Slice: Scalable Linear Extreme Classifiers Trained on 100 Million Labels for Related Searches

Himanshu Jain*
himanshu.j689@gmail.com

Venkatesh Balasubramanian†
t-venkb@microsoft.com

Bhanu Chunduri†
bhanuc@microsoft.com

Manik Varma*†
manik@microsoft.com

ABSTRACT

This paper reformulates the problem of recommending related queries on a search engine as an extreme multi-label learning task. Extreme multi-label learning aims to annotate each data point with the most relevant *subset* of labels from an extremely large label set. Each of the top 100 million queries on Bing was treated as a separate label in the proposed reformulation and an extreme classifier was learnt which took the user's query as input and predicted the relevant subset of 100 million queries as output. Unfortunately, state-of-the-art extreme classifiers have not been shown to scale beyond 10 million labels and have poor prediction accuracies for queries. This paper therefore develops the Slice algorithm which can be accurately trained on low-dimensional, dense deep learning features popularly used to represent queries and which efficiently scales to 100 million labels and 240 million training points. Slice achieves this by reducing the training and prediction times from linear to logarithmic in the number of labels based on a novel negative sampling technique. This allows the proposed reformulation to address some of the limitations of traditional related searches approaches in terms of coverage, density and quality.

Experiments on publically available extreme classification datasets with low-dimensional dense features as well as related searches datasets mined from the Bing logs revealed that Slice could be more accurate than leading extreme classifiers while also scaling to 100 million labels. Furthermore, Slice was found to improve the accuracy of recommendations by 10% as compared to state-of-the-art related searches techniques. Finally, when added to the ensemble in production in Bing, Slice was found to increase the trigger coverage by 52%, the suggestion density by 33%, the overall success rate by 2.6% and the success rate for tail queries by 12.6%. Slice's source code can be downloaded from [21].

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; *Supervised learning by classification*;

KEYWORDS

Extreme multi-label learning, query recommendation, large-scale learning, negative sampling

ACM Reference Format:
Himanshu Jain, Venkatesh Balasubramanian, Bhanu Chunduri, and Manik Varma. 2019. Slice: Scalable Linear Extreme Classifiers trained on 100 Million Labels for Related Searches. In *The Twelfth ACM International Conference on Web Search and Data Mining (WSDM '19), February 11–15, 2019, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3289600.3290979

1 INTRODUCTION

Objective: This paper develops the Slice (Scalable LInear extreme ClassifiErs) algorithm for extreme multi-label learning based on low-dimensional dense features. Slice accurately and efficiently scales to problems involving 100 million labels and 240 million training points which are far beyond the scaling capabilities of all other extreme classifiers. This allows the reformulation of the related searches problem as an extreme classification task and Slice was shown to significantly increase the coverage, density and quality of related searches on Bing when added to the ensemble of leading related searches techniques in production.

Related searches: Given a query (referred to as the trigger) submitted by a user on a search engine, related searches aims to recommend related queries (referred to as suggestions) that might serve the user's information requirements better or recommend suggestions that the user might ask in addition to get more information on the topic. Traditional approaches to related searches are based on sessions information, the query-url, query-flow and term-query graphs, query synthesis and deep query embedding techniques. Unfortunately, many of these techniques might not be able to make recommendations for tail and previously unseen triggers thereby reducing the trigger coverage. Furthermore, many of these techniques recommend very few suggestions for tail triggers thereby also reducing the suggestion density. This can be seen in Figure 2 of the supplementary material where Bing, even though it has a large ensemble of leading techniques in production, was able to recommend only a single suggestion "cam newton shoulder surgery" in response to the trigger "cam procedure shoulder". Finally, the quality of suggestions recommended by many of these techniques might also be suspect for tail triggers. For instance, the trigger "cam procedure shoulder" was a query about the Comprehensive Arthroscopic Management shoulder surgery procedure whereas Bing recommended an irrelevant suggestion about the shoulder surgery of the American football quarterback Cam Newton. Slice tackles each of these three limitations by reformulating related searches as an extreme multi-label learning task.

*Indian Institute of Technology Delhi
†Microsoft AI & Research

WSDM '19, February 11–15, 2019, Melbourne, VIC, Australia

ACM ISBN 978-1-4503-5940-5/19/02…$15.00
https://doi.org/10.1145/3289600.3290979

Extreme classification: Extreme multi-label learning addresses the problem of learning a classifier that can annotate a data point with the most relevant *subset* of labels from an extremely large label set. For instance, there are more than a million labels (tags) on Wikipedia and one might wish to build an extreme multi-label classifier that tags a new article or web page with the subset of the most relevant Wikipedia labels. Note that multi-label classification is distinct from multi-class classification which aims to predict a single mutually exclusive label.

Reformulation: Each of the top 100 million queries on Bing was treated as a separate label in the proposed reformulation. An extreme classifier was learnt which took the trigger's feature vector as input and predicted the relevant subset of 100 million queries as the recommended suggestions. Each trigger was represented by its 64 dimensional CDSSM [18, 52] embedding as state-of-the-art techniques have shown that low-dimensional dense deep embeddings [14, 16, 18, 25, 36, 49, 52] are better suited for representing queries than high-dimensional sparse bag-of-words features used in extreme classification thus far. Reformulating related searches as a classification problem could help increase trigger coverage as the extreme classifier was designed from the outset to make predictions on previously unseen triggers. Suggestion density could also be increased as the classifier was trained to predict from 100 million labels (suggestions). Finally, the accuracy of recommendations could also be increased as a high-capacity classifier with more than 6.4 billion parameters (not counting the CDSSM parameters) was learnt so as to model the variability of tail queries.

