Multi-Label Learning with Millions of Labels: Recommending Advertiser Bid Phrases for Web Pages

Rahul Agrawal
Microsoft AdCenter

Archit Gupta
IIT Delhi

Yashoteja Prabhu
Microsoft Research

Manik Varma†
Microsoft Research

ABSTRACT

Recommending phrases from web pages for advertisers to bid on against search engine queries is an important research problem with direct commercial impact. Most approaches have found it infeasible to determine the relevance of all possible queries to a given ad landing page and have focussed on making recommendations from a small set of phrases extracted (and expanded) from the page using NLP and ranking based techniques. In this paper, we eschew this paradigm, and demonstrate that it is possible to efficiently predict the relevant subset of queries from a large set of monetizable ones by posing the problem as a multi-label learning task with each query being represented by a separate label.

We develop Multi-label Random Forests to tackle problems with millions of labels. Our proposed classifier has prediction costs that are logarithmic in the number of labels and can make predictions in a few milliseconds using 10 Gb of RAM. We demonstrate that it is possible to generate training data for our classifier automatically from click logs without any human annotation or intervention. We train our classifier on tens of millions of labels, features and training points in less than two days on a thousand node cluster. We develop a sparse semi-supervised multi-label learning formulation to deal with training set biases and noisy labels harvested automatically from the click logs. This formulation is used to infer a belief in the state of each label for each training ad and the random forest classifier is extended to train on these beliefs rather than the given labels. Experiments reveal significant gains over ranking and NLP based techniques on a large test set of 5 million ads using multiple metrics.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms, Performance, Experimentations

Keywords

Multi-Label Learning, Semi-Supervised Learning, Random Forests, Large Scale Learning, Bid Phrase Recommendation

†E-mail: manik@microsoft.com

WWW'13, May 13-17, 2013, Rio de Janeiro, Brazil
ACM 978-1-4503-2035-1/13/05.

1. INTRODUCTION

Our objective is to design an algorithm for automatically recommending bid phrases to an advertiser from a given ad landing page. The advertiser can choose to bid on specific recommendations so as to maximize the ad's chances of getting triggered in response to a relevant search engine query. This is an important research problem from a scientific as well as a commercial perspective as it facilitates the automation of large scale campaign creation and enables both premium and small advertisers to make more informed choices about their bid phrases.

A natural way of formulating the problem, from a machine learning perspective, would be to pose it as a supervised multi-label learning task with each query that could potentially be asked at a search engine corresponding to a separate label. The objective would be to learn a multi-label classifier which could predict the most relevant *set* of labels (queries) in response to a given ad landing page. However, such an approach has traditionally been considered to be infeasible due to four primary reasons. First, supervised learning techniques typically deal with problems involving hundreds to thousands of labels and it is not clear how to formulate problems involving an infinite number of labels. Second, obtaining good quality annotated training data would be a daunting task at this scale. Third, training such a high capacity classifier in a reasonable amount of time might require significant resources. Fourth, it might be difficult to ensure that such a complex classifier could process a novel ad landing page and make predictions in a few milliseconds as required by most applications.

In order to avoid these stiff challenges, state-of-the-art techniques for bid phrase recommendation take a very different approach based on ranking [32, 46]. They typically learn a ranker, or binary classifier, which takes both the ad landing page and a potential bid phrase as input and predicts whether the potential bid phrase is relevant to the given ad landing page or not. The classifier is dependent on joint ad-phrase features and can therefore only be applied to phrases present on the ad landing page. In addition, a query language model is used to ensure that phrases selected by the classifier are well formed, potentially monetizable, queries.

The ranking based paradigm sidesteps the challenges faced by the multi-label learning approach. In particular, one never needs to deal with an infinite label space but only make binary predictions regarding whether a phrase is a relevant, well-formed query or not. Furthermore, getting annotated training data is straightforward and linear binary classifiers can be trained very efficiently. Prediction costs are also low

as the classifier is limited to testing only the phrases present on the given ad landing page.

However, the ranking paradigm also suffers from two shortcomings. The inability of the classifier to suggest bid phrases not present on the page is a severe limitation since many ad landing pages have very little text on them. It would appear that brevity is the soul of advertising. It would also appear that a picture is worth a thousand words as many premium advertisers prefer conveying their message visually (including converting the text on their ad landing pages into embedded images so as to make sure they render well even when the user does not have the proper fonts installed). In a large scale evaluation, we found the ranking based technique of [46] was unable to recommend even ten bid phrases for millions of ad landing pages. One could try to expand the set of recommended bid phrases using query expansion or monolingual machine translation techniques [32]. However, such techniques were not found to be very helpful in situations where the seed set being expanded was small. One could also potentially tackle this problem by using document expansion techniques, or by bringing in meta-stream information from the logs, but such approaches are typically avoided as they increase the cost of prediction.

The second major limitation is due to the use of a single, low capacity, binary classifier which cannot model the rich relationship between billions of bid phrases and ad landing pages. As a result, even when there is enough text present on the ad landing page for the binary classifier to make predictions, the quality of the recommendations can sometimes be poor. Both limitations are highlighted in Figure 1.

In this paper, we demonstrate that it is possible to tractably formulate bid phrase recommendation as a supervised multi-label learning task. While the number of queries that could potentially be asked at a search engine is infinite, the number of queries that generate the vast majority of revenue is around ten million. We therefore treat each of these ten million queries as a separate label in our multi-label classifier. The advantage of our formulation is that it naturally allows recommending queries not present on the given ad landing page. Furthermore, by learning a high capacity classifier in a large output space, we hope that we can have a much richer model for capturing the relationship between ad landing pages and relevant queries.

Our contributions are as follows: We pose bid phrase recommendation as a multi-label learning problem with ten million labels. We design a Multi-Label Random Forest (MLRF) classifier whose prediction costs are logarithmic in the number of labels and which can make predictions in a few milliseconds using 10 GB of RAM. We develop efficient algorithms for training our classifier on 90 million training points, 10 million labels and 6 million features in less than a day on a cluster of a thousand nodes. We harvest training data automatically from our click logs and require no manual annotation. We develop a novel sparse semi-supervised multi-label learning formulation to mitigate the effect of training set biases which creep in due to automatic training set generation. Experiments on a large test set of 5 million ads reveal that our MLRF classifier can make at least a hundred recommendations on almost all ad landing pages and that these recommendations are significantly better than those made by ranking based techniques.

The rest of the paper is organized as follows. We discuss related work in Section 2 and review bid phrase recommendation approaches as well as techniques for multi-label learning with a large number of labels. We then develop our multi-label formulation in Section 3. We discuss how to automatically generate training data for our Multi-Label Random Forest classifier and show how it can be trained efficiently and used for making predictions in a few milliseconds. We develop a sparse semi-supervised multi-label learning formulation in Section 4 to mitigate the effects of biases introduced in automatic training set generation. We compare our multi-label approach to ranking based techniques in Section 5 and draw conclusions in Section 6.

