Predicting Unseen Labels Using Label Hierarchies in Large-Scale Multi-label Learning

Jinseok Nam[1,2,3](✉), Eneldo Loza Mencía[2,3], Hyunwoo J. Kim[4], and Johannes Fürnkranz[2,3]

[1] Knowledge Discovery in Scientific Literature, TU Darmstadt, Darmstadt, Germany
nam@cs.tu-darmstadt.de
[2] Research Training Group AIPHES, TU Darmstadt, Darmstadt, Germany
[3] Knowledge Engineering Group, TU Darmstadt, Darmstadt, Germany
[4] Department of Computer Sciences, University of Wisconsin-Madison, Madison, USA

Abstract. An important problem in multi-label classification is to capture label patterns or underlying structures that have an impact on such patterns. One way of learning underlying structures over labels is to project both instances and labels into the same space where an instance and its relevant labels tend to have similar representations. In this paper, we present a novel method to learn a joint space of instances and labels by leveraging a hierarchy of labels. We also present an efficient method for pretraining vector representations of labels, namely label embeddings, from large amounts of label co-occurrence patterns and hierarchical structures of labels. This approach also allows us to make predictions on labels that have not been seen during training. We empirically show that the use of pretrained label embeddings allows us to obtain higher accuracies on unseen labels even when the number of labels are quite large. Our experimental results also demonstrate qualitatively that the proposed method is able to learn regularities among labels by exploiting a label hierarchy as well as label co-occurrences.

1 Introduction

Multi-label classification is an area of machine learning which aims to learn a function that maps instances to a label space. In contrast to multiclass classification, each instance is assumed to be associated with more than one label. One of the goals in multi-label classification is to model the underlying structure of the label space because in many such problems, the occurrences of labels are not independent of each other.

Recent developments in multi-label classification can be roughly divided into two bodies of research. One is to build a classifier in favor of statistical dependencies between labels, and the other is devoted to making use of prior information over the label space. In the former area, many attempts have been made to exploit label patterns [6,9,24]. As the number of possible configurations of labels grows exponentially with respect to the number of labels, it is required for multi-label classifiers to handle many labels efficiently [4] or to reduce the dimensionality of

An erratum to this chapter is available at DOI: 10.1007/978-3-319-23528-8_46

A. Appice et al. (Eds.): ECML PKDD 2015, Part I, LNAI 9284, pp. 102–118, 2016.
DOI: 10.1007/978-3-319-23528-8_7

a label space by exploiting properties of label structures such as sparsity [17] and co-occurrence patterns [7]. Label space dimensionality reduction (LSDR) methods allow to make use of latent information on a label space as well as to reduce computational cost. Another way of exploiting information on a label space is to use its underlying structures as a prior. Many methods have been developed to use hierarchical output structures in machine learning [27]. In particular, several researchers have looked into utilizing the hierarchical structure of the label space for improved predictions in multi-label classification [26,30,32].

Although extensive research has been devoted to techniques for utilizing implicitly or explicitly given label structures, there remain the scalability issues of previous approaches in terms of both the number of labels and documents in large feature spaces. Consider a very large collection of scientific documents covering a wide range of research interests. In an emerging research area, it can be expected that the number of publications per year grows rapidly. Moreover, new topics will emerge, so that the set of indexing terms, which has initially been provided by domain experts or authors to describe publications with few words for potential readers, will grow as well.

Interestingly, similar problems have been faced recently in a different domain, namely *representation learning* [2]. In language modeling, for instance, a word is traditionally represented by a K-dimensional vector where K is the number of unique words, typically hundreds of thousands or several millions. Clearly, it is desirable to reduce this dimensionality to much smaller values $d \ll K$. This can, e.g., be achieved with a simple log-linear model [21], which can efficiently compute a so-called *word embedding*, i.e., a lower-dimensional vector representations for words. Another example for representation learning is a technique for learning a joint embedding space of instances and labels [31]. This approach maximizes the similarity between vector representations of instances and relevant labels while projecting them into the same space.

Inspired by the log-linear model and the joint space embedding, we address large-scale multi-label classification problems, in which both hierarchical label structures are given *a priori* as well as label patterns occur in the training data. The mapping functions in the joint space embedding method can be used to rank labels for a given instance, so that relevant labels are placed at the top of the ranking. In other words, the quality of such a ranking depends on the mapping functions. As mentioned, two types of information on label spaces are expected to help us to train better joint embedding spaces, so that the performance on unseen data can be improved. We focus on exploiting such information so as to learn a mapping function projecting labels into the joint space. The vector representations of labels by using this function will be referred to as *label embeddings*. While *label embeddings* are usually initialized randomly, it will be beneficial to learn the joint space embedding method taking label hierarchies into consideration when label structures are known. To this end, we adopt the above-mentioned log-linear model which has been successfully used to learn *word embeddings*.

Learning *word embeddings* relies fundamentally on the use of the context information, that is, a fixed number of words surrounding that word in a sentence or a document. In order to adapt this idea to learning *label embeddings*, we need to define context information in a label space, where, unlike in textual documents, there is no sequence information which can be used to define the context of words. We use, instead, *pairwise* relationships in label hierarchies and in label co-occurrence patterns.

There are two major contributions of this work: 1) We build efficient multi-label classifiers which employ label hierarchies so as to predict unseen labels. 2) We provide a novel method to efficiently learn label representations from hierarchical structures over labels as well as their co-occurrence patterns.

2 Multi-label Classification

In *multi-label classification*, assuming that we are given a set of training examples $\mathcal{D} = \{(\mathbf{x}_n, \mathcal{Y}_n)\}_{n=1}^N$, our goal is to learn a classification function $f : \mathbf{x} \rightarrow \mathcal{Y}$ which maps an instance $\mathbf{x}$ to a set of *relevant* labels $\mathcal{Y} \subseteq \{1, 2, \cdots, L\}$. All other labels $\bar{\mathcal{Y}} = \{1, 2, \cdots, L\} \backslash \mathcal{Y}$ are called *irrelevant*. Often, it is sufficient, or even required, to obtain a list of labels ordered according to some relevance scoring functions.