State-of-the-art extreme classifiers: Leading extreme classifiers such as DISMEC [3], PDSparse [64] and PPDSparse [63] scale to about a million labels. Their training and prediction costs grow linearly with the number of labels rendering them unsuitable for the proposed reformulation. The state-of-the-art is represented by Parabel [47] which cuts down both training and prediction costs from linear to logarithmic. Unfortunately, Parabel has also not been shown to scale beyond 10 million labels and it is unable to make accurate predictions for low-dimensional features such as CDSSM.

Slice: Slice learns a separate linear classifier per label based on the 1-vs-All approach. Given a training set of N examples in D dimensions with L labels, Slice cuts down the training time from $O(NDL)$ to $O(ND \log L)$ by training each label's classifier on only $O(\frac{N}{L} \log L)$ of the most confusing negative examples rather than on all N examples. This is achieved efficiently on the basis of a novel negative sampling technique which is demonstrated to be significantly more accurate for low-dimensional dense feature representations than Parabel and PPDSparse's negative sampling techniques. Slice also cuts down the prediction time per test point from $O(DL)$ to $O(D \log L)$ by evaluating the classifiers of only the most probable $O(\log L)$ labels rather than all L labels.

Experiments: Experiments were carried out on publically available benchmark extreme classification datasets based on state-of-the-art deep learning XML-CNN [29] embeddings. Experiments were also carried out on related searches datasets mined from the Bing logs with queries represented using CDSSM [18, 52] embeddings. It was observed that Slice could surpass Parabel's prediction accuracies by as much as 16% on these datasets. Slice was also found to increase the accuracy of recommendations by at least 10% as compared to leading related searches techniques. Finally, on being added to the related searches ensemble in production on Bing, Slice was observed to increase the trigger coverage by 52%, the suggestion density by 33%, the overall success rate by 2.6% and the success rate for tail queries by 12.6%.

Contributions: This paper: (a) reformulates the related searches problem as an extreme classification task; (b) develops the Slice algorithm for extreme multi-label learning with low-dimensional dense features that scales to 100 million labels and 240 million training points and (c) demonstrates that Slice could significantly improve related searches on Bing when deployed in production. Slice's source code can be downloaded from [21].

2 RELATED WORK

Extreme multi-label learning: Much progress has recently been made in extreme multi-label learning [1, 3, 6, 11, 12, 17, 22, 23, 28, 29, 37, 41, 46–48, 53, 54, 57, 60, 61, 63–65, 67–69] and details of some of these methods can be found in Section 1 of the supplementary material. Unfortunately, none of these approaches have been shown to scale beyond 10 million labels. Furthermore, the prediction accuracy of some of these methods, including PPDSparse [63] and Parabel [47], can degrade significantly on moving from high-dimensional sparse bag-of-words features to low-dimensional dense embeddings as reported in Section 4.

Negative sampling: Techniques for subsampling the negative training points have been developed for various applications [30, 35, 36, 38, 42, 50, 62] but might not apply directly to extreme classification. For instance, negative sampling techniques for imbalanced data [30, 38] focus on improving the AUC rather than on improving scalability without sacrificing accuracy as is needed for extreme multi-label learning. Similarly, the popular approach of subsampling the negatives uniformly at random as done in traditional recommender systems [35, 50, 62] or sampling according to the label frequency as done in language modelling [36] can lead to a large loss in accuracy in extreme classification. The negative sampling techniques that are most relevant to Slice are the ones developed for scaling extreme classifiers such as PPDSparse and Parabel. Unfortunately, as discussed in more detail in Section 3, neither of these techniques works well for low-dimensional dense features. Slice therefore proposes a novel negative sampling technique based on a generative model which can be evaluated accurately and efficiently for low-dimensional dense features using Approximate Nearest Neighbour Search (ANNS).

ANNS: A survey of ANNS can be found in [27]. Slice relies on the state-of-the-art Hierarchical Navigable Small World Graph (HNSW) algorithm [31] for sampling its negative training points efficiently. While other algorithms [2, 10, 19, 26, 39, 58, 66] could also have been used, HNSW was found to perform slightly better on the related searches datasets. Note that the accuracy of HNSW and other ANNS algorithms degrades as the dimensionality of the feature space increases. Slice is therefore best suited for representations having at most a few thousand dimensions rather than millions of dimensions as in the bag-of-words case. Finally, ANNS should not be confused with k-nearest neighbour (kNN) classification. While Slice relies on ANNS for scalability during training and prediction, its 1-vs-All

formulation is significantly more accurate than approximate kNN classification based on HNSW as demonstrated in the experiments.

Related searches: Most related searches techniques can be categorized as being based on sessions information [7, 8, 13, 43, 51, 55], the query-url, query-flow or term-query graphs [4, 33, 59], query synthesis methods [20, 24] or deep learning techniques [55]. Unfortunately, the trigger coverage, suggestion density and quality of recommendations could be poor for many of these techniques. For instance, some sessions and query graph based methods [7, 43, 51] can only recommend suggestions for previously seen triggers thereby limiting their trigger coverage. The suggestion density could also be limited for tail queries which weren't issued in the same session as a head query. Furthermore, intent changes within a session could lead to poor recommendations. While many techniques have been proposed to address these limitations [8, 55] they are often not scalable or do not generate sufficiently diverse recommendations.