2. RELATED WORK

Bid phrase recommendation is an important research problem with direct commercial impact. Many approaches assume that the advertiser has provided a seed set of bid phrases from the ad landing page and focus on expanding the seed set [1, 2, 10, 11, 23, 28, 29]. Recent approaches have argued for extracting bid phrases from the ad landing page directly [14,32,44,46] as premium advertisers can then avoid the onerous task of providing seed sets for the thousands of landing pages in a large campaign. Small advertisers, who might not be familiar with the nuances of online advertising, also benefit from not having to specify seed sets since their choices can sometimes include irrelevant bid phrases and are sometimes too small for expansion algorithms to give good results.

The KEX system of [46] is a ranking based approach which learns a binary logistic regression classifier to predict whether each phrase present on the given ad landing page is a relevant, well-formed query or not. The classifier is trained on two types of features. Joint phrase-document features, such as how often does the candidate phrase appear on the ad landing page, does the candidate phrase appear in the title, *etc.* are helpful in determining the relevance of the candidate phrase to the ad landing page. Note that these features evaluate to null if the candidate phrase does not occur on the ad landing page. Phrase only features, such as is the candidate phrase a short noun-phrase, how much revenue did the candidate phrase generate in the recent past, *etc.* are helpful in determining whether the candidate phrase is a well formed, potentially monetizable query. All the candidate phrases present on the ad landing page are ranked according to their probability as estimated by the logistic regression classifier. As discussed, being restricted to recommending only those phrases that are present on the ad landing page and using a low capacity binary classifier are limitations which we hope to overcome using our multi-label learning formulation.

An alternative approach to recommending phrases not present on the ad landing page is to expand the original suggestions using query expansion or monolingual translation techniques [32]. However, we did not find such techniques to be helpful in situations where the original seed set being expanded was very small. Note that the topic of bid phrase and query expansion is, in a sense, orthogonal to our work since even our recommendations can be expanded using existing techniques.

Finally, the ideas developed in [14, 44] cover related work but are not directly relevant to this paper as [44] focusses on augmenting KEX with EBay specific features while [14] studies the advanced matching of bid phrases to retrieve relevant ads.

The objective in multi-label learning is to predict a *set* of relevant labels for a given data point. This is in contrast to multi-class classification which aims to predict only a single label. The problem of multi-label classification with more labels than samples was studied from a theoretical perspective in [17]. They assume the existence of an efficient base classifier, such as our proposed random forest, and focus their analysis on a procedure to prune away labels from the label space so as to improve the base classifier's accuracy. Their analysis is quite interesting but orthogonal to many of the base classifier issues that we address in this paper. Almost all the other work on large category multi-label classification deals with thousands of labels [21,24,41,42] and it is not immediately clear that these techniques will scale to millions of labels and beyond.

Random forests have been studied extensively for binary and multi-class classification [9]. Multi-label decision trees and random forests have also been well studied [6,15,25,26, 34,42,45]. However, none of these methods scales to millions of labels.

Hierarchies can make it easier to learn from a large number of labels [5, 12, 33]. While hierarchies can be obtained for some applications from Wikipedia or WordNet, this is not the case for us. Recent innovative work has focussed on learning hierarchies [3, 4, 18] for multi-class problems with thousands of categories. We compare our method to such approaches in Section 3.

3. LEARNING WITH MILLIONS OF LABELS

The objective in multi-label learning is to predict a *set* of relevant labels for a given data point. This is in contrast to multi-class classification which aims to predict only a single label. In this Section, we describe our approach for harvesting training data automatically, developing a classifier with prediction costs that are logarithmic in the number of labels and training such a classifier in a reasonable amount of time.

3.1 Training Data

In order to generate our training data set, we downloaded 90 million ad landing pages off the Web and converted them to a simple bag-of-words feature representation. Our vocabulary contains approximately 6 million words and was generated by taking all the words present in the 90 million ad landing pages and removing stop words and words which occurred in less than ten pages. The representation of each ad landing page is therefore a sparse 6 million dimensional feature vector containing TF-IDF values for each word computed from the ad landing page's raw HTML. Words occurring in the title and anchor text have their counts boosted by 10 due to their significance. Other features, such as bigrams, or features derived from the ad copy or from the campaign, or even image based computer vision features or category level information about the ad landing page could be incorporated if desired.

Generating good quality annotations is a challenge. Training labels are typically obtained through an editorial process where human experts annotate training data points with relevant labels. However, at our scale, it is impossible for a human to sift through ten million potential candidates to determine the exact set of relevant labels for a given data point (note that this is not so much of a problem in the multi-class setting since selecting a single label is still relatively straightforward). Thus, all training, validation and test data points are likely to suffer from missing labels. It is therefore infeasible to ask an editorial team to annotate 90 million ad landing pages when even a single ad landing page cannot be properly annotated.

We therefore turn to automated procedures for harvesting training labels. In particular, we mine the click logs over a finite time window so as to determine the set of queries that resulted in a click on a given ad landing page. These queries are then treated as the set of positive labels for that training page. We discarded all queries that did not generate at least five clicks in total. Thus, the training labels for each ad landing page are the subset of the 10 million queries which resulted in a click on that ad over a given time window.

Note that this procedure introduces many biases into the training set. Missing and incorrect labels abound in particular. Incorrect labels occur when users click on ads not relevant to their query or due to intent changes. Such labels can be potentially detected and discarded by human experts but doing so is costly at our scale. Missing labels occur since not all the queries relevant to an ad are likely to be observed, or generate enough clicks to pass our threshold, during a finite data gathering time period. As discussed, missing labels are impossible to correct manually since no human expert can go through a list of 10 million labels and mark the exact relevant subset. Note that irrelevant and missing labels would occur even in alternative data gathering strategies including training on conversion data or currently selected bid phrases. Other biases creep in due to the ad serving and click position bias, the fact that we inferred only a subset of the positive labels for a given ad landing page and did not infer the set of negative labels (irrelevant queries) for that page. We mitigate the effect of some of these biases by developing a sparse semi-supervised multi-label learning formulation in Section 4.

Another source of training set bias is the skew in the distribution of search engine queries. Due to the heavy tail, most labels (queries) have at most a handful of positive training examples while a few labels dominate with many training data points. State-of-the-art discriminative classifiers such as M3L [20] and MetaLabeler [38] would have, for most labels, the daunting task of learning from a data set where the ratio of positive to negative data points is 1 is to 10 million. This problem is slightly mitigated for our proposed MLRF classifier since it learns from only positive data and avoids the suboptimal approach of treating missing and unobserved labels as negative.

3.2 Logarithmic Prediction using Multi-Label Random Forests

We seek to develop a classifier that can make predictions in a few milliseconds using 10 Gb of RAM. Almost all the multi-label classifiers that have been proposed in the literature have prediction costs that are at least linear in the number of labels [8,19–22,24,38–42,47]. However, even linear prediction costs are infeasible at our scale. For instance, with 10 million labels, the SVM based M3L classifier [20] would take hours to make a single prediction and would require terabytes of RAM for storing the model parameters.

