actually be less than that of BR. We believe that the increased time is due



(a) delicious (b) mediamill

Fig. 6. Number of examples processed by binary classifiers of HOMER and variations



(a) delicious (b) mediamill

Fig. 7. Training time of HOMER and variations

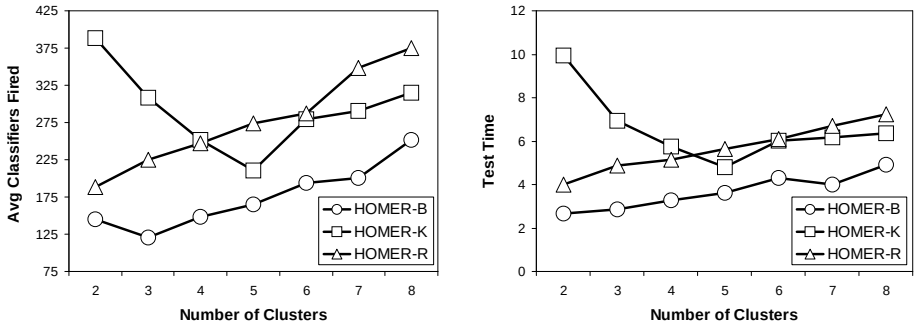
to some dominating non-optimized part of HOMER's implementation, probably the part performing the filtering of data from parent to children, as training time seems to be linearly correlated with the number of classifiers. An optimization of HOMER's implementation is among the priorities of our future work.

5.3 Testing Performance

Testing performance is measured through total wall time of testing and an additional index: average number of binary classifiers activated during the multilabel classification of a new instance.

Figure 8 shows the average number of activated classifiers and total testing time by HOMER and variations in delicious. BR activates 983 classifiers and takes 69.4 minutes of testing time. The benefit of HOMER and variants is evident, as substantially less classifiers are activated, leading to reduced total

testing time. We also notice that for HOMER and HOMER-R the average number of nodes fired and corresponding times increase with the number of clusters, while for HOMER-K there is an initial decreasing trend, which is subsequently reversed. The default version of HOMER achieves the best results.



(a) (b)

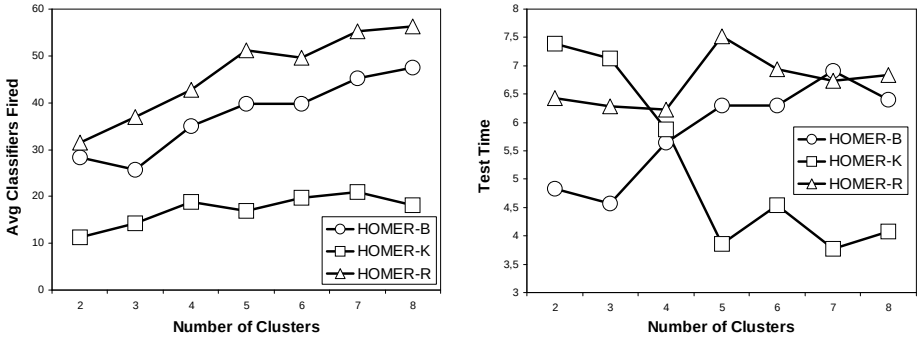
Fig. 8. Average classifiers fired and testing time of HOMER and variations in delicious with respect to the number of clusters

Figure 9 shows the average number of activated classifiers and total testing time by HOMER and variations in mediamill. BR activates 101 classifiers and takes 7.6 minutes of testing time. Improvements are noticed for this dataset as well, although to a smaller extent compared to delicious. We also notice that the number of nodes fired increases with respect to the clusters, especially for HOMER-B and HOMER-R but also for HOMER-K at a smaller pace. Test time graphs do not directly follow these trends, perhaps influenced by occasional machine loads by other processes. Measuring process time instead of wall time is another future work priority with respect to the experiments.

Most importantly, we notice again that in this domain HOMER-K performs better than HOMER-B, which consistently outperforms HOMER-R. The same explanations as in the case of quality predictions hold. It seems that either balancing is more useful in domains with larger number of models, or clustering is more useful in domains with an underlying conceptual hierarchy.

6 Conclusions and Future Work

This paper introduced a new algorithm for effective and efficient multilabel classification in domains with large number of labels, called HOMER, and presented a theoretical analysis of its expected computational benefits. In addition, the paper has empirically shown that HOMER provides more accurate predictions than the popular binary relevance method in less time. To the best of our knowl-



(a) (b)

Fig. 9. Average classifiers fired and testing time of HOMER and variations in mediamill with respect to the number of clusters

edge, the scalability of multilabel classification algorithms with respect to the dimensionality of the label space has not been explicitly considered in the past.

As a side contribution, a new algorithm for balanced clustering was proposed, called balanced k means. We stress the adjective *side*, as HOMER can operate with any balanced clustering algorithm. We do not perform an extensive study of the balanced k means algorithm in this paper, nor argue for its superiority against the related work. We merely present it as a potential instantiation of the label distribution component of HOMER, which is the main focus of this paper.