In *hierarchical multi-label classification* (HMLC) labels are explicitly organized in a tree usually denoting a *is-a* or *composed-of* relation. Several approaches to HMLC have been proposed which replicate this structure with a hierarchy of classifiers which predict the paths to the correct labels [5,30,32]. Although there is evidence that exploiting the hierarchical structure in this way has advantages over the flat approach [3,5,30], some authors unexpectedly found that ignoring the hierarchical structure gives better results. For example, in [32] it is claimed that if a strong flat classification algorithm is used the lead vanishes. Similarly, in [30] it was found that learning a single decision tree which predicts probability distributions at the leaves outperforms a hierarchy of decision trees. One of the reasons may be that hierarchical relations in the output space are often not in accordance with the input space, as claimed by [15] and [32]. Our proposed approach aims at overcoming this problem as it learns an embedding space where similarities in the input, output and label hierarchies are jointly respected.

3 Model Description

3.1 Joint Space Embeddings

Weston et al. [31] proposed an efficient online method to learn ranking functions in a joint space of instances and labels, namely *Wsabie*. Under the assumption that instances which have similar representation in a feature space tend to be associated with similar label sets, we find joint spaces of both instances and labels where the relevant labels for an instance can be separated from the irrelevant ones with high probability.

Formally, consider an instance $\mathbf{x}$ of dimension D and a set of labels $\mathcal{Y}$ associated with $\mathbf{x}$. Let $\phi(\mathbf{x}) = \mathbf{W}\mathbf{x}$ denote a linear function which projects the original feature representations of an instance $\mathbf{x}$ to a d-dimensional joint space, where $\mathbf{W} \in \mathbb{R}^{d \times D}$ is a transformation matrix. Similarly, let $\mathbf{U}$ be a $d \times L$ matrix that maps labels into the same joint d-dimensional space. A label $i \in \mathcal{Y}$ can then be represented as a d-dimensional vector $\mathbf{u}_i$, which is the i-th column vector of $\mathbf{U}$. We will refer to the matrix $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_L]$ as *label embeddings*. The objective function is given by

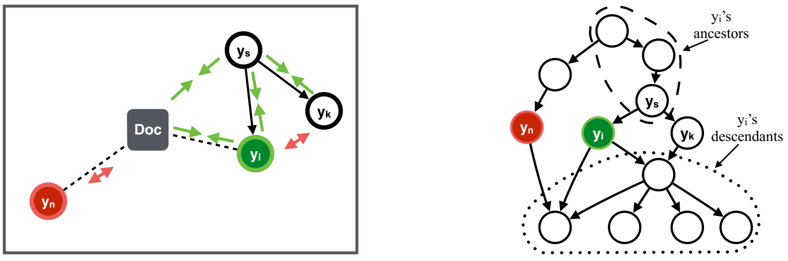
$$\mathcal{L}(\boldsymbol{\Theta}_F; \mathcal{D}) = \sum_{n=1}^{N} \sum_{i \in \mathcal{Y}_n} \sum_{j \in \overline{\mathcal{Y}}_n} h(r_i(\mathbf{x}_n))\, \ell(\mathbf{x}_n, y_i, y_j) \tag{1}$$

with the *pairwise hinge loss* function $\ell(\mathbf{x}_n, y_i, y_j) = \left[m_a - \mathbf{u}_i^T \phi(\mathbf{x}_n) + \mathbf{u}_j^T \phi(\mathbf{x}_n)\right]_+$ where $r_i(\cdot)$ denotes the rank of label i for a given instance $\mathbf{x}_n$, $h(\cdot)$ is a function that maps this rank to a real number (to be introduced shortly in more detail), $\overline{\mathcal{Y}}_n$ is the complement of $\mathcal{Y}_n$, $[x]_+$ is defined as x if $x > 0$ and 0 otherwise, $\boldsymbol{\Theta}_F = \{\mathbf{W}, \mathbf{U}\}$ are model parameters, and m_a is a real-valued parameter, namely the *margin*. The relevance scores $\mathbf{s}(\mathbf{x}) = [s_1(\mathbf{x}), s_2(\mathbf{x}), \cdots, s_L(\mathbf{x})]$ of labels for a given instance $\mathbf{x}$ can be computed as $s_i(\mathbf{x}) = \mathbf{u}_i^T \phi(\mathbf{x}) \in \mathbb{R}$. Then, the rank of label i with respect to an instance $\mathbf{x}$ can be determined based on the relevance scores $r_i(\mathbf{x}) = \sum_{j \in \overline{\mathcal{Y}}, j \neq i} [m_a - s_i(\mathbf{x}) + s_j(\mathbf{x})]_+$. It is prohibitively expensive to compute such rankings exactly when L is large. We use instead its approximation to update parameters $\boldsymbol{\Theta}_F$ given by $r_i(\mathbf{x}) \approx \left\lfloor \frac{L - |\mathcal{Y}|}{T_i} \right\rfloor$ where $\lfloor \cdot \rfloor$ denotes the floor function and T_i is the number of trials to sample an index j yielding incorrect ranking against label i such that $m_a - s_i(\mathbf{x}) + s_j(\mathbf{x}) > 0$ during stochastic parameter update steps. Having an approximate rank $r_i(\mathbf{x})$, we can obtain a weighted ranking function $h(r_i(\mathbf{x})) = \sum_{k=1}^{r_i(\mathbf{x})} \frac{1}{k}$, which is shown to be an effective way of optimizing precision at the top of rankings.

3.2 Learning with Hierarchical Structures Over Labels

Wsabie is trained in a way that the margin of similarity scores between positive associations $\mathbf{u}_p^T \phi(\mathbf{x})$ and negative associations $\mathbf{u}_n^T \phi(\mathbf{x})$ is maximized, where $\mathbf{u}_p$ and $\mathbf{u}_n$ denote the embeddings of relevant and irrelevant labels, respectively, for an instance $\mathbf{x}$. In practice, this approach works well if label patterns of test instances appear in training label patterns. If there are few or no training instances for some labels, the model may fail to make predictions accurately on test instances associated with those labels. In such cases, a joint space learning method could benefit from label hierarchies. In this section, we introduce a simple and efficient joint space learning method by adding a regularization term which employs label hierarchies, hereafter referred to as *Wsabie*$_H$.

