Subset Labeled LDA: A Topic Model for Extreme Multi-label Classification

Yannis Papanikolaou(✉) and Grigorios Tsoumakas

School of Informatics, Aristotle University of Thessaloniki, Thessaloniki, Greece
{ypapanik,greg}@csd.auth.gr

Abstract. Labeled Latent Dirichlet Allocation (LLDA) is an extension of the standard unsupervised Latent Dirichlet Allocation (LDA) algorithm, to address multi-label learning tasks. Previous work has shown it to perform en par with other state-of-the-art multi-label methods. Nonetheless, with increasing number of labels LLDA encounters scalability issues. In this work, we introduce Subset LLDA, a topic model that extends the standard LLDA algorithm, that not only can efficiently scale up to problems with hundreds of thousands of labels but also improves over the LLDA state-of-the-art in terms of prediction accuracy. We conduct experiments on eight data sets, with labels ranging from hundreds to hundreds of thousands, comparing our proposed algorithm with the other LLDA algorithms (Prior–LDA, Dep–LDA), as well as the state-of-the-art in extreme multi-label classification. The results show a steady advantage of our method over the other LLDA algorithms and competitive results compared to the extreme multi-label classification algorithms.

Keywords: Machine learning · Extreme classification · Topic models
Multi-label classification

1 Introduction

Multi-label learning addresses supervised learning problems, where each training example is associated with more than one labels at the same time [12]. Examples include tasks such as image annotation or assigning concepts to documents. Although a great body of prior work has dealt with developing methods for multi-label tasks (e.g. [16]), the majority of these algorithms struggle to scale to data sets having more than a few thousand labels. The ever increasing flow and volumes of data in modern-day applications call for multi-label learning algorithms that can scale up effectively and efficiently.

Extreme multi-label classification (XMLC) is an emerging field that attempts to address the above challenge, by proposing algorithms that can tackle problems with extremely large label sets ($> 10^4$ labels).

We modify an already existing algorithm, Labeled Latent Dirichlet Allocation (LLDA) [8] to successfully deal with such tasks. LLDA was introduced as an extension of standard, unsupervised Latent Dirichlet Allocation (LDA) [2], to

C. Ordonez and L. Bellatreche (Eds.): DaWaK 2018, LNCS 11031, pp. 152–162, 2018.
https://doi.org/10.1007/978-3-319-98539-8_12

deal with multi-label learning tasks. Apart from delivering results competitive with state-of-the-art multi-label algorithms, LLDA's training is by design fit for large-scale and extreme learning problems, since its training time complexity is not dependent of the label set size L, but on the average number of labels assigned per instance. During testing though, LLDA reduces to LDA and the algorithm is linearly dependent of L, which makes it unfit for XMLC.

In order to make LLDA appropriate for such tasks, we propose an extension to LLDA, Subset LLDA. Our algorithm is different only during prediction. It first determines a set of relevant labels for each new instance, through its nearest neighbors in the training space, and then constrains the LLDA's inference on this particular subset of labels. By doing so, we manage to decrease the algorithm's testing time complexity and improve LLDA's quality of predictions, since by constraining the algorithm to search in a subset of the entire label space, we alleviate LLDA's tendency to converge to local optima (we describe this more in detail in Sect. 3).

We conduct experiments on four small scale data sets and four large scale data sets, with L ranging from 101 to 670,000 comparing our approach to Prior–LDA and Dep–LDA as well as two of the top performing extreme classification algorithms, Fast XML and PfastreXML. The results show a consistent advantage of our method compared to the other LLDA algorithms and competitive results with the extreme classification methods. Our motivation by introducing Subset LLDA, is to contribute one more method to the extreme classification inventory, that may be more apt than other methods for specific experimental scenarios.

2 Background and Related Work

We present here the main methods in the literature to tackle XMLC tasks and then present LDA, LLDA, and the other two LLDA extensions, Prior–LDA and Dep–LDA.Throughout the paper we assume that the Collapsed Gibbs Sampling (CGS) algorithm [4] is employed for all LLDA methods.