A few approaches have tried to alleviate this problem by compressing the feature and label spaces [21, 24, 43]. If one could compress the 10 million dimensional label space and the 6 million dimensional feature space to a thousand dimensional common subspace then prediction could be carried out

very efficiently. Unfortunately, compression tends to lose information. Distinct queries such as "car", "motor vehicle" and "auto" might get compressed to the same label. This is undesirable since "car" and "auto" generate more revenue than "motor vehicle" and should be ranked higher when being recommended to advertisers. Thus, we need to make fine grained distinctions between queries with the same semantic meaning, queries with spelling variations and mistakes, queries with different word orderings, *etc.* since they have different monetization potentials. Compression, by itself, is therefore undesirable. A post-processing step disambiguating the compressed queries might help but would increase prediction costs.

Another approach to speeding up prediction would be to learn a hierarchy or tree in label space. Recent work has developed techniques for multi-class problems involving many thousands of categories [3, 4, 18]. The objective is to grow a tree by partitioning a parent's label space between its children so that each individual child has to deal with far fewer categories than the parent. The tree is grown until there is a single category left at each leaf node and predictions can therefore be made in time that is logarithmic in the total number of categories. However, learning trees in label space is a hard problem and none of the label-tree algorithms have been tried out on multi-label problems with ten million labels or more.

We propose a much simpler solution based on learning a gating tree in feature space rather than learning a hierarchy in label space. Our objective is to grow a tree by partitioning a parent's feature space between its children so that each individual child has to deal with far fewer training data points than the parent. The hope is that since the number of labels is almost the same as the number of training points, partitioning the feature space will also lead to a reduction in the number of labels that each individual child has to deal with. It is far simpler to grow such a gating tree in feature space than learn a hierarchy in label space and algorithms for growing trees in feature space, such as decision trees and random forests, have been around for many decades.

Our gating tree is grown until the number of training points in a leaf node drops below a threshold – typically chosen so as to ensure that the number of labels left in the leaf node is logarithmic in the total number of labels. When recommendations are sought for a new ad landing page, the page is passed down the gating tree until it reaches a leaf node. One can now use a multi-label classifier of choice, such as a multi-label SVM, by restricting it to the small set of labels present at the leaf node.

However, using an SVM as the leaf node classifier in the gating tree would still be relatively expensive during prediction (as well as training). We therefore use a very simple classifier based on randomization which turns our gating trees into multi-label random forests. We learn an ensemble of randomized gating trees rather than a single tree. At prediction time, a novel ad landing page is passed down each tree to reach a leaf node with a different sparse distribution over labels. The distributions can be aggregated over leaf nodes and the most popular labels can be recommended to the advertiser.

Predictions using our multi-label random forest can be carried out very efficiently. The cost of traversing each tree is logarithmic in the total number of training points which is almost the same as being logarithmic in the total number of labels. The cost of aggregating and sorting the leaf node distributions is also logarithmic in the total number of labels. As a result, recommendations for a novel ad landing page can be made in a few milliseconds and storing the trees takes less than 10 Gb of RAM.

3.3 Efficiently Training Multi-Label Random Forests

We briefly review binary and multi-class random forests before discussing how multi-label random forests might be trained at scale.

3.3.1 Random Forests

A random forest is an ensemble of decision trees, each of which is trained on a randomly drawn sample of data points and features. Trees are grown by splitting leaf nodes into a left and right child and partitioning the node's data points between the two children based on a splitting condition. Splitting conditions (typically) compare a point's selected feature value to a threshold to determine whether the point should be passed on to the left or right child. The particular feature to select and the value of the threshold are determined by optimizing a cost function such as the class entropy or the Gini index. In more detail, one selects a random set of features as well as a set of thresholds for these features, and then chooses the single best feature and threshold that result in minimal Gini index for the two children. This process is repeated until all the trees are fully grown or no further reduction in the Gini index is possible. There exist many variants for each of the stages of random forest training and some are covered in [9]. Predictions are made by passing a test point through all the trees and then aggregating the class distributions of the leaf nodes containing the test point.

3.3.2 Multi-Label Random Forests

We need to define and optimize an appropriate node splitting cost function in order to extend random forests to efficiently handle multi-label problems with millions of labels. The label entropy and the Gini index of the child nodes continue to be attractive criteria since they ensure that the distribution of labels in each individual child node is as compact as possible. They also ensure that points within a node are similar to each other and distinct from points in the sibling node. However, for multi-label learning, the space over which these compactness measures need to be calculated is the power set of labels rather than the labels themselves. Thus, we need to define probability measures over sets and compute them efficiently. While one can come up with mathematically elegant definitions for such probabilities, we are thwarted by the fact that the power set is exponentially large in the number of labels. Calculating the Gini index over the power set is therefore both computationally and statistically inefficient.

Assuming that the labels are independent would bring down the computational cost from exponential to not just linear but to logarithmic. For instance, if the labels were assumed to be independent, the expression for the Gini index

of a node would simplify to

$$G = \sum_{\mathbf{k} \in \{0,1\}^K} p(\mathbf{l} = \mathbf{k})(1 - p(\mathbf{l} = \mathbf{k})) \quad (1)$$

$$= 1 - \sum_{\mathbf{k} \in \{0,1\}^K} \left(\prod_{k=1}^{K} p(l_k = k_k) \right)^2 \quad (2)$$

$$= 1 - \prod_{k=1}^{K} \left(p^2(l_k = 1) + (1 - p(l_k = 1))^2 \right) \quad (3)$$

where $K = 10$ million is the total number of labels, $p(l_k = 1) = N_k/N$ is the probability of observing label k as a positive label in the node, N_k is the number of points in the node which have label k as a positive label and N is the total number of points in the node. Note that $p(l_k = 1) = 0$ for labels not present in the node and these do not contribute to the product. The expression for the Gini index therefore depends only on the set of positive labels present in the node. Since an ad landing page has only $O(\log K)$ positive labels on average, the Gini index G can be computed very efficiently as one goes down the tree and particularly towards the leaf nodes. A similar result can be derived for the label entropy. However, this expression for G implicitly assumes that missing or unobserved labels are negatives and this is not the case for us.