Notations. Consider multi-label problems where label hierarchies exist. Label graphs are a natural way to represent such hierarchical structures. Because it is possible for a label to have more than one parent node, we represent a hierarchy of labels in a directed acyclic graph (DAG). Consider a graph $\mathcal{G} = \{V, E\}$ where V denotes a set of nodes and E represent a set of connections between nodes. A



(a) A joint space of instances and labels (b) A label hierarchy

Fig. 1. An illustrative example of our proposed method. A label y_i (green circle) indicates a relevant label for a document (rectangle) while y_n (red circle) is one of the irrelevant labels. In the joint space, we learn representations for the relevant label, its ancestor y_s, and the document to be similar whereas the distance between the document and the irrelevant label is maximized. Also, the parent label, y_s, and its children are forced to be similar while sibling labels of y_i, i.e. y_k, are kept away from each other.

node $u \in V$ corresponds to a label. A directed edge from a node u to a node v is denoted as $e_{u,v}$, in which case we say that u is a parent of v and v is a child of u. The set of all parents / children of v is denoted with $\mathcal{S}_{\mathcal{P}}(v)$ / $\mathcal{S}_{\mathcal{C}}(v)$. If there exists a directed path from u to v, u is an ancestor of v and v is a descendant of u, the set of all ancestors / descendants is denoted as $\mathcal{S}_{\mathcal{A}}$ / $\mathcal{S}_{\mathcal{D}}(u)$.

Label structures as regularizers. As an example, let us consider three labels, "computer science" (CS), "artificial intelligence" (AI), and "software engineering" (SE). The label CS can be viewed as a parent label of AI and SE. Given a paper dealing with problems in artificial intelligence and having AI as a label, we wish to learn a joint embedding model in a way that it is also highly probable to predict CS as a relevant label. Following our hypothesis in label spaces, even though we have no paper of software engineering, a label hierarchy allows us to make *reasonable* predictions on such a label by representing label SE close to label CS in a vector space. In order to prevent the model from converging to trivial solutions that representations of all three labels are identical, it is desired that sibling labels such as AI and SE in the hierarchy are well separated from each other in a joint embedding space. For an illustration of our method, see Fig. 1.

Formally, we can achieve this by defining a regularization term Ω, which takes into account the hierarchical label structure

$$\begin{aligned}\Omega(\mathbf{\Theta}_H) = &\sum_{n=1}^{N} \frac{1}{\mathcal{Z}_A} \sum_{i \in \mathcal{Y}_n} \sum_{s \in \mathcal{S}_{\mathcal{A}}(i)} - \log p(y_s | y_i, \mathbf{x}_n) \\ &+ \sum_{l=1}^{L} \sum_{q \in \mathcal{S}_{\mathcal{P}}(l)} \sum_{\substack{k \in \mathcal{S}_{\mathcal{C}}(q) \\ k \neq l}} h(r_q(\mathbf{u}_l)) \left[m_b - \mathbf{u}_q^T \mathbf{u}_l + \mathbf{u}_k^T \mathbf{u}_l \right]_+ \end{aligned} \tag{2}$$

where m_b is the margin, $\mathcal{Z}_A = |\mathcal{Y}_n||\mathcal{S}_\mathcal{A}(i)|$, and $p(y_s|y_i, \mathbf{x}_n)$ denotes the probability of predicting an ancestor label s of a label i given i and an instance $\mathbf{x}_n$ for which i is relevant. More specifically, the probability $p(y_s|y_i, \mathbf{x}_n)$ can be defined as

$$p(y_s|y_i, \mathbf{x}_n) = \frac{\exp(\mathbf{u}_s^T \hat{\mathbf{u}}_i^{(n)})}{\sum_{v \in L} \exp(\mathbf{u}_v^T \hat{\mathbf{u}}_i^{(n)})}, \tag{3}$$

where $\hat{\mathbf{u}}_i^{(n)} = \frac{1}{2}(\mathbf{u}_i + \phi(\mathbf{x}_n))$ is the averaged-representation of a label i and the n-th instance in a joint space. Intuitively, this regularizer forces labels, which share the same parent label, to have similar vector representations as much as possible while keeping them separated from each other. Moreover, an instance $\mathbf{x}$ has the potential to make good predictions on some labels even though they do not appear in the training set only if their descendants are associated with training instances.

Adding Ω to Eq. 1 results in the objective function of *Wsabie*$_H$

$$\mathcal{L}(\mathbf{\Theta}_H; \mathcal{D}) = \sum_{n=1}^{N} \sum_{i \in \mathcal{Y}_n} \sum_{j \in \bar{\mathcal{Y}}_n} h(r_i(\mathbf{x}_n))\, \ell(\mathbf{x}_n, y_i, y_j) + \lambda \Omega(\mathbf{\Theta}_H) \tag{4}$$

where λ is a control parameter of the regularization term. If we set $\lambda = 0$, then the above objective function is equivalent to the objective function of *Wsabie* in Eq 1.

3.3 Efficient Gradients Computation

Due to the high computational cost for computing gradients for the softmax function in Eq. 3, we use *hierarchical softmax* [22,23] which reduces the gradient computing cost from $\mathcal{O}(L)$ to $\mathcal{O}(\log L)$. Similar to [21], in order to make use of the *hierarchical softmax*, a binary tree is constructed by Huffman coding, which yields binary codes with variable length to each label according to $|\mathcal{S}_\mathcal{D}(\cdot)|$. Note that by definition of the Huffman coding all L labels correspond to leaf nodes in a binary tree, called the Huffman tree. Instead of computing L outputs, the *hierarchical softmax* computes a probability of $\lceil \log L \rceil$ binary decisions over a path from the root node of the tree to the leaf node corresponding to a target label, say, y_j in Eq. 3.

More specifically, let $C(y)$ be a codeword of a label y by the Huffman coding, where each bit can be either 0 or 1, and $I(C(y))$ be the number of bits in the codeword for that label. $C_l(y)$ is the l-th bit in y's codeword. Unlike for *softmax*, for computing the *hierarchical softmax* we use the output label representations $\mathbf{U}'$ as vector representations for inner nodes in the Huffman tree. The *hierarchical softmax* is then given by

$$p(y_j|y_i) = \prod_{l=1}^{I(C(y_j))} \sigma([\![C_l(y_j) = 0]\!]\, {\mathbf{u}'}_{n(l,y_j)}^T \mathbf{u}_i) \tag{5}$$

where $\sigma(\cdot)$ is the logistic function, $[\![\cdot]\!]$ is 1 if its argument is true and -1 otherwise, and $\mathbf{u'}_{n(l,y_j)}$ is a vector representation for the l-th node in the path from the root node to the node corresponding to the label y_j in the Huffman tree. While L inner products are required to compute the normalization term in Eq. 3, the *hierarchical softmax* needs $I(C(\cdot))$ computations. Hence, the *hierarchical softmax* allows substantial improvements in computing gradients if $\mathbb{E}\left[I(C(\cdot))\right] \ll L$.