2.1 Extreme Classification Methods

Algorithms aiming to tackle extreme classification tasks mainly take one of the following approaches:

- learn a hierarchy out of the training set, either over the labels or the features, and solve the training and prediction procedures locally at each node. Examples in this category include Label Partitioning by Sublinear Ranking (LPSR) [13], FastXML [7] and PfastreXML [5].
- construct an embedding of the output space in a lower dimension. Embedding-based methods render training and prediction tractable by assuming that the training label matrix is low-rank, reducing the label set size by projecting the high dimensional label vectors onto a low dimensional linear subspace. Most characteristic methods in this category are LEML [15] and SLEEC [1].

2.2 LDA and LLDA

Let us denote as L the number of labels, l being a label and V the number of features, v being a feature type and w_i being a feature token at position i of the instance. M is the number of instances (M_{TRAIN} and M_{TEST} will represent the training and testing set sizes respectively), m being an instance. Also, L_m will denote the number of m's labels and N_m the number of its non-zero features.

LDA assumes that, given a set of instances, there exist two sets of distributions, the label-features distributions named $\boldsymbol{\phi}$ and the instances-labels distributions named $\boldsymbol{\theta}$[1]. CGS LDA marginalizes out $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$, and uses only the latent variable assignments $\mathbf{z}$. The algorithm employs two count matrices during sampling, the number of times that v is assigned to l across the data set, represented by n_{lv} and the number of feature tokens in m that have been assigned to l, represented by n_{ml}. During sampling, CGS updates the hard assignment z_i of w_i to one of $l \in \{1...L\}$. This update is performed sequentially for all tokens in the data set, for a fixed number of iterations. The update equation giving the probability of setting z_i to label l, conditional on w_i, m, the hyperparameters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, and the current label assignments of all other feature tokens (represented by $\cdot$) is:

$$p(z_i = l \,|\, w_i = v, m, \boldsymbol{\alpha}, \boldsymbol{\beta}, \cdot) \propto \frac{n_{lv\neg i} + \beta}{\sum\limits_{v'=1}^{V} (n_{lv'\neg i} + \beta)} \cdot \frac{n_{ml\neg i} + \alpha_l}{N_m + \sum\limits_{l'=1}^{L} \alpha_{l'}} \,. \tag{1}$$

Upon completion of the above procedure, point estimates can be calculated for $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ parameters. Instead of the standard CGS estimators [4], we employ the CGS^p equations [6], that employ the full distribution over feature tokens. These methods calculate the expected values of the standard CGS estimators and are therefore especially suited for the large-scale setting that we are addressing in this work, since they allow us to achieve better performance by drawing fewer samples than with the standard estimators.

LLDA modifies LDA by constraining the possible assignments for a token to a label to the instance's observed labels. Inference on test instances is performed similarly to unsupervised CGS - LDA: the label–features distributions, $\boldsymbol{\phi}$, are fixed to those previously learned on the training data, and the test instances' $\boldsymbol{\theta}$ distributions are estimated.

LLDA employs a symmetric α hyperparameter over labels, giving equal weight on them. Nevertheless, in most real-world tasks labels tend to have skewed distributions. Moreover, modeling label dependencies can improve performance [9], especially as the number of labels increases. To address these issues, the authors of [10] have proposed Prior–LDA and Dep–LDA respectively. Prior–LDA incorporates the label frequencies observed in the training set via an asymmetric

[1] Specifically, LDA is defining $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ in terms of topics since it is unsupervised, but to ease understanding we consider through the paper that topics and labels are equivalent.

α hyperparameter: a frequent label will have a larger α'_l value than a rare one. Specifically, it is set to

$$\alpha'_l = \eta \cdot f_l + \alpha \tag{2}$$

with η, α being user defined parameters and f_l representing the frequency of l in the training corpus, $f_l \in [0, 1]$.