3 SLICE

Introduction: Designing multi-label classifiers that scale to 100 million labels and 240 million training points while maintaining accuracy is an extremely challenging task. 1-vs-All approaches, where a separate linear classifier is learnt per label, have surpassed tree [1, 22, 23, 48, 53, 54] and embedding [6, 11, 12, 17, 28, 37, 60, 69] based methods in terms of accuracy but face scalability issues due to their linear training and prediction costs. A well established approach to scale up training [47, 63] is to learn from all the positive examples, as these are relatively few in number, and just a few of the negative examples from the millions available. The key technical challenge then lies in efficiently identifying and training on the most confusing negative examples for each label. Slice therefore efficiently estimates $O(\frac{N}{L}\log L)$ of these "hard" negatives based on a generative model approximation of its discriminative counterpart thereby reducing training time to $O(ND\log L)$. Slice also reduces the prediction time to $O(D\log L)$ so as to meet the throughput and latency requirements of real-world applications. Slice achieves this by first finding the most probable positive labels using the generative model and then evaluating discriminative classifiers only for the shortlisted labels.

Discriminative model: Given a set of N independent and identically distributed training points $\{(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^N\}$ with unit normalized features lying on the D dimensional unit hypersphere $\mathbf{x}_i \in \mathbb{S}^D$ and L dimensional label vectors $\mathbf{y}_i \in \{-1, +1\}^L$, Slice follows the multi-label 1-vs-All approach and assumes that the joint label distribution $\mathbb{P}(\mathbf{y}|\mathbf{x})$ factorizes as the product of independent marginal logistic distributions such that $\mathbb{P}(\mathbf{y}_i|\mathbf{x}_i; \mathbf{W}^D, \mathbf{b}^D) = \prod_{l=1}^L \mathbb{P}(y_{il}|\mathbf{x}_i; \mathbf{w}_l^D, b_l^D) = \prod_{l=1}^L \left(1 + e^{-y_{il}(\mathbf{w}_l^{D^\top}\mathbf{x}_i + b_l^D)}\right)^{-1}$. The maximum *a posteriori* estimate of Slice's parameters can be obtained by solving the following L independent optimization problems

$$w_l^D, b_l^D = \underset{w_l, b_l}{\operatorname{argmin}} \frac{1}{2}\mathbf{w}_l^\top \mathbf{w}_l + C\sum_{i=1}^N \log\left(1 + e^{-y_{il}(\mathbf{w}_l^\top \mathbf{x}_i + b_l)}\right) \quad (1)$$

$$\Rightarrow \mathbf{w}_l^D = C\sum_{i=1}^N \mathbb{P}(-y_{il}|\mathbf{x}_i; \mathbf{w}_l^D, b_l^D) y_{il}\mathbf{x}_i \quad (2)$$

where (2) can be obtained by differentiating (1) and equating to zero. This indicates that the training points for which $\mathbb{P}(-y_{il}|\mathbf{x}_i; \mathbf{w}_l^D, b_l^D) < \epsilon$ contribute little to the optimal solution and can potentially be discarded from the optimization procedure without a significant loss in accuracy. Unfortunately, (2) is unhelpful in itself as $\mathbb{P}(-y_{il}|\mathbf{x}_i; \mathbf{w}_l^D, b_l^D)$ cannot be evaluated as $\mathbf{w}_l^D$ and b_l^D are unknown. Furthermore, even if $\mathbb{P}(-y_{il}|\mathbf{x}_i; \mathbf{w}_l^D, b_l^D)$ was accessible, evaluating it for all training points and labels in order to determine which ones to keep and which ones to discard would incur $O(NDL)$ cost thereby defeating the very purpose of subsampling.

Reducing complexity: Slice addresses these limitations by leveraging the popular assumption that the average number of positive labels per point is at most $M\log L$ [22, 46, 47] where $M < 7$ is a small constant across datasets (see Table 1). Slice approximates $\mathbb{P}(-y_{il}|\mathbf{x}_i; \mathbf{w}_l^D, b_l^D)$ by $\mathbb{P}(-y_{il}|\mathbf{x}_i; \mathbf{w}_l^G, b_l^G)$ using a generative model which can be learnt in time $O(ND\log L)$. It then identifies $O\left(\frac{N}{L}\log L\right)$ negative examples for each label having some of the largest values of $\mathbb{P}(-y_{il}|\mathbf{x}_i; \mathbf{w}_l^G, b_l^G)$ using Approximate Nearest Neighbour Search (ANNS) in time $O((N+L)D\log L)$. Finally, Slice is trained in time $O\left(\frac{N}{L}D\log L\right)$ for each label by optimising (1) over all the label's positive examples as well as the negative examples identified by ANNS for the label. Slice's overall training complexity is therefore $O(ND\log L)$ as $N > L$.