3.4 Label Ranking to Binary Predictions

It is often sufficient in practice to just predict a ranking of labels instead of a bipartition of labels, especially in settings where the learning system is comprehended as supportive [8]. On the other hand, there are several ways to convert ranking results into a bipartition. Basically all of them split the ranking at a certain position depending on a predetermined or predicted threshold or amount of relevant labels.

Instead of experimenting with different threshold techniques, we took a pragmatic stance, and simply assume that there is an oracle which tells us the actual number of relevant labels for an unseen instance. This allows us to evaluate and compare the ranking quality of our approaches independently of the performance of an underlying thresholding technique. The bipartition measures obtained by this method could be interpreted as a (soft) upper bound for any thresholding approach.

4 Experimental Setup

Datasets. We benchmark our proposed method on two textual corpora consisting of a large number of documents and with label hierarchies provided.

The RCV1-v2 dataset [19] is a collection of newswire articles. There are 103 labels and they are organized in a tree. Each label belongs to one of four major categories. The original train/test split in the RCV1-v2 dataset consists of 23,149 training documents and 781,265 test documents. In our experiments, we switched the training and the test data, and selected the top 20,000 words according to the document frequency. We chose randomly 10,000 training documents as the validation set.

The second corpus is the OHSUMED dataset [16] consisting of 348,565 scientific articles from MEDLINE. Each article has multiple index terms known as Medical Subject Headings (MeSH). In this dataset, the training set contains articles from year 1987 while articles from 1988 to 1991 belong to the test set. We map all MeSH terms in the OHSUMED dataset to 2015 MeSH vocabulary[1] in which 27,483 MeSH terms are organized in a DAG hierarchy. Originally, the OHSUMED collection consists of 54,710 training documents and 293,856 test documents. Having removed all MeSH terms that do not appear in the 2015 MeSH vocabulary, we excluded all documents that have no label from the corpus. To represent documents in a vector space, we selected unigram words that

[1] http://www.nlm.nih.gov/pubs/techbull/so14/so14_2015_mesh_avail.html

Table 1. Number of instances (M), Size of vocabulary (D), Number of labels (L), Average number of labels per instance (C), and the type of label hierarchy (HS). L subscripted k and u denote the number of *known* and *unseen* labels, respectively.

	Original datasets			**Modified datasets**				D	HS
	M	L	C	M	L_k	L_u	C		
RCV1-v2	804 414	103	3.24	700 628	82	21	1.43	20 000	Tree
OHSUMED	233 369	27 483	9.07	100 735	9570	17 913	3.87	25 892	DAG

occur more than 5 times in the training set. These pre-processing steps left us with 36,883 train documents and 196,486 test documents. Then, 10% of the training documents were randomly set aside for the validation set. Finally, for both datasets, we applied *log tf-idf* term-weighting and then normalized document vectors to unit length.

Preparation of the datasets in zero-shot settings. We hypothesize that label hierarchies provide possibilities of learning representations of unseen labels, thereby improving predictive performance for unseen data. To test our hypothesis, we modified the datasets. For the RCV1-v2 dataset, we removed all labels corresponding to non-terminals in the label hierarchy from training data and validation data while these non-terminal labels remain intact in the test set. In other words, we train models with labels corresponding to the leaves in the label hierarchy, then test them on the modified test set which only contains *unseen* labels.

Since the train and test examples of the OHSUMED dataset was split by year, the training data does not cover all labels in the test set. More specifically, there are 27,483 labels in the label hierarchy (cf. Table 1), of which only 9,570 occur in both training and test sets, which will be referred to as the set of *known* labels. Of the 12,568 labels that occur in the test set, 2,998 cannot be found in the *known* labels set, and thus form a set of *unseen* labels together with the 14,915 labels which are only available in the label hierarchy, but not present in the label patterns. In order to test predictive performance on these unseen labels, we omitted all labels in the *known* label set from the test examples. This resulted in some test examples having an empty set of labels, which were ignored for the evaluation. Finally, the above preprocessing steps left us 67,391 test examples.

The statistics of the datasets and the modified ones are summarized in Table 1.

Representing parent-child pairs of MeSH terms in a DAG. As mentioned earlier, we use parent-child pairs of MeSH terms in the 2015 MeSH vocabulary as the label hierarchy for the OHSUMED dataset. If we represent parent-child pairs of labels as a graph, it may contain cycles. Hence, we removed edges resulting in cycles as follows: 1) Pick a node that has no parent as a starting node. 2) Run Depth-First Search (DFS) from the starting node in order to detect edges pointing to nodes visited already, then remove such edges. 3) Repeat the 1 & 2 steps until all nodes having no parents are visited. There are 16 major categories in the MeSH vocabulary. In contrast to RCV1-v2, the MeSH terms are formed in complex structures so that a label can have more than one parent.

Baselines. We compare our algorithm, Wsabie$_H$, which uses hierarchical information for label embeddings, to Wsabie ignoring label hierarchies and several other benchmark algorithms. For *binary relevance* (BR), which decomposes a multi-label problem into L binary problems, we use LIBLINEAR [12] as a base learner which is a good compromise between efficiency and effectiveness in multi-label text document classification.

To address the limitations of BR, specifically, when L is large, dimensionality reduction method on label spaces, namely *Principal Label Space Transformation* (PLST) and *Conditional Principal Label Space Transformation* (CPLST), have been proposed [7,29] which try to capture label correlations before learning per-label classifiers. Instead of directly predicting labels for given instances, the *LSDR approach* learns d-output linear predictors in a reduced label space. Then, the original label space is reconstructed from the outputs of the linear predictors using the transformation matrix for reducing the label dimensionality. We use ridge regression as a linear predictor.