We therefore come up with an efficient alternative that learns from positive labels alone. We select the feature f^* and threshold t^* satisfying

$$\begin{aligned} \underset{f,t}{\text{argmin}} \quad & n_l \sum_{k=1}^{K} p_l(l_k = 1)(1 - p_l(l_k = 1)) + \\ & n_r \sum_{k=1}^{K} p_r(l_k = 1)(1 - p_r(l_k = 1)) \end{aligned} \quad (4)$$

where n_l and n_r are the number of points in the left and right child and $p_l(l_k = 1)$ and $p_r(l_k = 1)$ are the probabilities of observing label k as a positive label in the left and right child respectively (note that n_l, n_r, p_l and p_r are all functions of f and t). Furthermore, we define the probability of observing a positive label in a node as

$$p_l(l_k = 1) = \sum_{i=1}^{n_l} p_l(l_k = 1|i) p_l(i) \quad (5)$$

where the sum is over all the ad landing pages present in the left child, $p_l(l_k = 1|i)$ is the probability of sampling label k from the set of positive labels from ad landing page i and $p(i)$ is the probability of sampling ad landing page i in the left child (the distributions for the right child can be set up in a similar manner). Different choices of these probability distributions correspond to different underlying assumptions about the data. Empirically, we found that setting $p(l_k = 1|i)$ to be uniform on the positive labels yielded the best results. Thus, we set $p(l_k = 1|i) = y_{ik}/\sum_m y_{im}$ where we have used the notation that the labels for ad landing page i are specified in the vector $\mathbf{y}_i \in \{0,1\}^K$ with y_{ik} being 1 if label k is assigned as a positive label to ad landing page i and 0 otherwise. Rather than choosing $p(i)$ to also be uniform, we found it better to sample ad landing pages with probability proportional to how many positive labels they had – i.e. $p(i) = \sum_k y_{ik}/\sum_{ik} y_{ik}$. The intuition is that a landing page which has received clicks from a hundred queries is probably a lot more important than a landing page which has received clicks from only a single query. These choices were empirically found to lead to better results as compared to assuming that labels were independent but that unobserved labels were implicitly negative. The Gini index was also found to yield slightly better results than the label entropy.

Algorithm 1 MLRFEvalObj($\mathbf{X}, \mathbf{Y}, f, t$)

1: $I_l = \{i | \mathbf{x}_{if} \leq t\}$; $I_r = \{i | \mathbf{x}_{if} > t\}$
2: $n_l = |I_l|$; $n_r = |I_r|$
3: $p_l(l_k = 1|i) = \frac{y_{ik}}{\sum_m y_{im}}$; $p_r(l_k = 1|i) = \frac{y_{ik}}{\sum_m y_{im}}$
4: $p_l(i) = \frac{\sum_k y_{ik}}{\sum_{k, j \in I_l} y_{jk}}$; $p_r(i) = \frac{\sum_k y_{ik}}{\sum_{k, j \in I_r} y_{jk}}$
5: $p_l(l_k = 1) = \sum_{i \in I_l} p_l(l_k = 1|i) p_l(i)$;
$p_r(l_k = 1) = \sum_{i \in I_r} p_r(l_k = 1|i) p_r(i)$
6: $G = \frac{n_l}{n_l + n_r} \sum_{k=1}^{K} p_l(l_k = 1)(1 - p_l(l_k = 1)) +$
$\frac{n_r}{n_l + n_r} \sum_{k=1}^{K} p_r(l_k = 1)(1 - p_r(l_k = 1))$
7: **return** G

Note that these choices of label distributions intuitively generalize multi-label random forests from multi-class random forests by considering a node to be a bag of positive labels with the probability of a label being the probability of sampling it from the bag. Also note that the theoretical results for standard random forests [9] can be readily derived for our MLRF classifier. For instance, it is straightforward to show that as the number of trees increases asymptotically, MLRF's predictions will converge to the expected value of the ensemble generated by randomly choosing all parameters and that the generalization error of MLRF is bounded above by a function of the correlation between trees and the average strength of the trees. Finally, while we did assume label independence during random forest construction, label correlations present in the training data will be learnt and implicitly taken into account while making predictions. For instance, if two labels are perfectly correlated then they will end up in the same leaf nodes and hence will be either predicted, or not predicted, together. We do not take label correlations into account explicitly, such as via a second order label correlation model as is done in structured output prediction [39, 40], as prediction time would become quadratic in the number of labels.

All in all, these choices allow MLRF to scale to large problems and train efficiently on millions of points, labels and features without having to ever load the feature or label matrix into RAM. Care should be taken during training to ensure that the trees are properly regularized and balanced and do not over fit.

3.3.3 *Distributed Training*

We leverage the MapReduce framework [16] to distribute training over a thousand compute nodes each having only 2GB RAM. The naive strategy of growing a separate tree on each machine doesn't work due to the high cost of copying the training data. Planet [31] provides an implementation of a single label, single shallow tree model with tens of features. This approach didn't work well for us since we have a hundred, deep trees (which do not fit in 2GB RAM) and millions of labels and features. We introduce an extra layer into the Planet hierarchy to cope with the large number of features and the 10 million dimensional label distributions.

Each available worker machine is assigned the task of calculating the splitting cost for only a small subset of a node's data points. Combiner machines then aggregate the splitting cost across worker machines. Finally, maximizer machines choose the feature and threshold pair with minimal cost across all combiners. This hierarchy performs efficient load balancing while keeping communication overheads low and speeds up training significantly over Planet. As a result, we were able to train on 90 million points, 10 million labels and 6 million features in less than a day.

Data Set	Num Train (M)	Num Features (M)	Num Labels (M)	Num Test (M)
Wikipedia	1.53	1.88	0.97	0.66
Ads1	8.00	1.58	1.21	0.50
Web	40.00	2.62	1.21	1.50
Ads2	90.00	5.80	9.70	5.00

Table 1: Large scale multi-label learning data set statistics. Ads1, Ads2 and Web are proprietary data sets while Wikipedia is publicly available.

4. SPARSE SEMI-SUPERVISED MULTI-LABEL LEARNING FOR MITIGATING TRAINING SET BIASES

The multi-label random forest formulation developed so far does not take into account the fact that numerous positive labels are missing and that a few might have been inferred incorrectly during the automatic training set harvesting process. Other biases such as the ad serving bias, click position bias, query distribution bias have also not been dealt with explicitly. One could try to mitigate the effect of these biases by taking specific measures tailored to each individual bias. For instance, negative labels might be inferred using a pSkip model and the effect of the ad serving bias might be mitigated by training on a mix of organic search and ad clicks. In this Section, we develop an orthogonal procedure based on sparse semi-supervised multi-label learning which can complement other bias specific measures.

One approach to mitigating the effect of some of these biases would be to post-process the MLRF predictions – for instance, by performing a random walk on the click graph. We avoid such approaches in this paper as they would increase the cost of prediction. An alternative would be to clean up the training set annotations by first inferring real valued beliefs indicating the state of each label. Labels with a strong belief that they were in an incorrect state could be flipped to clean up the training data. A standard multi-label classifier could then be trained on the cleaned labels. However, this involves taking hard decisions about whether to flip labels or not early on in the training process. We found it more beneficial to adapt our classifier to directly train on the inferred belief vectors rather than the cleaned up labels.

To take a concrete example, suppose a car insurance ad had not been labelled with the marginally relevant bid phrase "cheap SUV insurance" during the automatic training set generation phase. Our sparse semi-supervised learning procedure (detailed below) estimated that the label should have been present with a belief of 0.6. Rather than adding the label with complete certainty to the training set, we get MLRF to train on the label with a belief of 0.6. The hope would be that MLRF would learn to predict the label for similar novel ads but with a low score and rank it below highly relevant labels such as "cheap car insurance" which were present with belief 1.0 during training.

It would be necessary to reformulate most classifiers in order for them to train on probabilistic, rather than certain, labellings (note that this setting is very different from regression where one also trains on real valued targets). However, by deliberate design, we need to make no changes to our random forest formulation or implementation as discussed in section 3. The only conceptual change is that now $\mathbf{y}_i \in \Re_+^K$ and that predictions are made by data points in leaf nodes voting for labels with non-negative real numbers rather than casting binary votes.