Dep–LDA is a two-stage algorithm: first, an unsupervised LDA model is trained with T topics and using as training data, each instance's label set[2]. The estimated LDA model will incorporate information about the label dependencies, since relevant labels will tend to be described by the same topic(s). Second, an LLDA model is trained. During prediction, the previously estimated $\boldsymbol{\theta'}, \boldsymbol{\phi'}$ parameters of the unsupervised LDA model are used to calculate an asymmetrical α'_{ml}. Specifically, the $\boldsymbol{\alpha_m}'$ vector will be

$$\boldsymbol{\alpha'_m} = \eta(\boldsymbol{\theta'_m} \cdot \boldsymbol{\phi'}) + \alpha \tag{3}$$

3 Subset LLDA

When dealing with tasks with very large L ($> 10^4$), LLDA and its extensions can not scale in a satisfying manner since they are linearly dependent on L during prediction. To alleviate this, we propose a simple extension to standard LLDA during prediction, by constraining the label space in which the algorithm can search for solutions. Specifically, our method proceeds in two stages:

- First, for each test instance, a set of candidate labels, denoted as $\mathcal{L}_{m_{Rel}}$, is determined. A number of approaches can be followed to determine this candidate list, for simplicity we retrieve the n-nearest instances from the training set, denoted as $\mathcal{M}_{m_{Rel}}$, and set the candidate list as the union of the retrieved instances tags.
- Second, we predict with LLDA, but constrain the possible labels to $\mathcal{L}_{m_{Rel}}$. Similarly to Prior–LDA, we modify the prior on the instance-topics distributions to reflect the frequencies of the labels among the n-most similar instances. For instance, if $n = 10$ and a given label l_A has appeared in three of the similar instances, while a label l_B has appeared in five of the similar instances, we set $\alpha'_{l_A} = \eta + 0.3 \cdot \alpha, \alpha'_{l_B} = \eta + 0.5 \cdot \alpha$.

Formally, our topic model makes the following assumptions: First, given an instance m and given a set of already tagged instances $\mathcal{M}_{Train}$, m's label set $\mathcal{L}_m$ will be included in the union of the n most similar instances from $\mathcal{M}_{Train}$. Clearly, that assumption holds always as $n \to M_{Train}$ but in any other case it will represent an approximation and we expect that $\mathcal{L}_m \subseteq \mathcal{L}_{m_{Rel}}$. A second assumption relates to each label's weight during Gibbs sampling: we hypothesize that for a given instance m, its labels have been generated by sampling from a multinomial distribution ϕ'_m. This assumption is very similar to the one employed in [10], when introducing Prior–LDA, except that we constrain the set from which the instance's labels are generated to $\mathcal{L}_{m_{Rel}}$, instead of $\mathcal{L}$. Figure 1 illustrates our proposed model in graphical model notation.

[2] i.e., the feature tokens of each instance are its labels.

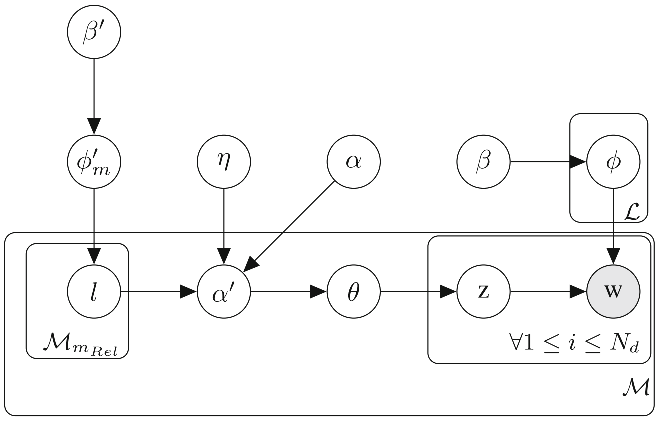

Fig. 1. Subset LLDA in graphical model notation. Our model assumes that, given an instance m, its labels have been generated by sampling from a multinomial distribution over m's similar instances.