Pairwise decomposition has been already successfully applied for multi-label text classification [14,20]. Here, one classifier is trained for each pair of classes, i.e., a problem with L different classes is decomposed into $\frac{L(L-1)}{2}$ subproblems. At test time, all of the $\frac{L(L-1)}{2}$ base classifiers make a prediction for one of its two corresponding classes, which is interpreted as a full vote (0 or 1) for this label. Adding these up results in a ranking over the labels. To convert the ranking into a multi-label prediction, we use the *calibrated label ranking* (CLR) approach. Though CLR is able to predict cutting points of ranked lists, in this work, in order to allow a fair comparison, it also relies on an oracle to predict the number of relevant labels for a given instance (cf Section 3.4). We denote by CLR$_{svm}$ the use of CLR in combination with SVMs.

Evaluation measures. There are several measures to evaluate multi-label algorithms and they can be split into two groups; *ranking* and *bipartition* measures. If an algorithm generates a list of labels, which is sorted by relevance scores, for a given instance, ranking measures need to be considered. The most widely used ranking measures are *rank loss* (RL) and *average precision* ($AvgP$). The *rank loss* accounts for the ratio of the number of mis-ordered pairs between relevant and irrelevant labels in a ranked list of labels to all possible pairs, which defined as $RL = \frac{1}{|\mathcal{Y}||\bar{\mathcal{Y}}|}\sum_{i,j\in\mathcal{Y}\times\bar{\mathcal{Y}}}\left[r(i) > r(j)\right]_+ + \frac{1}{2}\left[r(i) = r(j)\right]_+$ where $r(\cdot)$ denotes the position of a label in the ranked list. The *average precision* quantifies the average precision at each point in the ranking where a relevant label is placed and is computed as $AvgP = \frac{1}{|\mathcal{Y}|}\sum_{i\in\mathcal{Y}}\left(\left|\left\{j\in\mathcal{Y}|r(j)\leq r(i)\right\}\right|/r(i)\right)$. Label-based measures for binary predictions can be also considered. In this work, we report the micro- and macro-averaged F-score defined as: MiF $= (\sum_{l=1}^{L} 2tp_l)/(\sum_{l=1}^{L} 2tp_l + fp_l + fn_l)$, MaF $= \frac{1}{L}\sum_{l=1}^{L} 2tp_l/(2tp_l + fp_l + fn_l)$ where tp_l, fp_l and fn_l are the number of true positives, false positives and

Table 2. Comparison of Wsabie$_H$ to baselines on the benchmarks. (Best in bold)

	RCV1-v2						OHSUMED			
	BR	PLST	CPLST	CLR$_{svm}$	Wsabie	Wsabie$_H$	BR	PLST	Wsabie	Wsabie$_H$
AvgP	94.20	92.75	92.76	**94.76**	94.34	94.39	45.00	26.50	45.72	**45.76**
RL	0.46	0.78	0.76	**0.40**	0.44	0.44	4.48	15.06	4.09	**3.72**

false negatives for label l, respectively. Throughout the paper, we present the evaluation scores of these measures multiplied by 100.

Training Details. All hyperparameters were empirically chosen based on AvgP on validation sets. The dimensionality of the joint space d was selected in a range of $\{16, 32, 64, 128\}$ for the RCV1-v2 dataset and $\{128, 256, 512\}$ for the OHSUMED dataset. The margins m_a and m_b were chosen ranging from $\{10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}\}$. We used Adagrad [10] to optimize parameters $\mathbf{\Theta}$ in Eq. 1 and 4. Let $\Delta_{i,\tau}$ be the gradient of the objective function in Eq. 4 with respect to a parameter $\theta_i \in \mathbf{\Theta}$ at time τ. Then, the update rule for parameters indexed i at time τ is given by $\theta_i^{(\tau+1)} = \theta_i^{(\tau)} - \eta_i^{(\tau)} \Delta_{i,t}$ with an adaptive learning rate per parameter $\eta_i^{(\tau)} = \eta_0 / \sqrt{\sum_{t=1}^{\tau} \Delta_{i,t}^2}$ where $\eta_0 \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ denotes a base learning rate which decrease by a factor of 0.99 per epoch. We implemented our proposed methods using a lock-free parallel gradient update scheme [25], namely *Hogwild!*, in a shared memory system since the number of parameters involved during updates is sparse even though the whole parameter space is large. For BR and CLR$_{svm}$, LIBLINEAR[12] was used as a base learner and the regularization parameter $C = \{10^{-2}, 10^{0}, 10^{2}, 10^{4}, 10^{6}\}$ was chosen by validation sets.

5 Experimental Results

5.1 Learning All Labels Together

Table 2 compares our proposed algorithm, *Wsabie$_H$*, with the baselines on the benchmark datasets in terms of two ranking measures. It can be seen that CLR$_{svm}$ outperforms the others including *Wsabie$_H$* on the RCV1-v2 dataset, but the performance gap across all algorithms in our experiments is not large. Even BR ignoring label relationship works competitively on this dataset. Also, no difference between *Wsabie* and *Wsabie$_H$* was observed. This is attributed to characteristics of the RCV1-v2 dataset that if a label corresponding to one of the leaf nodes in the label hierarchy is associated with an instance, (almost) all nodes in a path from the root node to that node are also present, so that the hierarchical information is implicitly present in the training data.

Let us now turn to the experimental results on the OHSUMED dataset which are shown on the right-hand side of Table 2. Since the dataset consists of many labels, as an LSDR approach, we include PLST only in this experiment because

Table 3. The performance of *Wsabie*$_H$ compared to its baseline on the benchmarks in *zero-shot* learning settings.

	RCV1-v2				**OHSUMED**			
	AvgP	RL	MiF	MaF	AvgP	RL	MiF	MaF
Wsabie	2.31	62.29	0.00	0.00	0.01	56.37	0.00	0.00
Wsabie$_H$	**9.47**	**30.39**	**0.50**	**1.64**	**0.06**	**39.91**	0.00	0.00

CPLST is computationally more expensive than PLST, but no significant difference was observed. Similarly, due to the computational cost of CLR$_{svm}$ with respect to the number of labels, we excluded it from the experiment. It can be seen that regardless of the choice of the regularization term, the Wsabie approaches perform better than the other methods. PLST performed poorly under the settings where the density of labels, i.e., C/L in Table 1, is very low. Moreover, we projected the original label space $L =$27,483 into a much smaller dimension $d = 512$ using a small amount of training examples. Although the difference to BR is rather small, the margin is more pronounced that on RCV1.