Generative model: Slice assumes that each training point i was generated according to the following procedure. Each label l was either independently activated ($y_{il} = +1$) or deactivated ($y_{il} = -1$) with probability π_l and $1 - \pi_l$ respectively such that $\mathbb{P}(\mathbf{y}_i; \boldsymbol{\pi}) = \prod_{l=1}^L \pi_l^{\frac{1}{2}(1+y_{il})}(1-\pi_l)^{\frac{1}{2}(1-y_{il})}$. Having sampled the label vector $\mathbf{y}_i$, a feature vector lying on the surface of the D dimensional unit hypersphere $\mathbb{S}^D$ was sampled from the distribution

$$\mathbb{P}(\mathbf{x}_i|\mathbf{y}_i) = \begin{cases} e^{\frac{1}{2}\sum_{l=1}^L (1+y_{il})\left(\gamma_l^+ \mathbf{x}_i^\top \boldsymbol{\mu}_l^+ + Z_l^+\right) + (1-y_{il})\left(\gamma_l^- \mathbf{x}_i^\top \boldsymbol{\mu}_l^- + Z_l^-\right)}, & \mathbf{x} \in \mathbb{S}^D \\ 0, & \mathbf{x} \notin \mathbb{S}^D \end{cases}$$

with parameters $\{(\gamma_l^+, \gamma_l^-, \boldsymbol{\mu}_l^+, \boldsymbol{\mu}_l^-)_{l=1}^L\}$ where $Z_l^\pm$ are normalizing constants dependent on the modified Bessel function of order $D/2-1$ evaluated at $\gamma_l^\pm$. The discriminative counterpart of the proposed generative model can then be straightforwardly obtained as

$$\begin{aligned}\mathbb{P}(\mathbf{y}_i|\mathbf{x}_i; \mathbf{W}^G, \mathbf{b}^G) &= \mathbb{P}(\mathbf{x}_i|\mathbf{y}_i; \{(\gamma_l^+, \gamma_l^-, \boldsymbol{\mu}_l^+, \boldsymbol{\mu}_l^-)_{l=1}^L\})\mathbb{P}(\mathbf{y}_i; \boldsymbol{\pi})/\mathbb{P}(\mathbf{x}_i) \\ &= \prod_{l=1}^L \left(1 + e^{-y_{il}(\mathbf{w}_l^{G^\top}\mathbf{x}_i + b_l^G)}\right)^{-1} \quad (3)\end{aligned}$$

where $\mathbf{w}_l^G = \gamma_l^+\boldsymbol{\mu}_l^+ - \gamma_l^-\boldsymbol{\mu}_l^-$ and $b_l^G = Z_l^+ - Z_l^- + \log\frac{\pi_l}{1-\pi_l}$. Under perfect modelling conditions, and as the training data grows asymptotically, $\mathbf{w}_l^G$ can converge to $\mathbf{w}_l^D$ and b_l^G to b_l^D. As such, sampling the negative training points from $\mathbb{P}(\mathbf{y}|\mathbf{x}; \mathbf{W}^G, \mathbf{b}^G)$ might be almost as effective as sampling from $\mathbb{P}(\mathbf{y}|\mathbf{x}; \mathbf{W}^D, \mathbf{b}^D)$.

Tractability: Given that each training point has $O(\log L)$ positive labels on average (see the dataset statistics in Table 1), the maximum likelihood estimates of $\{(\boldsymbol{\mu}_l^\pm)_{l=1}^L\}$ can be obtained in time $O(ND\log L)$ as $\boldsymbol{\mu}_l^+ = \frac{\sum_{i:y_{il}=1}\mathbf{x}_i}{\|\sum_{i:y_{il}=1}\mathbf{x}_i\|_2}$ and $\boldsymbol{\mu}_l^- = \frac{\boldsymbol{\mu} - \sum_{i:y_{il}=1}\mathbf{x}_i}{\|\boldsymbol{\mu} - \sum_{i:y_{il}=1}\mathbf{x}_i\|_2}$ where $\boldsymbol{\mu} = \sum_{i=1}^N \mathbf{x}_i$. Unfortunately, the concentration parameters $\gamma_l^\pm$ cannot be estimated efficiently [56]. As a result, neither can $\mathbf{w}_l^G$

or b_l^G (since $Z_l^{\pm}$ depend on $\gamma_l^{\pm}$). Furthermore, even if $\mathbf{w}_l^G$ and b_l^G were known, negative sampling by selecting $O(\frac{N}{L}\log L)$ of the negative training points having the highest values of $\mathbb{P}(-y_{il}|\mathbf{x}_i;\mathbf{W}^G,\mathbf{b}^G)$ for each label l would incur $O(NDL)$ cost if done exactly. These problems can be ameliorated based on the observation that $\boldsymbol{\gamma}_l^-$ could be negligible as compared to $\boldsymbol{\gamma}_l^+$ as the millions of negative examples could be far less concentrated than the small number of positive examples. For instance, the positive concentration could be a hundred times larger than the negative concentration on the related searches dataset with 100 million labels. Thus, $\gamma_l^-\boldsymbol{\mu}_l^-$ can be negligible as compared to $\gamma_l^+\boldsymbol{\mu}_l^+$ (as $\|\boldsymbol{\mu}_l^{\pm}\|_2 = 1$) leading to the approximation $\mathbf{w}_l^G = \gamma_l^+\boldsymbol{\mu}_l^+$. This implies that, for a given label l, the negative training points with the largest values of $\mathbb{P}(-y_{il}|\mathbf{x}_i;\mathbf{W}^G,\mathbf{b}^G)$ will be those with the largest values of ${\boldsymbol{\mu}_l^+}^\top\mathbf{x}_i$ and that γ_l^+ and b_l^G do not need to be estimated as they do not affect the ranking of points for the given label. These points will also have the smallest values of $\|\boldsymbol{\mu}_l^+ - \mathbf{x}_i\|_2^2$ as $\frac{1}{2}\|\boldsymbol{\mu}_l^+ - \mathbf{x}_i\|_2^2 = 1 - {\boldsymbol{\mu}_l^+}^\top\mathbf{x}_i$ and can therefore be determined approximately based on ANNS as follows. First, the HNSW algorithm can be used to learn an ANNS data structure over the training points $\mathbf{x}_i$ in time $O(ND\log N)$. Then, the data structure can be queried with each μ_l^+ to determine the negative training examples for label l as $\mathcal{N}_l^{\mathbf{x}} = \{i|1 \le i \le N, y_{il} = -1, \mathbf{x}_i \in \text{ANNS}(\mu_l^+)\}$ in time $O(D\log N)$ per label. This would make negative sampling both accurate and efficient. Unfortunately, this would not address the equally critical problem of reducing prediction time. Low latency and high throughput applications such as related searches need to make predictions in milliseconds per test point and cannot afford the $O(DL)$ cost of applying all L classifiers during prediction.