From the above, the generative process for Subset LLDA will be:

– For each $l \in \mathcal{L}$, sample a distribution $\phi_l \sim$ Dirichlet(β) over $\mathcal{V}$
– for each instance m
  • Sample n instances from $\mathcal{M}_{Train}$
  • Set $\mathcal{L}_{m_{Rel}} = \bigcup \mathcal{L}_{m_i}$ with $m_i \in \mathcal{M}_{m_{Rel}}$ and $i \in \{1..n\}$
  • Sample a multinomial distribution $\phi'_m \sim$ Dirichlet(β')
  • Calculate α' according to Eq. 2.
  • Sample a distribution $\theta_m \sim$ Dirichlet(α') over $\mathcal{L}_{m_{Rel}}$
  • For each feature position $i \in \{1..N_m\}$
    ∗ Sample a label z_i ∼Multinomial(θ_m)
    ∗ Sample a feature type v_i ∼Multinomial(ϕ_l) with $l = z_i$.

Constraining the label set during prediction can also be useful for an additional reason. In that phase, LLDA needs to search the entire label space to recommend labels for a given test instance. Since this is a probabilistic method, the algorithm may converge to local optima, especially as L is increasing. To concretize, let us consider a trained LLDA model for which a specific feature v has a high probability ϕ_{lv} for several labels. Also, let us consider a test instance m that contains v, for which only one of the aforementioned labels is semantically relevant. In that case, it is possible that these noisy labels, coupled with LLDA's probabilistic nature, will lead the algorithm to favor one of the irrelevant labels at the expense of the correct label. This problem can of course be relieved by averaging over many samples and Markov Chains (MC) [6,10], but in most real cases this is too expensive time-wise.

Finally, we note that to retrieve the most similar training instances we use the tf-idf representation for each instance and employ the cosine similarity, setting $n = 10$.

Table 1. Statistics for the data sets used in the experiments.

Data set	D_{Train}	D_{Test}	L	V
Bibtex	4,880	2,515	159	1,836
Delicious	12,910	3,181	983	500
Mediamill	30,993	12,914	101	120
EUR-Lex	15,539	3,809	3,993	5,000
Wiki10	14,146	6,616	30,938	101,938
BioASQ	40,000	10,000	19,218	36,480
Delicious-200k	196,606	100,095	205,443	782,585
Amazon-670k	490,449	153,025	670,091	135,909

3.1 Time Complexity

The CGS algorithm for LDA proceeds as follows: for every feature token of every instance in the corpus, it calculates probability distribution over all labels and then samples a label for the token, out of this calculated probability. In this way, standard LDA is linearly dependent on L. Formally, it will be

$$T_{LDA} \propto \mathcal{O}(M \cdot \overline{V_m} \cdot L). \tag{4}$$

LLDA, introduces supervision during training, by constraining the possible labels that a feature token can get on the instance's label set L_d. Formally, during training LLDA's complexity will be

$$T_{LLDA} \propto \mathcal{O}(M_{TRAIN} \cdot \overline{V_m} \cdot \overline{L_m}). \tag{5}$$

As explained, during testing LLDA is equivalent to LDA so its complexity will be given by Eq. 4. To alleviate this, with our approach we constrain Subset LLDA during testing, to only consider labels from the n-most similar instances. The total complexity of Subset LLDA, involves also finding the n-most similar instances which in our cases will be $\mathcal{O}(M_{TEST} \cdot M_{TRAIN})$:

$$T_{SubsetLLDA} \propto \mathcal{O}(M_{TEST} \cdot \overline{V_m} \cdot \nu \cdot \overline{L_m} + M_{TEST} \cdot M_{TRAIN}). \tag{6}$$

4 Empirical Evaluation

We here present the data sets, the setup and the results of the experiments that we carried out. We compare Subset LLDA with Prior–LDA, Dep–LDA, Fast XML and PfastreXML.