5.2 Learning to Predict Unseen Labels

Over the last few years, there has been an increasing interest in *zero-shot learning*, which aims to learn a function that maps instances to classes or labels that have not been seen during training. Visual attributes of an image [18] or textual description of labels [13,28] may serve as additional information for zero-shot learning algorithms. In contrast, in this work, we focus on how to exploit label hierarchies and co-occurrence patterns of labels to make predictions on such *unseen* labels. The reason is that in many cases it is difficult to get additional information for some specific labels from external sources. In particular, while using a semantic space of labels' textual description is a promising way to learn vector representations of labels, sometimes it is not straightforward to find suitable mappings of specialized labels.

Table 3 shows the results of *Wsabie* against *Wsabie*$_H$ on the modified datasets which do not contain any known label in the test set (cf. Sec. 4). As can be seen, *Wsabie*$_H$ clearly outperforms *Wsabie* on both datasets across all measures except for MiF and MaF on the OHSUMED dataset. Note that the key difference between *Wsabie*$_H$ and *Wsabie* is the use of hierarchical structures over labels during the training phase. Since the labels in the test set do not appear during training, *Wsabie* can basically only make random predictions for the unknown labels. Hence, the comparison shows that taking only the hierarchical relations into account already enables a considerable improvement over the baseline. Unfortunately, the effect is not substantial enough in order to be reflected w.r.t. MiF and MaF on OHSUMED. Note, however, that a relevant, completely unknown label must be ranked approximately as one of the top 4 labels out of 17,913 in order to count for bipartition measures in this particular setting.

In summary, these results show that the regularization of joint embedding methods is an effective way of learning representations for *unseen* labels in a tree-structured hierarchy of a small number of labels . However, if a label hierarchy is defined on more complex structures and while a fewer number of training examples exists per label, it might be difficult for $Wsabie_H$ to work well on unseen data.

6 Pretrained Label Embeddings as Good Initial Guess

From the previous experiments, we see that the regularization of $Wsabie_H$ using the hierarchical structure of labels allows us to obtain better performance for *unseen* labels. The objective function (Eq. 4) penalizes parameters of observable labels in the training data by the negative log probability of predicting their ancestors in a hierarchy. If we initialize label spaces parameterized by $\mathbf{U}$ at random, presumably, the regularizer may rather act as noise at a beginning stage of the training. Especially for OHSUMED, the label hierarchy is complex and positive documents are very few for some labels.

We address this by exploiting both label hierarchies and co-occurrence patterns between labels in the training data. Apart from feature representations of a training instance, it is possible to capture underlying structures of the label space based on the co-occurrence patterns. Hence, we propose a method to learn label embeddings from hierarchical information and *pairwise* label relationships.

The basic idea of pretraining label embeddings is to maximize the probability of predicting an ancestor given a particular label in a hierarchy as well as predicting co-occurring labels with it. Given the labels of N training instances $\mathcal{D}_\mathcal{Y} = \{\mathcal{Y}_1, \mathcal{Y}_2, \cdots, \mathcal{Y}_N\}$, the objective function is to maximize the average log probability given by

$$\sum_{n=1}^{N} \left[\frac{(1-\alpha)}{\mathcal{Z}_\mathcal{A}} \sum_{i \in \mathcal{Y}_n} \sum_{j \in \mathcal{S}_\mathcal{A}(i)} \log p\left(y_j | y_i\right) + \frac{\alpha}{\mathcal{Z}_\mathcal{N}} \sum_{i \in \mathcal{Y}_n} \sum_{\substack{k \in \mathcal{Y}_n \\ k \neq i}} \log p\left(y_k | y_i\right) \right] \tag{6}$$

where α determines the importance of each term ranging from 0 to 1, $\mathcal{Z}_\mathcal{A} = |\mathcal{Y}_n||\mathcal{S}_A(\cdot)|$ and $\mathcal{Z}_\mathcal{N} = |\mathcal{Y}_n|(|\mathcal{Y}_n| - 1)$. The probability of predicting an ancestor label j of a label i, i.e., $p(y_j|y_i)$, can be computed similarly to Eq. 3 by using *softmax* and slight modifications. Thus, the log-probability can be defined by

$$p(y_j|y_i) = \frac{\exp(\mathbf{u'}_j^T \mathbf{u}_i)}{\sum_{v \in L} \exp(\mathbf{u'}_v^T \mathbf{u}_i)} \tag{7}$$

where $\mathbf{u}_i$ is the i-th column vector of $\mathbf{U} \in \mathbb{R}^{d \times L}$ and $\mathbf{u'}_j$ is a vector representation for label j and the j-th column vector of $\mathbf{U'} \in \mathbb{R}^{d \times L}$. The softmax function in Eq. 7 can be viewed as an objective function of a neural network consisting of a linear activation function in the hidden layer and two weights $\{\mathbf{U}, \mathbf{U'}\}$, where $\mathbf{U}$ connects the input layer to the hidden layer while $\mathbf{U'}$ is used to convey

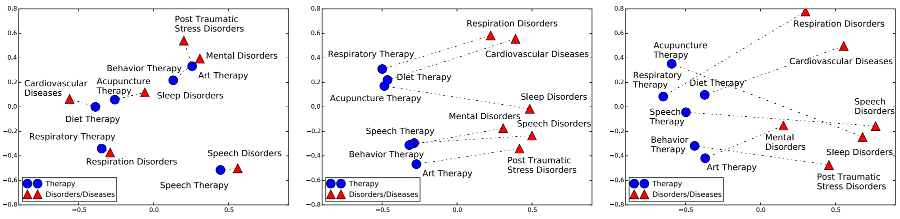


Fig. 2. Visualization of learned label embeddings by the log-linear model (Eq 6). (*left*) using only label co-occurrence patterns $\alpha = 1$ (*middle*) using a hierarchy as well as co-occurrences $\alpha = 0.5$ (*right*) using only a hierarchy $\alpha = 0$.

the hidden activations to the output layer. Here, $\mathbf{U}$ and $\mathbf{U}'$ correspond to vector representations for input labels and output labels, respectively. Like Eq. 3, we use *hierarchical softmax* instead of Eq. 7 to speed up pre-training label embeddings.