Negative sampling: Slice therefore speeds up both training and prediction by carrying out ANNS over the generative model parameters $\boldsymbol{\mu}_l^+$ rather than over the training points $\mathbf{x}_i$. The ANNS data structure over $\boldsymbol{\mu}_l^+$ can be learnt in time $O(LD\log L)$. It can then be queried for all N training points $\mathbf{x}_i$ in time $O(ND\log L)$ to determine the set of labels $\mathcal{S}(\mathbf{x}_i) = \{l|1 \le l \le L, \boldsymbol{\mu}_l^+ \in \text{ANNS}(\mathbf{x}_i)\}$ having the largest values of ${\boldsymbol{\mu}_l^+}^\top\mathbf{x}_i$ for point i. Finally, for each label l, the set of negative training examples $\mathcal{N}_l^{\boldsymbol{\mu}}$ can be selected such that $\mathcal{N}_l^{\boldsymbol{\mu}} = \{i|1 \le i \le N, y_{il} = -1, l \in \mathcal{S}(\mathbf{x}_i)\}$. This leads to an overall complexity of $O((N+L)D\log L)$ for determining $\mathcal{N}^{\boldsymbol{\mu}}$ allowing it to be determined more efficiently than $\mathcal{N}^{\mathbf{x}}$. For instance, identifying $\mathcal{N}^{\mathbf{x}}$ could take up to 7x more time and 4x more RAM than identifying $\mathcal{N}^{\boldsymbol{\mu}}$ even on the small scale datasets. Furthermore, training on either set of negative examples was found to be just as effective since $\mathcal{N}^{\boldsymbol{\mu}}$ was found to contain 75.76% of the points having the highest values of $\mathbb{P}(-y_{il}|\mathbf{x}_i;\mathbf{w}_l^D,\mathbf{b}_l^D)$ while $\mathcal{N}^{\mathbf{x}}$ was found to contain 77.77%. In fact, training on $\mathcal{N}^{\boldsymbol{\mu}}$ yielded state-of-the-art results matching the accuracies obtained by training on all N points on almost all datasets and being just 1.5% lower on a single dataset.

Comparison to Parabel and PPDSparse: For low-dimensional dense features, Slice's negative sampling procedure compares favourably with those used in PPDSparse [63] and Parabel [47] which represent the state-of-the-art in extreme classification. Parabel learns a tree over labels such that similar labels end up in the same leaf node. Negative training examples for a label are selected by taking the positive examples of all the other labels present in the same leaf as the given label. The negatives therefore contain samples from the most similar and confusing labels which are likely to get misclassified and are therefore critical for training. Unfortunately, Parabel's tree cannot be learnt accurately in low-dimensions as the linear separator learnt at each internal node does not have enough capacity to ensure that similar labels get partitioned together. Parabel could therefore be up to 15% less accurate than Slice (see Table 2).

PPDSparse, like Slice, learns a separate linear classifier $\mathbf{w}_l$ per label. During each training iteration, $\mathbf{w}_l$ is optimized over all of label l's positive training examples as well as an active set of negative examples. Negative examples are added to the active set at each iteration if they are misclassified or violate the margin according to the current estimate of $\mathbf{w}_l$. This can be done approximately in $O(\hat{N}\hat{D})$ time by evaluating $\mathbf{w}_l^\top\mathbf{x}_i$ on $\hat{D}$ rather than D features and assuming that each feature occurs in only $\hat{N}$ points on average (accessed through an inverted index). Unfortunately, PPDSparse's negative sampling technique has $O(NDL)$ cost for low-dimensional dense features, as $\hat{N} = N$ when features are dense while $\hat{D} \approx D$ when they are low-dimensional (running PPDSparse with the default low value of $\hat{D}$ could reduce accuracy by up to 13%).

Training: Slice learns a linear classifier for each label l by optimizing (1) over the label's positive training points $\mathcal{P}_l = \{i|1 \le i \le N, y_{il} = +1\}$ as well as the negative examples selected for that label $\mathcal{N}_l^{\boldsymbol{\mu}}$. Note that (1) is convex and can be optimized efficiently using Liblinear [15] for all L labels in $O(ND\log L)$ time as $|\mathcal{P}_l| = |\mathcal{N}_l^{\boldsymbol{\mu}}| = O\left(\frac{N}{L}D\log L\right)$ on average. Slice's overall training complexity is therefore $O(ND\log L)$ for estimating $\boldsymbol{\mu}^+$ + $O(LD\log L)$ for constructing the HNSW data structure + $O(ND\log L)$ for selecting $\mathcal{N}^{\boldsymbol{\mu}}$ + $O(ND\log L)$ for optimising (1) = $O(ND\log L)$ in total.