In our experiments, we employed four small scale and four large scale data sets, their statistics being illustrated in Table 1. The motivation behind using the four small scale data sets is to be able to compare our algorithm with the other LLDA variants, as well as to provide an empirical comparison against

Table 2. Micro-F and Macro-F results for the LLDA-based and the extreme classification methods. A ▽ indicates a statistically significant difference between Subset LLDA and the best performing method, with a z-test and a significance level of 0.05. The - sign is used if the algorithm could not deliver predictions after 48 hours.

(a) Micro-F

	FastXML	PfastreXML	PriorLDA	DepLDA	Subset LLDA
Bibtex	0.382	**0.397**	0.363	0.314	0.384
Delicious	**0.347**▽	0.322	0.255	0.273	0.304
Mediamill	0.577	0.572	0.511	0.512	**0.580**
EUR-Lex	0.413	0.436	0.321	0.357	**0.443**▽
BioASQ	0.382	0.370	-	-	**0.428**▽
Wiki10	0.317	**0.33**▽	-	-	0.283
Delicious-200k	**0.162**▽	0.129	-	-	0.135
Amazon	0.192	**0.316**▽	-	-	0.243
Avg. Rank	2.125	2.000			1.875

(b) Macro-F

	FastXML	PfastreXML	PriorLDA	DepLDA	Subset LLDA
Bibtex	0.273	0.288	0.257	0.204	**0.292**
Delicious	0.153	**0.165**▽	0.100	0.090	0.136
Mediamill	0.070	**0.101**▽	0.027	0.027	0.068
EUR-Lex	0.427	0.416	0.305	0.336	**0.444**
BioASQ	0.392	0.399	-	-	**0.451**▽
Wiki10	**0.305**▽	0.285	-	-	0.272
Delicious-200k	**0.105**▽	0.067	-	-	0.078
Amazon	0.426	0.483	-	-	**0.486**
Avg. Rank	2.125	2.000			1.875

the other extreme classification methods, in standard multi-label classification settings. All data sets apart from *BioASQ*[3] [11] were retrieved from the extreme classification repository, with the respective training/testing splits.

Fast XML and PfastreXML were used with default parameters, with the relevant software packages provided in the extreme classification repository. For the LLDA models, we provide our Java implementation[4]. We trained the same model for all algorithms in order to ensure fairness of comparison, with $\alpha_l = \frac{50}{L}$. For Dep–LDA, we additionally need to train an LDA model to calculate the hyperparameter on θ (ref. to Eq. 3). For its training we use 100 topics and 200 iterations, 50 burn-in iterations and we set $\alpha = 0.1$, $\beta = 0.01$. During prediction, we set $\eta = 50$, $\alpha = \frac{30}{L}$, $\beta = 0.01$ for all LLDA models. In both training and prediction and across all data sets, we used one Markov Chain with 200 iterations and 50 iterations burn-in, averaging across samples to obtain the respective parameter estimates for each method. All algorithms output rankings of relevant

[3] The exact data set used in the experiments is available upon request to the authors.
[4] https://www.dropbox.com/s/rypmlt18zk6jxdh/sllda.zip?dl=0.

Table 3. Precision@1 and Precision@5 results for the LLDA-based and the extreme classification methods.

(a) Precision@1

	FastXML	PfastreXML	PriorLDA	DepLDA	Subset LLDA
Bibtex	**0.645**▽	0.565	0.551	0.501	0.579
Delicious	**0.705**▽	0.546	0.488	0.526	0.525
Mediamill	**0.873**▽	0.867	0.794	0.794	0.813
EUR-Lex	0.637	0.644	0.622	0.631	**0.663**▽
BioASQ	0.176	**0.189**▽	-	-	0.077
Wiki10	**0.825**▽	0.757	-	-	0.773
Delicious-200k	**0.432**▽	0.261	-	-	0.163
Amazon	0.286	**0.368**▽	-	-	0.350
Avg. Rank	1.625	2.000			2.375