One way of inferring real valued scores for the missing labels would be by performing one-class collaborative filtering [30, 36] on the label matrix. Unfortunately, this seemed not to work well in our case. What did work was the intuition that even if a point was missing a label, some of its nearest neighbours were very likely to have that label. Thus, a point could infer its belief about the state of a particular label based on the beliefs of its neighbours. This notion of belief smoothness has been exploited in graph based semi-supervised learning (SSL) [48] though not in this particular context. We therefore formulate our sparse multi-label SSL problem as

$$\underset{\mathbf{F}}{Min}\, P(\mathbf{F}) = \tfrac{1}{2}\, Tr(\mathbf{F}^t(\mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}})\mathbf{F}) + \frac{\beta}{2}|\mathbf{F} - \mathbf{Y}|_{Fr}^2$$
$$s.t.\ |\mathbf{F}|_0 \leq L \tag{6}$$

where $\mathbf{F}$ and $\mathbf{Y}$ are our 90M×10M belief and label matrices with non-negative entries, $\mathbf{W}$ is a 90M×90M positive definite matrix with non-negative entries representing the similarity between two training points by the intersection of their label sets and $\mathbf{D}$ is a diagonal matrix representing the row or column sums of $\mathbf{W}$. The l_0 constraint on $\mathbf{F}$ is added as we know that most data points can only have relatively few positive labels. While other forms of sparsity have been investigated in SSL, such as sparsity in constructing the graph or sparsity in the use of unlabelled data [37], our formulation promoting sparsity in the belief vectors is novel and is also different from previous multi-label SSL formulations [13, 27] which do not learn sparse solutions.

The non-convex l_0 constraint results in a hard optimization problem which is typically solved by state-of-the-art

Data Set	Click Labels (%)				
	KEX	KEX +KSP	MLRF	MLRF +SSL	MLRF +SSL +KSP
Wikipedia	0.81	0.78	0.71	0.66	0.63
Ads1	0.83	0.76	0.71	0.65	0.61
Web	0.73	0.68	0.65	0.62	0.58
Ads2	0.77	0.73	0.69	0.63	0.59

Table 2: Automatic evaluation of phrase recommendations using the edit distance as proposed in [32] (smaller numbers are better). MLRF+SSL is almost 15% better than KEX on the Ads data sets and the same is true for MLRF+SSL+KSP as compared to KEX+KSP.

optimizers using greedy methods [35]. For instance, the forward greedy method computes the gradient $\nabla_{\mathbf{F}} P$ at each iteration but takes only a greedy co-ordinate step to maintain sparsity. Such algorithms require at least L iterations to obtain a L-sparse solution. This is infeasible at our scale since L runs into the billions and computing $\nabla_{\mathbf{F}} P$ is relatively expensive. Instead, we adapt the well-known iterative hard thresholding algorithm which takes fixed length gradient steps followed by a projection onto the l_0 ball to maintain sparsity. The iterates are given by

$$\mathbf{F}_{t+1} = \text{Top}_L(\frac{1}{1+\beta}\mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}\mathbf{F}_t + \frac{\beta}{1+\beta}Y) \quad (7)$$

where $\mathbf{F}_0 = \mathbf{Y}$ and the Top_L operator retains the top L elements of its argument while setting everything else to zero. The optimization is carried out before MLRF training and the resultant beliefs are then used as training labels for MLRF. In practice, we found that terminating the algorithm after 15 iterations resulted in a significant improvement in MLRF's predictions. Note that, in principle, this procedure could also be used as a post-processing step to clean up MLRF's predictions for novel ads. However, we wish to minimize prediction costs and hence do not explore this option further in this paper.

It is straightforward to show, using the technique of [7], that the iterative hard thresholding algorithm will converge to a stationary point which will be a local or global minimum of our sparse semi-supervised multi-label learning formulation. The following formal statements can be made

THEOREM 1. *Let $\mathbf{W}_{N\times N}$ be a positive definite matrix with non-negative entries and $\mathbf{Y}_{N\times K} \in \{0,1\}^{N\times K}$ be a binary label matrix. Then: (a) $P(\mathbf{F}_0) \geq P(\mathbf{F}_1) \geq P(\mathbf{F}_2) \geq \cdots$; (b) the sequence $\mathbf{F}_0, \mathbf{F}_1, \ldots$ converges to a stationary point $\mathbf{F}^*$; (c) $\mathbf{F}^*$ is the global optimum if $|\mathbf{F}^*|_0 < L$; and (d) $\mathbf{F}^*$ is a local optimum if $|\mathbf{F}^*|_0 = L$.*

5. EXPERIMENTS

In this Section, we evaluate the performance of the proposed MLRF and MLRF+SSL classifiers. It is demonstrated that our multi-label learning approach can recommend significantly better bid phrases as compared to ranking and NLP based techniques.

5.1 Data sets and ground truth

We present results on three proprietary Ads and web data sets of HTML documents for which phrase recommendations are sought. We also include Wikipedia, which is publically available, for the reproducibility of our MLRF results. Table 1 lists the statistics of these data sets out of which the largest has 5 million test ad landing pages. For each data set, we use a simple bag-of-words model and extract TF-IDF features. We remove stop words and any word which occurs in less than ten documents. Words occurring in the title and anchor text have their counts boosted by 10 due to their significance.

It is hard to obtain good quality ground truth labels in order to carry out large scale automated performance evaluation. At the end of the day. advertisers would prefer those recommendations which would increase their conversion ratio. However, conversion data might not always be readily available and might take time to measure even when it is. We therefore used clicks and relevance as proxies for conversions to generate ground truth.

For the Ads and Web data sets, a search engine query was assigned as a label to a document if that query resulted in a click on the document. All labels that failed to receive more than five clicks were discarded. As discussed, this resulted in biased training, validation and test sets. In particular, test points had missing and potentially incorrect labels. To mitigate some of these effects, the test sets were constructed by including only those documents that had more than twenty five queries attached as labels. This biased the test set towards more popular documents but allowed us to carry out large scale performance evaluation. For Wikipedia, the label space was chosen to be the set of Wikipedia categories, and each page was automatically labelled with the set of categories found at its bottom. The test set was uniformly sampled and was not biased towards the more popular Wikipedia pages.

Asking a team of human experts to measure relevance might potentially sidestep some of the biases introduced by automatically inferring ground truth labels from click logs. However, human judgements are expensive and it is often not possible to obtain them at large scale. Therefore, we also carried out evaluations on a uniformly sampled smaller test set of five hundred documents where performance was assessed by a team of three human experts judging the relevance of the recommendations to the given documents.