Distributed and parallel implementation: One of the advantages of Slice is that it can not only be learnt efficiently on a single core but that it can also be parallelised across several cores of a single machine or GPU or distributed across multiple machines. Each of the L classifiers in Slice can be learnt in parallel as (1) can be optimised independently for each label. Furthermore, Slice has low network communication costs and memory requirements as only $O(\frac{N}{L}\log L)$ points on average need to be transmitted over the network or loaded into memory for learning a classifier. This is a significant advantage over DiSMEC [3] and PPDSparse [63] which require the entire training data to be transmitted or loaded in memory to train even a single classifier. As a result, unlike DiSMEC and PPDSparse, Slice can be efficiently trained on large problems on GPUs even when the training data does not fit in GPU RAM. All the other parts of Slice's pipeline including estimating $\boldsymbol{\mu}^+$, constructing the HNSW data structure and querying it for selecting $\mathcal{N}^{\boldsymbol{\mu}}$ can be parallelised. For instance, [9] provides details and code for parallel HNSW data structure construction and querying. All these factors put together allow Slice to efficiently scale to datasets with 100 million labels and 240 million training examples.

Efficient prediction: Extreme classification applications, including related searches, require only the labels with the largest values of $\mathbb{P}(y_l|\mathbf{x})$ to be predicted for a test point $\mathbf{x}$. Determining such labels by evaluating $\mathbb{P}(y_l|\mathbf{x})$ for all L labels, as done in DiSMEC, PDSparse and PPDSparse, would incur $O(DL)$ cost making such approaches infeasible for low-latency and high-throughput

applications. Slice addresses this limitation by shortlisting $O(\log L)$ of the most probable labels according to $\mathbb{P}(y_l|\mathbf{x}, \mathbf{w}_l^G, b_l^G)$ and then evaluating $\mathbb{P}(y_l|\mathbf{x}, \mathbf{w}_l^D, b_l^D)$ for these labels alone. Slice approximates $\mathbf{w}_l^G = \gamma \boldsymbol{\mu}_l^+$ (as $\gamma_l^- \boldsymbol{\mu}_l^-$ is negligible) and $b_l^G = b$ so that $\mathbb{P}(y_l|\mathbf{x}, \mathbf{w}_l^G, b_l^G) = \left(1 + e^{-y_l(\gamma \boldsymbol{\mu}_l^{+\top} \mathbf{x}_i + b)}\right)^{-1}$ where γ and b can be learnt through maximum likelihood estimation [45], by validation, *etc.* It was empirically observed that setting $\gamma = \frac{1}{L}\sum_{l=l}^{L} \|\mathbf{w}_l^D\|_2$ and learning b by validation worked slightly better than maximum likelihood estimation. The shortlist of most probable labels can then be obtained as $\mathcal{S}(\mathbf{x}) = \{l | 1 \leq l \leq L, \boldsymbol{\mu}_l^+ \in \text{ANNS}(\mathbf{x})\}$. Note that the shortlist can be obtained efficiently in time $O(D \log L)$ by reusing the ANNS data structure learnt over $\boldsymbol{\mu}_l^+$ during training and that this would not have been possible had the data structure been learnt over the points $\mathbf{x}_i$ instead. The marginal probabilities of the shortlisted labels can then be obtained for both the discriminative and generative models in time $O(D \log L)$. The final label probability is obtained by marginalizing over the discriminative and generative models as it is well recognized that the discriminative model will work better for head labels having lots of training examples while the generative model will work better for tail labels having only a few training examples [40]. Assuming that $\mathbb{P}(\mathbf{w}_l, b_l) = \frac{1}{2}(\delta(\mathbf{w}_l^D, b_l^D) + \delta(\mathbf{w}_l^G, b_l^G))$ yields $\mathbb{P}(y_l|\mathbf{x}) = \int \mathbb{P}(y_l|\mathbf{x}, \mathbf{w}_l, b_l)\mathbb{P}(\mathbf{w}_l, b_l) d\mathbf{w}_l db_l = \frac{1}{2}(\mathbb{P}(y_l|\mathbf{x}, \mathbf{w}_l^D, b_l^D) + \mathbb{P}(y_l|\mathbf{x}, \mathbf{w}_l^G, b_l^G))$. Slice's overall predictions for a test point $\mathbf{x}$ are therefore made in time $O(D \log L)$ by sorting $\mathbb{P}(y_l|\mathbf{x})$ for the shortlisted labels $l \in \mathcal{S}(\mathbf{x})$.

Extension to other loss functions: The arguments presented thus far can be extended to other loss functions that align with the log loss. The MAP estimate of $\mathbf{w}_l$ for a differentiable loss function $\mathcal{L}(y_{il}\mathbf{w}_l^\top \mathbf{x}_i)$ that decomposes over individual labels is given by $\mathbf{w}_l^* = C\sum_{i=1}^{N} \alpha_{il}^* y_{il} \mathbf{x}_i$ where $\alpha_{il}^* = -\nabla_{y_{il}\mathbf{w}_l^{*\top}\mathbf{x}_i} \mathcal{L}(y_{il}\mathbf{w}_l^{*\top}\mathbf{x}_i)$. Slice can be trained on losses for which there is a reasonable overlap between the set of negative points having the largest values of α_{il}^* and $\mathbb{P}(-y_{il}|\mathbf{x}_i; \mathbf{w}_l^D)$. For most classification losses $\mathcal{L}$, this implies that there should be reasonable overlap between $\mathcal{L}$ and the log loss on the set of negative training points that are misclassified or are close to the decision boundary – i. e. points for which $y_{il} = -1$ and $\mathbf{w}_l^\top \mathbf{x}_i$ is large. Such negative training points could be identified efficiently through ANNS as they would have large values of $\boldsymbol{\mu}_l^{+\top}\mathbf{x}_i$ allowing Slice to be trained accurately in logarithmic time. For instance, as shown in Table 2, Slice was found to work particularly well when trained using the hinge loss squared $\mathcal{L} = \max(0, 1 - y_{il}\mathbf{w}_l^\top \mathbf{x}_i)^2$ which is a popular loss for extreme classification [3, 47].