(b) Precision@5

	FastXML	PfastreXML	PriorLDA	DepLDA	Subset LLDA
Bibtex	**0.286**▽	0.254	0.238	0.215	0.243
Delicious	**0.596**▽	0.482	0.391	0.416	0.427
Mediamill	0.531	0.534	0.480	0.471	**0.539**
EUR-Lex	0.425	0.445	0.332	0.358	**0.457**▽
BioASQ	0.170	0.169	-	-	**0.174**
Wiki10	0.570	**0.572**	-	-	0.560
Delicious-200k	**0.362**▽	0.262	-	-	0.201
Amazon	0.195	**0.320**▽	-	-	0.246
Avg. Rank	2.000	1.875			2.125

labels for each instance, so we used the *rcut* thresholding strategy [14] to compute the Micro-F and Macro-F scores.

4.1 Results

In Tables 2, 3 and 4 we report the results of our experiments. For the LLDA methods, we report the average over five runs.

First, let us consider the differences in prediction accuracy among the LLDA methods. Subset LLDA steadily outperforms both Prior–LDA and Dep–LDA in all settings and for all measures. It should be noted here, that Dep–LDA by design would benefit by averaging over more samples and more than one MC more than the other methods, since it employs the parameters learned from an unsupervised LDA model to calculate the hyperparameters for θ, therefore it will be more prone than the other algorithms to get stuck in local optima. In other words, the LDA model introduces an additional factor of uncertainty, which could be alleviated with more samples and chains. We nevertheless restrict our experiments to only one MC and relatively few samples (thirty) since our main goal is to address large-scale tasks in which multiple MC averaging and averaging over many samples is not feasible. One more interesting observation is that for tasks with few labels (*Bibtex*, *Mediamill*), Dep–LDA performs equally or worse to Prior LDA which may be explained by the fact that modeling dependencies does not necessarily help improving performance in small scale tasks.

Comparing Subset LLDA with the extreme classification methods, we observe that different algorithms fare well in different evaluation measures. By considering the average rank per evaluation measure (last row of the tables), we observe that Subset LLDA achieves the first place for two measures, Micro-F and Macro-F, PfastreXML for precision@5 and FastXML for precision@1. FastXMl and PfastreXML are better in predicting a few relevant labels per instance, while our algorithm is better in balancing precision and recall (through the F-measure) both when treating all labels equally (Macro-F) and when weighting them by their frequency (Micro-F).

Table 4. Training and prediction duration, in seconds, for the LLDA-based and the extreme classification methods.

	FastXML	PfastreXML	PriorLDA	DepLDA	Subset LLDA
Bibtex	**3+1**	**3+1**	29+73	32+64	29+22
Delicious	**6+3**	7+5	133+177	77+183	77+15
Mediamill	**54+3**	56+3	335+238	306+286	306+101
EUR-Lex	**493+62**	514+72	941+22,492	900+22,761	900+214
BioASQ	**571+428**	623+561	-	-	1,380+553
Wiki10	1,651+251	1,692+262	-	-	**1,131+192**
Delicious-200k	**20,444**+5,760	20,578+5,891	-	-	36,231+**4,080**
Amazon	**14,931**+984	15,044+1,194	-	-	38,510+**221**

Results vary greatly with respect to data sets too. For the small scale data sets, Subset LLDA achieves the best result in 7 out of 16 cases (four data sets times four evaluation measures), FastXML in 6 and PfastreXML in 3. For the large scale data sets, both FastXML and PfastreXML achieves the best result in 6 out of 16 cases, while Subset LLDA in 4. These small differences among the methods, suggest that there exists no one-size-fits-all algorithm for extreme learning tasks and each problem should be treated separately.