6.1 Understanding Label Embeddings

We begin by qualitatively demonstrating label embeddings trained on label co-occurrence patterns from the BioASQ dataset [1], which is one of the largest datasets for multi-label text classification, and label hierarchies in the 2015 MeSH vocabulary. The BioASQ dataset consists of more than 10 millions of documents. Note that we only use its label co-occurrence patterns. Its labels are also defined over the same MeSH vocabulary, so that we can use it for obtaining knowledge about the OHSUMED labels (cf Section 6). We trained the label embeddings using Eq. 6 by setting the dimensionality of the label embeddings to $d = \{128, 256, 512\}$ with different weighting values $\alpha = \{0, 0.5, 1\}$ for 100 epoch using SGD with a fixed learning rate of 0.1. If we set $d = 128$, training took about 6 hours on a machine with dual Xeon E5-2620 CPUs.

Analysis on learned label representations. Fig. 2 shows vector representations of labels related to Disorders/Diseases and their therapy in the 2015 MeSH vocabulary in 2D space.[2] It is likely that label pairs that co-occur frequently are close to each other. Particularly, on the *left* in Fig. 2, each therapy is close to a disorder for which the therapy is an effective treatment. If we make use of hierarchical information as well as co-occurrence label patterns during training, i.e., $\alpha = 0.5$ in Eq. 6, more interesting relationships are revealed which are not observed from the model trained only on co-occurrences ($\alpha = 1$). We can say that the learned vector representations has identified *Therapy-Disorders/Diseases* relationships (on the *middle* in Fig. 2). We also present label embeddings trained using only label hierarchies ($\alpha = 0$) on the *right* in Fig. 2.

Analogical reasoning in label spaces. One way to evaluate representation quality is analogical reasoning as shown in [21]. Upon the above observations (on

[2] Projection of 128-dim label embeddings into 2D was done by Principal Component Analysis.

Table 4. Analogical reasoning on learned vector representations of MeSH vocabulary

On learned representations *using* the hierarchy	
Analogy questions	Most probable answers
Cardiovascular Diseases : Diet Therapy ≈ Respiration Disorders : ?	Diet Therapy
	Enteral Nutrition
	Gastrointestinal Intubation
	Total Parenteral Nutrition
	Parenteral Nutrition
	Respiratory Therapy
Mental Disorders : Behavior Therapy ≈ PTSD : ?	Behavior Therapy
	Cognitive Therapy
	Rational-Emotive Psychotherapy
	Brief Psychotherapy
	Psychologic Desensitization
	Implosive Therapy

On learned representations *without using* the hierarchy	
Analogy questions	Most probable answers
Cardiovascular Diseases : Diet Therapy ≈ Respiration Disorders : ?	Respiration Disorders
	Respiratory Tract Diseases
	Respiratory Sounds
	Airway Obstruction
	Hypoventilation
	Croup
Mental Disorders : Behavior Therapy ≈ PTSD : ?	Behavior Therapy
	Psychologic Desensitization
	Internal-External Control
	PTSD
	Phobic Disorders
	Anger

the *middle* in Fig. 2), we performed analogical reasoning on both the representations trained with the hierarchy and ones without the hierarchy, specifically, regarding *Therapy-Disorders/Diseases* relationships (Table 4). As expected, it seems like the label representations trained with the hierarchy are clearly advantageous to the ones trained without the hierarchy on analogical reasoning. To be more specific, consider the first example, where we want to know what kinds of therapies are effective on "Respiration Disorders" as the relationship between "Diet Therapy" and "Cardiovascular Diseases". When we perform such analogical reasoning using learned embeddings with the hierarchy, the most probable answers to this analogy question are therapies that can be used to treat "Respiration Disorders" including nutritional therapies. Unlike the learned embeddings with the hierarchy, the label embeddings without the hierarchy perform poorly. In the bottom-right of Table 4, "Phobic Disorders" can be considered as a type of anxiety disorders that occur commonly together with "Post-traumatic Stress Disorders (PTSD)" rather than a treatment of it.

6.2 Results

The results on the modified zero-shot learning datasets in Table 5 show that we can obtain substantial improvements by the pretrained label embeddings. Please note that the scores obtained by using *random* label embeddings on the left in Table 5 are the same as those of *Wsabie* and $Wsabie_H$ in Table 3. In this experiment, we used very small base learning rates (i.e., $\eta_0 = 10^{-4}$ chosen by validation) for updating label embeddings in Eq. 4 after being initialized

Table 5. Initialization of label embeddings on OHSUMED under *zero-shot* settings.

	random label embeddings				***pretrained*** label embeddings			
	AvgP	RL	MiF	MaF	AvgP	RL	MiF	MaF
Wsabie	0.01	56.37	0.00	0.00	**1.64**	**2.82**	0.03	0.06
Wsabie$_H$	0.06	39.91	0.00	0.00	1.36	5.33	**0.08**	**0.14**

Table 6. Evaluation on the *full* test data of the OHSUMED dataset. Numbers in parentheses are standard deviation over 5 runs. Subscript P denotes the use of pretrained label embeddings.

	Wsabie	Wsaibe$_H$	Wsabie$_P$	Wsabie$_{HP}$
AvgP	45.72 (0.04)	45.76 (0.06)	45.88 (0.09)	**45.92** (0.02)
RL	4.09 (0.18)	3.72 (0.11)	3.44 (0.13)	**3.11** (0.10)
MiF	46.32 (0.04)	46.34 (0.04)	46.45 (0.07)	**46.50** (0.01)
MaF	13.93 (0.03)	13.96 (0.07)	14.19 (0.05)	**14.25** (0.02)

by the pretrained ones. This means that our proposed method is trained in a way that maps a document into the some point of label embeddings while the label embeddings hardly change. In fact, the pretrained label embeddings have interesting properties shown in Section 6.1, so that *Wsabie* starts learning at good initial parameter spaces. Interestingly, it was observed that some of the unseen labels are placed at the top of rankings for test instances, so that relatively higher scores of bipartition measures are obtained even for Wsabie. We also performed an experiment on the *full* OHSUMED dataset. The experimental results are given in Table 6. *Wsabie*$_{HP}$ combining pretrained label embeddings with hierarchical label structures is able to further improve, outperforming both extensions by its own across all measures.

7 Conclusions

We have presented a method that learns a joint space of instances and labels taking hierarchical structures of labels into account. This method is able to learn representations of labels, which are not presented during the training phase, by leveraging label hierarchies. We have also proposed a way of pretraining label embeddings from huge amounts of label patterns and hierarchical structures of labels.