4 EXPERIMENTS

Datasets and features: Experiments were carried out on related searches datasets mined from the Bing click logs. Each input trigger was represented by its 64 dimensional Convolutional Deep Structured Semantic Model (CDSSM) [18, 52] embedding. The labels for each trigger (data point) were chosen to be the set of suggestions that had been clicked in response to the trigger. Results are presented on datasets of varying sizes, ranging from 19 K to 101 M labels, as not all baseline algorithms were able to scale to the larger datasets. Each dataset was randomly partitioned into 80% for training and 20% for testing.

Table 1: Dataset statistics

Dataset	Train N	Features D	Labels L	Test N'	Avg. labels per point	Avg. points per label
EURLex-4K	11,585	1,024	3,956	3,865	5.32	15.59
RS-19K	739,563	64	19,450	184,891	1.01	38.52
Wikipedia-500K	1,646,302	512	501,070	711,542	4.87	16.03
Amazon-670K	490,449	512	670,091	153,025	5.45	3.99
RS-2M	37,178,983	64	1,944,958	12,390,515	1.53	29.26
RS-33M	133,263,449	64	33,104,673	33,311,373	1.44	5.78
RS-101M	240,754,546	64	101,135,892	38,774,604	1.32	3.14

Experiments were also carried out on benchmark extreme multi-label datasets publically available at The Extreme Classification Repository [5] including Amazon-670K [5, 6, 32], Wikipedia-500K [5] and EURLex-4K [5, 34]. The state-of-the-art in deep learning for extreme classification is XML-CNN [29]. Each dataset was therefore represented using low-dimensional, dense XML-CNN features rather than the traditional high-dimensional, sparse bag-of-words features available on The Repository. The XML-CNN features were learnt on the train/test splits used in [29] and were graciously shared by the authors. The experiments in this paper were therefore also carried out on the train/test splits used in [29], rather than those available on The Repository, so as to ensure that there was no leakage of information from the training to the test set.

Table 1 lists the statistics of all the datasets studied in this paper. Most of the experiments on the publically available datasets were carried out with the objective of reproducibly demonstrating that Slice's performance could be significantly better than all other extreme classifiers on low-dimensional, dense features which are more suitable for representing queries than bag-of-words features. Thus, while it is not this paper's objective to obtain state-of-the-art results with bag-of-words features, Table 1 of the supplementary material nevertheless demonstrates that Slice's prediction accuracies can match those of leading extreme classifiers on the Amazon-670K dataset. Finally, note that Slice was found not to be sensitive to the choice of deep learning features and Table 2 of the supplementary material reports the results for GloVE embeddings [44].

Baseline algorithms: Slice was compared to leading 1-vs-All, embedding and tree based extreme classifiers including Parabel [47], PPD-Sparse [63], PD-Sparse [64], DiSMEC [3], , SLEEC [6], LEML [67], WSABIE [60], CPLST [11], CS [17] and PfastreXML [22]. The implementations of all algorithms were provided by their authors apart from CPLST and CS. These algorithms were implemented by us while ensuring that the published results could be reproduced. Results have been reported for only those datasets to which an implementation scales. Results have been reported for Slice and DiSMEC trained on both the log loss (-l) and the hinge loss squared (-s). Results have also been reported for Slice-Generative which is Slice with just the generative model where shortlisted labels are ranked according to $\mathbb{P}(y_l|\mathbf{x}, \mathbf{w}_l^G, b_l^G) \forall l \in \mathcal{S}(x)$. Slice is also compared to kNN-HNSW which is a kNN classifier using HNSW for ANNS. Finally, Slice was compared to six state-of-the-art related searches techniques, referred to as M1-M6, that are currently in production in Bing and span the gamut of traditional approaches ranging from those based on sessions information [7, 8, 13, 43, 51, 55]

Table 2: Results on extreme classification datasets.