In Table 4, we additionally report the training and testing duration for each of the algorithms and data sets. The results show two clear tendencies, with Subset LLDA being significantly faster than the two other LLDA variants, while being slower across the majority of the data sets compared to FastXML and PfastreXML. We should note though, that our implementation could be further optimized by using much faster LDA implementations [3] and that both the extreme classification methods as well as our method can improve substantially by averaging over more trees or samples so a more detailed analysis should be conducted to assess the trade-off between duration and performance for each of the algorithms.

5 Conclusions and Future Work

In this work we have presented an extension of LLDA, to account for large-scale and XMLC tasks. Our algorithm Subset LLDA, proceeds in two stages, by first determining a set of relevant labels for a given test instance, and then constraining the CGS-LLDA algorithm to search only this label subspace for a solution. Experiments on eight data sets, with label sets sizes ranging from hundreds to hundreds of thousands, show a significant improvement over the best performing LLDA-based algorithms and competitive results with the state-of-the-art in extreme classification and suggest that Subset LLDA should be considered as a competitive alternative when dealing with XMLC tasks.

References

1. Bhatia, K., Jain, H., Kar, P., Varma, M., Jain, P.: Sparse local embeddings for extreme multi-label classification. In: Advances in Neural Information Processing Systems, pp. 730–738 (2015)
2. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet allocation. J. Mach. Learn. Res. **3**, 993–1022 (2003)
3. Chen, J., Li, K., Zhu, J., Chen, W.: WarpLDA: a cache efficient o (1) algorithm for latent Dirichlet allocation. Proc. VLDB Endowment **9**(10), 744–755 (2016)
4. Griffiths, T.L., Steyvers, M.: Finding scientific topics. Proc. Natl. Acad. Sci. **101**(Suppl. 1), 5228–5235 (2004)
5. Jain, H., Prabhu, Y., Varma, M.: Extreme multi-label loss functions for recommendation, tagging, ranking & other missing label applications. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 935–944. ACM (2016)
6. Papanikolaou, Y., Foulds, J.R., Rubin, T.N., Tsoumakas, G.: Dense distributions from sparse samples: improved gibbs sampling parameter estimators for LDA. J. Mach. Learn. Res. **18**(62), 1–58 (2017)
7. Prabhu, Y., Varma, M.: FastXML: a fast, accurate and stable tree-classifier for extreme multi-label learning. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 263–272. ACM (2014)
8. Ramage, D., Hall, D., Nallapati, R., Manning, C.D.: Labeled LDA: a supervised topic model for credit attribution in multi-labeled corpora. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, EMNLP 2009, vol. 1, pp. 248–256. Association for Computational Linguistics, Stroudsburg (2009)
9. Read, J., Pfahringer, B., Holmes, G., Frank, E.: Classifier chains for multi-label classification. In: Proceedings 20th European Conference on Machine Learning (ECML 2009), pp. 254–269 (2009)
10. Rubin, T.N., Chambers, A., Smyth, P., Steyvers, M.: Statistical topic models for multi-label document classification. Mach. Learn. **88**(1–2), 157–208 (2012)
11. Tsatsaronis, G., et al.: An overview of the BIOASQ large-scale biomedical semantic indexing and question answering competition. BMC Bioinformatics **16**, 138 (2015)
12. Tsoumakas, G., Katakis, I.: Multi-label classification: an overview. Int. J. Data Warehouse. Min. **3**(3), 1–13 (2007)

13. Weston, J., Makadia, A., Yee, H.: Label partitioning for sublinear ranking. In: Proceedings of the 30th International Conference on Machine Learning (ICML-13), pp. 181–189 (2013)
14. Yang, Y.: A study of thresholding strategies for text categorization. In: SIGIR 2001: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 137–145. ACM, New York (2001)
15. Yu, H.F., Jain, P., Kar, P., Dhillon, I.: Large-scale multi-label learning with missing labels. In: International Conference on Machine Learning, pp. 593–601 (2014)
16. Zhang, M.L., Zhou, Z.H.: A review on multi-label learning algorithms. IEEE Trans. Knowl. Data Eng. **99**(PrePrints), 1 (2013)