5.2 Algorithms and parameters

We compared the performance of MLRF and MLRF+SSL to KEX [46] (which was discussed in Section 2). KEX's major limitation was that it could only recommend those bid phrases that were present on the web page. This shortcoming was addressed in [32] using a monolingual translation model for candidate bid phrase expansion. We could not compare MLRF+SSL to [32] directly since their code is not available and no results have been published on a publically available data set. We therefore expanded KEX's candidate bid phrases using a research variant of Microsoft's bid phrase expansion engine referred to as KSP. KSP is a meta-algorithm which combines many state-of-the-art techniques for expanding a seed set of bid phrases based on how users rewrite queries in a session, co-click and co-bid information, *etc.* Note that, since MLRF doesn't rely on any of these signals, KSP could be used to expand the set of bid phrases recommended by MLRF as well even though MLRF is already capable of recommending bid phrases not present on the web page.

A hundred trees were learnt in MLRF's random forest for each data set. Care was taken to avoid over fitting and to ensure that the learnt trees were not lopsided. Trees were grown until each leaf node had no more than a thousand data points. We also investigated whether our multi-label sparse SSL formulation could mitigate the effect of biases introduced during automatic training set generation. The similarity matrix $\mathbf{W}$ was obtained from our click logs rather than being computed from feature vectors. In particular, W_{ij} was set to be the total number of queries which resulted in a click on both document i and document j while β was set to 0.6 and the sparsity factor L was set to twice the number of total positive labels in the data set.

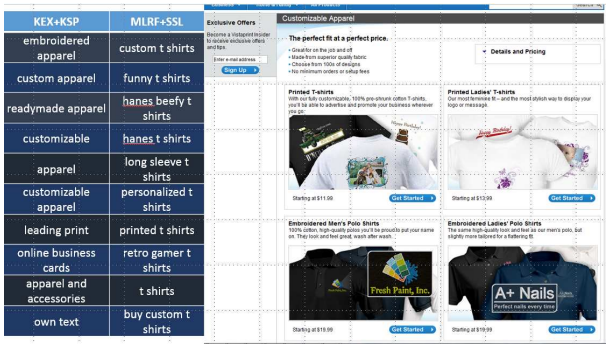
5.3 Metrics

The notion of what constitutes a good prediction changes when moving from thousands of labels to millions of labels.

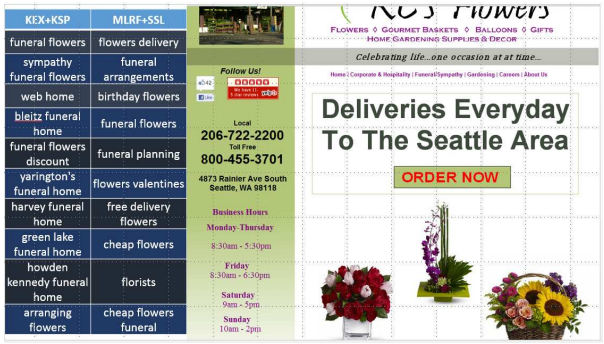
(a) Geico Insurance

(b) Dominos Pizza

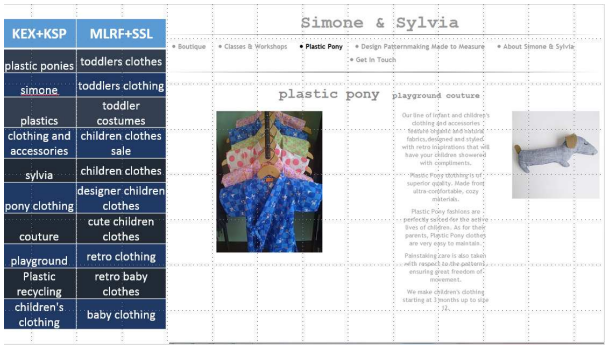
(c) MetLife Insurance



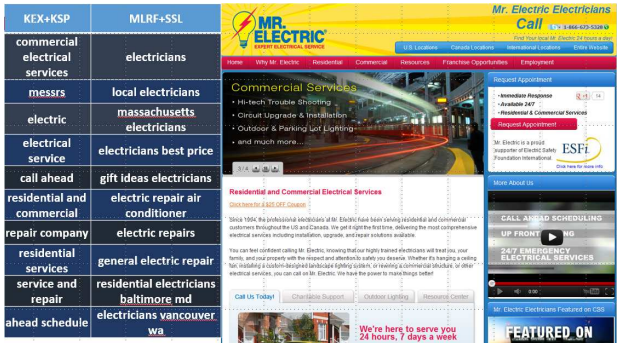
(d) Vistaprint Custom T-Shirts



(e) KCS Flowers



(f) Simone and Sylvia's Kid's Clothing



(g) Mr Electic

(h) Wanta Thai Restaurant

Figure 1: Phrase recommendations by KEX+KSP and MLRF+SSL: KEX+KSP did not recommend any phrases for (a) and (b) and produced poor recommendations for (c)-(g). MLRF+SSL recommended good phrases for (a)-(g) but failed for (h). See text for details. Figure best viewed under high magnification.

Data Set	Click Labels (%)					Human Verification(%)				
	KEX	KEX +KSP	MLRF	MLRF +SSL	MLRF+ SSL+KSP	KEX	KEX +KSP	MLRF	MLRF +SSL	MLRF+ SSL+KSP
Wikipedia	11.63	10.81	15.72	18.53	18.01	17.51	22.14	24.46	27.17	31.48
Ads1	11.96	12.38	18.13	19.88	21.54	41.95	43.27	45.86	47.53	51.08
Web	18.42	19.88	22.51	25.32	26.66	47.69	48.13	50.47	51.83	53.69
Ads2	12.45	14.35	15.91	17.12	19.24	36.69	40.07	41.28	43.78	46.77

Table 3: Automatic and human evaluation using precision at 10 for phrase recommendation (larger numbers are better). MLRF+SSL was found to be about 5% better than KEX on the Ads data sets according to both automatic and human evaluation. Expanding the recommendations using KSP improved the absolute performance of both methods but the difference in performance between the multi-label and the ranking approach remained around 5% for automatic evaluation and around 4% for human evaluation. Note that precision values were computed for only those documents for which KEX was able to generate recommendations otherwise the difference in performance would have been even greater.

Establishing relevance or irrelevance is no longer as straightforward. Competing requirements of diversity, specificity and succinctness also come into play. Performance evaluation is further complicated by the fact that relevant labels might be missing in the ground truth annotations. It can be shown that there do not exist performance metrics which are invariant to missing labels. However, one can come up with metrics, such as precision, where the loss incurred for predicting the ground truth labels for a given data point is no greater than any other set of labels even when labels are missing from the ground truth.

Most of the previous studies have recommended evaluating bid phrase suggestions using precision and the edit distance at the word level [32, 46]. We therefore report results using both metrics for automated evaluation on our large scale test sets. We also carry out a human evaluation where the predictions for various methods were passed on to an external team of 3 human experts who judged whether the predictions were relevant or not. We report the precision at 10 averaged over the 3 judgements per test point.