We demonstrated the joint space learning method on two multi-label text corpora that have different types of label hierarchies. The empirical results showed that our approach can be used to place relevant unseen labels on the top of the ranked list of labels. In addition to the quantitative evaluation, we also analyzed label representations qualitatively via a 2D-visualization of label representations. This analysis showed that using hierarchical structures of labels allows us to assess vector representations of labels by analogical reasoning. Further studies should be carried out to make use of such regularities in label embeddings at testing time.

References

1. Balikas, G., Partalas, I., Ngomo, A.N., Krithara, A., Paliouras, G.: Results of the BioASQ track of the question answering lab at CLEF 2014. In: Working Notes for CLEF 2014 Conference, pp. 1181–1193 (2014)
2. Bengio, Y., Courville, A., Vincent, P.: Representation learning: A review and new perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence **35**(8), 1798–1828 (2013)
3. Bi, W., Kwok, J.T.: Multilabel classification on tree- and DAG-structured hierarchies. In: Proceedings of the 28th International Conference on Machine Learning, pp. 17–24 (2011)
4. Bi, W., Kwok, J.T.: Efficient multi-label classification with many labels. In: Proc. of the 30th International Conference on Machine Learning, pp. 405–413 (2013)
5. Cesa-Bianchi, N., Gentile, C., Zaniboni, L.: Incremental algorithms for hierarchical classification. Journal of Machine Learning Research **7**, 31–54 (2006)
6. Chekina, L., Gutfreund, D., Kontorovich, A., Rokach, L., Shapira, B.: Exploiting label dependencies for improved sample complexity. Machine Learning **91**(1), 1–42 (2013)
7. Chen, Y.N., Lin, H.T.: Feature-aware label space dimension reduction for multi-label classification. In: Advances in Neural Information Processing Systems, pp. 1529–1537 (2012)
8. Crammer, K., Singer, Y.: A family of additive online algorithms for category ranking. The Journal of Machine Learning Research **3**, 1025–1058 (2003)
9. Dembczyński, K., Waegeman, W., Cheng, W., Hüllermeier, E.: On label dependence and loss minimization in multi-label classification. Machine Learning **88**(1–2), 5–45 (2012)
10. Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. The Journal of Machine Learning Research **12**, 2121–2159 (2011)
11. Elisseeff, A., Weston, J.: A kernel method for multi-labelled classification. Advances in Neural Information Processing Systems **14**, 681–687 (2001)
12. Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., Lin, C.J.: LIBLINEAR: A library for large linear classification. The Journal of Machine Learning Research **9**, 1871–1874 (2008)
13. Frome, A., Corrado, G.S., Shlens, J., Bengio, S., Dean, J., Ranzato, M., Mikolov, T.: Devise: A deep visual-semantic embedding model. Advances in Neural Information Processing Systems **26**, 2121–2129 (2013)
14. Fürnkranz, J., Hüllermeier, E., Loza Mencía, E., Brinker, K.: Multilabel classification via calibrated label ranking. Machine Learning **73**(2), 133–153 (2008)
15. Fürnkranz, J., Sima, J.F.: On exploiting hierarchical label structure with pairwise classifiers. SIGKDD Explorations **12**(2), 21–25 (2010)
16. Hersh, W., Buckley, C., Leone, T.J., Hickam, D.: Ohsumed: an interactive retrieval evaluation and new large test collection for research. In: Proceedings of the 17th Annual International ACM SIGIR Conference, pp. 192–201 (1994)
17. Hsu, D., Kakade, S., Langford, J., Zhang, T.: Multi-label prediction via compressed sensing. In: Advances in Neural Information Processing Systems 22, vol. 22, pp. 772–780 (2009)
18. Lampert, C.H., Nickisch, H., Harmeling, S.: Attribute-based classification for zero-shot visual object categorization. IEEE Transactions on Pattern Analysis and Machine Intelligence **36**(3), 453–465 (2014)

19. Lewis, D.D., Yang, Y., Rose, T.G., Li, F.: RCV1: A new benchmark collection for text categorization research. The Journal of Machine Learning Research **5**, 361–397 (2004)
20. Loza Mencía, E., Fürnkranz, J.: Pairwise learning of multilabel classifications with perceptrons. In: Proceedings of the International Joint Conference on Neural Networks, pp. 2899–2906 (2008)
21. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. Advances in Neural Information Processing Systems **26**, 3111–3119 (2013)
22. Mnih, A., Hinton, G.E.: A scalable hierarchical distributed language model. Advances in Neural Information Processing Systems **22**, 1081–1088 (2009)
23. Morin, F., Bengio, Y.: Hierarchical probabilistic neural network language model. In: Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics, pp. 246–252 (2005)
24. Read, J., Pfahringer, B., Holmes, G., Frank, E.: Classifier chains for multi-label classification. Machine Learning **85**(3), 333–359 (2011)
25. Recht, B., Re, C., Wright, S., Niu, F.: Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In: Advances in Neural Information Processing Systems, pp. 693–701 (2011)
26. Rousu, J., Saunders, C., Szedmák, S., Shawe-Taylor, J.: Kernel-based learning of hierarchical multilabel classification models. Journal of Machine Learning Research **7**, 1601–1626 (2006)
27. Silla Jr, C.N., Freitas, A.A.: A survey of hierarchical classification across different application domains. Data Mining and Knowledge Discovery **22**(1–2), 31–72 (2011)
28. Socher, R., Ganjoo, M., Manning, C.D., Ng, A.: Zero-shot learning through cross-modal transfer. In: Advances in Neural Information Processing Systems, pp. 935–943 (2013)
29. Tai, F., Lin, H.T.: Multilabel classification with principal label space transformation. Neural Computation **24**(9), 2508–2542 (2012)
30. Vens, C., Struyf, J., Schietgat, L., Džeroski, S., Blockeel, H.: Decision trees for hierarchical multi-label classification. Machine Learning **73**(2), 185–214 (2008)
31. Weston, J., Bengio, S., Usunier, N.: Wsabie: scaling up to large vocabulary image annotation. In: Proceedings of the 22nd International Joint Conference on Artificial Intelligence, pp. 2764–2770 (2011)
32. Zimek, A., Buchwald, F., Frank, E., Kramer, S.: A study of hierarchical and flat classification of proteins. IEEE/ACM Transactions on Computational Biology and Bioinformatics **7**, 563–571 (2010)