The experimental work involved a new textual dataset that was extracted from the `del.icio.us` social bookmarking site. To the best of our knowledge, this paper is the first attempt to model the modern application of automated tag suggestion as a multilabel classification task. However, we do not claim that such an approach is better or more appropriate than other existing approaches [13–15]. Such a comparison would be an interesting subject of future work, but it is outside the scope of this paper.

HOMER could be easily extended to work with an existing hierarchy. This way it will constitute a new general hierarchical multilabel classifier that can exploit recent advanced multilabel classification algorithms for increased performance. In the future, we intend to investigate this promising extension and compare to existing algorithms for hierarchical multilabel classification [16–18].

Within the next steps of this work, we also plan to put a lot of effort in enriching the empirical data. We intend to examine the scalability of BR in conjunction with Support Vector Machines (SVMs), which are more computationally demanding but also more effective algorithms. Since SVMs may have larger than linear complexity with respect to the size of the training set, the benefits of HOMER will be higher. We also intend to examine the scalability of multilabel classification methods with larger complexity than linear. Based on the theoretical analysis, the benefits of HOMER are expected to be higher in this case too.

Acknowledgements

This work was partially supported by a PENED program (EPAN M.8.3.1, No. 03Δ73), jointly funded by the European Union and the Greek Government (General Secretariat of Research and Technology/GSRT).

References

1. Tsoumakas, G., Katakis, I.: Multi-label classification: An overview. International Journal of Data Warehousing and Mining **3** (2007) 1–13
2. McCallum, A.: Multi-label text classification with a mixture model trained by em. In: Proceedings of the AAAI' 99 Workshop on Text Learning. (1999)
3. Schapire, R.E. Singer, Y.: Boostexter: a boosting-based system for text categorization. Machine Learning **39** (2000) 135–168
4. Clare, A., King, R.: Knowledge discovery in multi-label phenotype data. In: Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD 2001), Freiburg, Germany (2001) 42–53
5. Roth, V., Fischer, B.: Improved functional prediction of proteins by learning kernel combinations in multilabel settings. In: Proceeding of 2006 Workshop on Probabilistic Modeling and Machine Learning in Structural and Systems Biology (PMSB 2006), Tuusula, Finland (2006)
6. Snoek, C.G., Worring, M., van Gemert, J.C., Geusebroek, J.M., Smeulders, A.W.: The challenge problem for automated detection of 101 semantic concepts in multimedia. In: Proceedings of ACM Multimedia. (2006) 421–430
7. Chawla, N.V., Japkowicz, N., Kotcz, A.: Editorial: special issue on learning from imbalanced data sets. SIGKDD Explorations **6** (2004) 1–6
8. Tsoumakas, G., Vlahavas, I.: Random k-labelsets: An ensemble method for multilabel classification. In: Proceedings of the 18th European Conference on Machine Learning (ECML'07), Warsaw, Poland (2007) 406–417
9. Banerjee, A., Ghosh, J.: Scalable clustering algorithms with balancing constraints. Data Mining and Knowledge Discovery **13** (2006) 365–395
10. Preiss, B.R.: Data Structures and Algorithms with Object-Oriented Design Patterns in Java. Wiley (1999)
11. Banerjee, A., Ghosh, J.: Frequency-sensitive competitive learning for scalable balanced clustering on high-dimensional hyperspheres. IEEE Transactions on Neural Networks **15** (2004) 702–719
12. Bennett, K., Bradley, P., , Demiriz, A.: Constrained k-means clustering. Technical Report TR-2000-65, Microsoft Research (2000)
13. Jaschke, R., Marinho, L.B., Hotho, A., Schmidt-Thieme, L., Stumme, G.: Tag recommendations in folksonomies. In: Proceedings of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'07), Berlin, Heidelberg (2007) 506–514
14. Sood, S., Hammond, K., Owsley, S., Birnbaum, L.: TagAssist: Automatic Tag Suggestion for Blog Posts. In: Proceedings of the International Conference on Weblogs and Social Media (ICWSM 2007). (2007)
15. Chirita, P.A., Costache, S., Nejdl, W., Handschuh, S.: P-tag: large scale automatic generation of personalized annotation tags for the web. In: WWW '07: Proceedings of the 16th international conference on World Wide Web, New York, NY, USA, ACM (2007) 845–854

16. Cesa-Bianchi, N., Gentile, C., Zaniboni, L.: Incremental algorithms for hierarchical classification. Journal of Machine Learning Research **7** (2006) 31–54
17. Barutcuoglu, Z., Schapire, R.E., Troyanskaya, O.G.: Hierarchical multi-label prediction of gene function. Bioinformatics **22** (2006) 830–836
18. Blockeel, H., Schietgat, L., Struyf, J., Dzeroski, S., Clare, A.: Decision trees for hierarchical multilabel classification: A case study in functional genomics. In: Proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'06). (2006) 18–29