5.4 Results

MLRF recommended at least 100 bid phrases in almost 98% of the cases. KEX+KSP's coverage was poorer– for instance, it was only 82% on the Ads2 data set since it failed to recommend any bid phrases in many cases. Tables 3 and 4 report results for the edit distance and precision at 10 on only those test points for which KEX+KSP returned at least 10 predictions (the results for KEX+KSP would have been worse if computed over all ads). The ordering of algorithms is consistently that KEX < KEX+KSP < MLRF < MLRF+SSL < MLRF+SSL+KSP for all modes of evaluation except on Wikipedia for which KSP has not been tuned. MLRF+SSL was better than KEX by almost 15% using the edit distance on the Ads data sets. The same was true for MLRF+SSL+KSP as compared to KEX+KSP. The differences in performance were slightly lower in terms of precision at 10. MLRF+SSL was found to be about 5% better than KEX on the Ads data sets according to both automatic and human evaluation. Expanding the recommendations using KSP improved the absolute performance of both methods. KSP improved KEX's recommendations by bringing in words not present on the given web page. KSP also helped expand MLRF+SSL's recommendations by leveraging orthogonal training data. The difference in performance between the multi-label and the ranking approach remained around 5% for automatic evaluation and around 4% for human evaluation.

The results also indicated that our approach of training MLRF on label beliefs inferred using the multi-label sparse SSL formulation was beneficial. Performance on the Ads data sets improved by 6% using the edit distance and by 1-2% using precision at 10.

The difference in the quality of bid phrase recommendations can be observed in Figure 1. KEX did not make any recommendations for Geico and Domino's. We found many premium ad landing pages to be predominantly image or animation based and what appeared to be text on these pages was actually embedded images. As a result, KEX would make very few predictions, sometimes even none, as its binary classifier was restricted to evaluating a very small set of phrases which might not ultimately pass the recommendation threshold. On the other hand, MLRF+SSL found enough signal in the simple bag-of-words features extracted from the raw HTML to make good bid phrase recommendations. This demonstrates that MLRF is innately able to recommend phrases not present on the page. Furthermore, without any hand crafting, MLRF picked up the common trick of recommending a competitor's bid phrases. For instance, it recommended that Geico bid on AllState's phrase ("all state car insurance coupon code") and that MetLife bid on "geico insurance" in order to garner more visibility for MetLife's ad. Almost all of MLRF's predictions were good or borderline for MetLife's ad whereas only 4 of KEX+KSP's recommendations were good. The rest were found to be poor and could potentially cause the advertiser to lose money. The same was true for ads for Vistaprint Custom T-Shirts, KCS Flowers, Simone & Sylvia's Kid's Clothing and Mr. Electric. For instance, KEX latched on to the irrelevant bid phrase "online business cards" from Vistaprint's website template. MLRF, on the other hand, automatically learnt that business cards were not relevant to ads for T-Shirts and instead focussed on recommending different types of T-Shirts. In addition, KEX+KSP sometimes recommended fairly generic phrases such as "massachusetts", "customizable" and "apparel" which might lead to many clicks and exhaust the advertiser's budget but lead to few conversions. It also frequently recommended irrelevant phrases such as "call ahead' and "own text". MLRF's recommendations tended to be more specific, targeted and relevant. Even in terms of diversity, MLRF's predictions could be much better than KEX+KSP's, as can be seen by

the recommendations for KCS Flowers. KEX+KSP took a rather morbid view and recommended only funeral related bid phrases whereas MLRF also recommended flowers for birthdays and valentines.

5.5 Limitations

By and large, our proposed multi-label MLRF learning approach was found to be better than the ranking based technique of KEX+KSP. However, MLRF also has some limitations. In particular, MLRF is limited to suggesting phrases from the set of 10 million training queries. While this set is large enough to cover most of the commercially important phrases, it doesn't include new brand names and new products from small businesses. We retrain MLRF+SSL frequently to mitigate this fact and bring in new brand names and products into our label set. More serious errors can happen due to incorrect labels in the training set. For instance, our predictions for the Wanta Thai Restaurant in Figure 1 were quite poor. Many of the suggestions, particularly those beyond the top 10, were more relevant to an Italian restaurant rather than a Thai restaurant. This can be easily debugged in the random forest framework by tracing the ad down to its leaf nodes and examining its nearest neighbours. We found two sources of error. First, a few Italian restaurant ads had been mislabelled as Thai restaurants during training and were therefore ending up in the same leaf node as Thai restaurants and contributing Italian restaurant labels. The second source of error was an ad for a half Thai-half Italian restaurant.

6. CONCLUSIONS

We posed the problem of recommending bid phrases from a given ad landing page as a supervised multi-label learning task with millions of labels. Each label, in our formulation, corresponds to a separate bid phrase. We developed a novel multi-label random forest classifier with prediction costs that are logarithmic in the number of labels while avoiding feature and label space compression. This enabled us to efficiently carry out fine grained bid phrase recommendation in a few milliseconds using 10 Gb of RAM. Our node splitting criterion learns from positive data alone allowing us to train non-linear classifiers from large data sets with 90 million training points, 10 million labels and 6 million features. We developed distributed training algorithms which parallelize efficiently over a thousand nodes each having only 2Gb of RAM and which never need to load the feature or label matrix into memory. As a result, we were able to train our multi-label random forest classifier on a medium sized cluster in less than a day. Standard generalization bounds for our proposed classifier can readily be derived in terms of the correlation between the trees in the forest and the prediction accuracy of individual trees.

Training data for our classifier was mined automatically from the click logs. This introduced various types of biases into the training set including numerous missing, and a few incorrect, labels for each ad landing page. We therefore infer beliefs in the state of each label using a novel sparse semi-supervised multi-label learning formulation. The formulation is optimized via an iterative hard thresholding algorithm, for which a straightforward proof of convergence can be derived, and which is orders of magnitude faster than competing state-of-the-art greedy co-ordinate descent strategies at our scale. We then extend our MLRF formulation to train on the inferred beliefs in the state of each label and show that this leads to better bid phrase recommendations as compared to the standard supervised learning paradigm of directly training on the given labels.

We evaluated the bid phrase recommendations of our multi-label random forest classifier on a test set of 5 million ads. We demonstrated that our proposed MLRF technique has many advantages over ranking based methods such as KEX. In particular, unlike KEX, MLRF can recommend bid phrases not present on the ad landing page. This can be particularly helpful while dealing with ad landing pages that are text impoverished. MLRF's recommendations can also be significantly better as its high capacity classifier can potentially learn a much richer model of the relationship between ad landing pages and relevant bid phrases. Monolingual translation models and query expansion techniques, such as KSP, bring in complementary information and can be used to improve the recommendations of both MLRF and KEX.

In conclusion, this paper demonstrates that bid phrase recommendation can be posed as a multi-label learning task and that learning with millions of labels can be made tractable and accurate. Our formulation is general and offers a potentially different way of thinking about query recommendation problems in search and advertising.

Acknowledgements

We are grateful to Deepak Bapna, Samy Bengio, Prateek Jain, A. Kumaran, Mehul Parsana, Krishna Leela Poola, Adarsh Prasad, Varun Singla and Scott Yih for helpful discussions and feedback.

7. REFERENCES

[1] V. Abhishek and K. Hosanagar. Keyword generation for search engine advertising using semantic similarity between terms. In *ICEC*, pages 89–94, 2007.

[2] I. Antonellis, H. G. Molina, and C. C. Chang. Simrank++: query rewriting through link analysis of the click graph. *VLDBE*, 1(1):408–421, 2008.

[3] S. Bengio, J. Weston, and D. Grangier. Label embedding trees for large multi-class tasks. In *NIPS*, 2010.

[4] A. Beygelzimer, J. Langford, Y. Lifshits, G. Sorkin, and A. Strehl. Conditional probability tree estimation analysis and algorithms. In *UAI*, 2009.

[5] W. Bi and J. T. Kwok. Multilabel classification on tree- and dag-structured hierarchies. In *ICML*, 2011.

[6] H. Blockeel, L. D. Raedt, and J. Ramon. Top-down induction of clustering trees. In *ICML*, pages 55–63, 1998.

[7] T. Blumensath and M. Davies. Iterative thresholding for sparse approximations. *Journal of Fourier Analysis and Applications*, 14:629–654, 2004.

[8] M. Boutell, J. Luo, X. Shen, and C. Brown. Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757–1771, 2004.

[9] L. Breiman. Random forests. *Machine Learning*, 45(1), 2001.

[10] A. Broder, P. Ciccolo, E. Gabrilovich, V. Josifovski, D. Metzler, L. Riedel, and J. Yuan. Online expansion of rare queries for sponsored search. In *WWW*, pages 511–520, 2009.

[11] C. Carpineto and G. Romano. A survey of automatic query expansion in information retrieval. *CSUR*, 44(1):1:1–1:50, 2012.

[12] N. Cesa-Bianchi, C. Gentile, and L. Zaniboni. Incremental algorithms for hierarchical classification. *JMLR*, 7, 2006.

[13] G. Chen, Y. Song, F. Wang, and C. Zhang. Semi-supervised multi-label learning by solving a sylvester equation. In *SDM*, pages 410–419, 2008.

[14] Y. Choi, M. Fontoura, E. Gabrilovich, V. Josifovski, M. Mediano, and B. Pang. Using landing pages for sponsored search ad selection. In *WWW*, pages 251–260, 2010.

[15] A. Clare and R. D. King. Knowledge discovery in multi-label phenotype data. In *PKDD*, pages 42–53, 2001.

[16] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *OSDI*, 2004.

[17] O. Dekel and O. Shair. Multiclass-multilabel classification with more classes than examples. In *AISTATS*, 2010.

[18] J. Deng, S. Satheesh, A. C. Berg, and F. Li. Fast and balanced: Efficient label tree learning for large scale object recognition. In *NIPS*, 2011.

[19] A. Elisseeff and J. Weston. A kernel method for multi-labelled classification. In *NIPS*, pages 681–687, 2001.

[20] B. Hariharan, S. V. N. Vishwanathan, and M. Varma. Efficient max-margin multi-label classification with applications to zero-shot learning. *ML*, 2012.

[21] D. Hsu, S. Kakade, J. Langford, and T. Zhang. Multi-label prediction via compressed sensing. In *NIPS*, 2009.

[22] S. Ji, L. Sun, R. Jin, and J. Ye. Multi-label multiple kernel learning. In *NIPS*, pages 777–784, 2008.

[23] A. Joshi and R. Motwani. Keyword generation for search engine advertising. In *ICDMW*, pages 490–496, 2006.

[24] A. Kapoor, R. Viswanathan, and P. Jain. Multilabel classification using bayesian compressed sensing. In *NIPS*, 2012.

[25] D. Kocev, C. Vens, J. Struyf, and S. Dzeroski. Ensembles of multi-objective decision trees. In *ECML*, pages 624 – 631, 2007.

[26] A. Z. Kouzani and G. Nasireding. Multilabel classification by bch code and random forests. 2, 2009.

[27] Y. Liu, R. Jin, and L. Yang. Semi-supervised multi-label learning by constrained non-negative matrix factorization. In *AAAI*, 2006.

[28] A. Malekian, C.-C. Chang, R. Kumar, and G. Wang. Optimizing query rewrites for keyword-based advertising. In *EC*, pages 10–19, 2008.

[29] M. Munsey, J. Veilleux, S. Bikkani, A. Teredesai, and M. D. Cock. Born to trade: A genetically evolved keyword bidder for sponsored search. In *CEC*, pages 1–8, july 2010.

[30] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. M. Lukose, M. Scholz, and Q. Yang. One-class collaborative filtering. In *ICDM*, 2008.

[31] B. Panda, J. Herbach, S. Basu, and R. J. Bayardo. Planet: Massively parallel learning of tree ensembles with mapreduce. *PVLDB*, 2(2), 2009.

[32] S. Ravi, A. Broder, E. Gabrilovich, V. Josifovski, S. Pandey, and B. Pang. Automatic generation of bid phrases for online advertising. In *WSDM*, 2010.

[33] J. Rousu, C. Saunders, S. Szedmak, and J. Shawe-Taylor. Kernel-based learning of hierarchical multilabel classification models. *JMLR*, 7, 2006.

[34] M. R. Segal. Tree-structured methods for longitudinal data. 87(418):407–418, 1992.

[35] S. Shalev-Shwartz, N. Srebro, and T. Zhang. Trading accuracy for sparsity in optimization problems with sparsity constraints. *SIAM J. on Optimization*, 20(6), 2010.

[36] V. Sindhwani, S. S. Bucak, J. Hu, and A. Mojsilovic. One-class matrix completion with low-density factorizations. In *ICDM*, 2010.

[37] S. Sun and J. Shawe-Taylor. Sparse semi-supervised learning using conjugate functions. *JMLR*, 2010.

[38] L. Tang, S. Rajan, and V. K. Narayanan. Large scale multi-label classification via metalabeler. In *WWW*, pages 211–220, 2009.

[39] B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In *NIPS*, 2003.

[40] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *JMLR*, 6, 2005.

[41] G. Tsoumakas, I. Katakis, and I. Vlahavas. Effective and efficient multilabel classification in domains with large number of labels. In *ECML/PKDD*, 2008.

[42] G. Tsoumakas and I. P. Vlahavas. Random k -labelsets: An ensemble method for multilabel classification. In *ECML*, 2007.

[43] J. Weston, S. Bengio, and N. Usunier. Large scale image annotation: Learning to rank with joint word-image embeddings. *Machine Learning*, 81(1), 2010.

[44] X. Wu and A. Bolivar. Keyword extraction for contextual advertisement. In *WWW*, pages 1195–1196, 2008.

[45] R. Yan, J. Tesic, and J. R. Smith. Model-shared subspace boosting for multi-label classification. In *KDD*, 2007.

[46] W. Yih, J. Goodman, and V. R. Carvalho. Finding advertising keywords on web pages. In *WWW*, 2006.

[47] M. L. Zhang, J. M. Peña, and V. Robles. Feature selection for multi-label naive bayes classification. *Inf. Sci.*, 179(19), 2009.

[48] X. Zhu. Semi-supervised learning literature survey. Technical Report 1530, University of Wisconsin Madison, 2008.