Method	P1 (%)	P3 (%)	P5 (%)	Training time (hr)	Test time / point (ms)
EURLex-4K					
Slice-s	**77.72**	**63.78**	**52.05**	0.02	1.23
Slice-l	76.25	63.00	51.44	0.03	1.23
Slice-Generative	66.96	53.95	44.56	**0.002**	**0.10**
kNN-HNSW	76.02	61.91	49.99	0.015	0.24
DiSMEC-s	76.12	62.91	51.51	0.13	4.36
DiSMEC-l	76.04	62.66	51.37	0.25	4.36
Parabel	74.54	61.72	50.48	0.01	0.91
PPD-Sparse	76.32	62.79	51.40	0.013	4.36
PfastreXML	73.63	60.31	49.69	0.037	1.82
SLEEC	74.31	60.00	49.11	0.35	4.87
PD-Sparse	73.53	60.80	49.37	0.12	4.36
CS	58.01	44.85	35.50	0.01	140.10
LEML	60.34	47.45	37.96	0.67	2.24
WSABIE	76.09	61.69	49.11	0.13	2.24
CPLST	61.01	47.43	38.04	0.18	2.24
Amazon-670K					
Slice-s ($\|\mathcal{S}\|$=2000)	**38.10**	**34.00**	**30.92**	6.08	19.59
Slice-l ($\|\mathcal{S}\|$=2000)	37.43	33.05	29.72	7.36	19.59
Slice-s	37.77	33.76	30.70	1.92	3.49
Slice-l	37.66	33.56	30.39	2.11	3.49
Slice-Generative	32.27	29.46	26.90	**1.24**	**0.25**
kNN-HNSW	31.44	28.02	25.49	1.52	0.37
DiSMEC-s	37.60	33.62	30.64	788.84	429
DiSMEC-l	35.11	30.86	27.69	1125.19	429
Parabel	33.93	30.38	27.49	1.54	2.85
PPD-Sparse	33.16	29.60	26.85	3.90	429
PfastreXML	28.51	26.06	24.17	2.85	19.35
SLEEC	18.77	16.50	14.97	7.12	22.54
Wikipedia-500K					
Slice-s ($\|\mathcal{S}\|$=2000)	62.62	41.79	31.57	15.61	11.14
Slice-l ($\|\mathcal{S}\|$=2000)	61.27	40.90	30.89	21.17	11.14
Slice-s	59.89	39.89	30.12	2.34	1.37
Slice-l	59.01	39.51	29.40	3.08	1.37
Slice-Generative	43.10	28.37	21.89	**0.87**	**0.15**
kNN-HNSW	60.20	39.51	19.62	2.56	0.22
DiSMEC-s	**63.70**	**42.49**	**32.26**	2133	316.29
DiSMEC-l	62.11	41.26	31.35	3062.13	316.29
Parabel	59.34	39.05	29.35	6.29	2.94
PPD-Sparse	50.40	33.15	25.54	5.85	316.29
PfastreXML	55.00	36.14	27.38	11.14	6.36

to the query-url/flow/term graphs [4, 33, 59] to query synthesis methods [20, 24] to deep learning techniques.

Hyper-parameters: Slice has three hyper-parameters: (a) the number of shortlisted labels $|\mathcal{S}|$ retrieved by HNSW; (b) the bias parameter b in the generative model and (c) the misclassification penalty C in the loss. The bias b was set by validation while default settings of $|\mathcal{S}| = 300$, $C = 1$ for the hinge loss squared and $C = 10$ for the log loss were used on all datasets. HNSW was also run with the default hyper-parameter settings of $\mathcal{M} = 100$, efC = 300 and efS = $|\mathcal{S}|$. Finally, results have also been reported for $|\mathcal{S}| = 2000$ on Amazon-670K and Wikipedia-500K. The hyper-parameters of the other algorithms were set as suggested by their authors wherever applicable and by fine-grained validation otherwise.

Table 3: Results on small-scale related searches datasets.

Method	P1 (%)	P3 (%)	P5 (%)	Training time (hr)	Test time / point (ms)
RS-19K					
Slice-s	**77.94**	29.69	18.28	0.18	0.44
Slice-l	77.70	29.82	18.35	0.44	0.44
Slice-Generative	70.01	27.73	17.30	**0.08**	**0.03**
kNN-HNSW	74.69	29.13	17.76	0.55	0.06
DiSMEC-s	77.52	29.51	18.16	11.46	2.99
DiSMEC-l	77.90	**29.88**	**18.41**	41.74	2.99
Parabel	62.61	26.28	16.63	0.50	1.16
PPD-Sparse	69.54	28.38	17.74	0.19	2.99
PfastreXML	74.05	28.40	17.52	0.76	1.18
SLEEC	38.08	21.10	14.48	2.17	10.65
RS-2M					
Slice-s	**25.59**	**17.11**	**13.14**	12.97	0.47
Slice-l	23.68	16.42	12.84	29.67	0.47
Slice-Generative	19.97	13.98	11.06	**4.33**	**0.05**
kNN-HNSW	23.26	16.03	12.49	48.48	0.14
Parabel	16.25	11.94	9.7	38.76	1.00
PPD-Sparse	12.40	9.00	7.58	167.96	380.9
PfastreXML	21.10	14.06	10.82	163.86	1.17

Evaluation metrics: Performance was evaluated using precision@k and nDCG@k (with $k = 1, 3$ and 5) which are widely used metrics [1, 3, 6, 29, 48, 63, 64, 68] for extreme classification and which are defined in the supplementary material. Performance has also been evaluated in the supplementary material using propensity scored precision@k which has recently been shown to be an unbiased, and more suitable, metric [22]. The propensity model and values available on The Repository were used. In addition, the training and prediction times of various algorithms have also been benchmarked on a single core of a 16 core Intel Xeon 2.3 GHz machine.

Results on small datasets: Tables 2 and 3 benchmark Slice's performance on datasets where many state-of-the-art extreme classifiers could scale. Slice's prediction accuracy could be up to 13% higher than PPDSparse's and 15% higher than Parabel's. This validates Slice's approximations and shows that Slice's negative sampling technique could be significantly more accurate than those of Parabel and PPDSparse for low-dimensional, dense features. In fact, Slice's negative sampling technique could be just as effective as training on all the negative points as Slice's prediction accuracy could match DiSMEC's (except on Wikipedia-500K). At the same time, Slice's negative sampling technique and label shortlisting procedure allowed it to be up to 10x and 500x faster at training and up to 100x faster at prediction than PPDSparse and DiSMEC respectively. The results also demonstrate the following about Slice. First, while the proposed negative sampling technique was developed for the log loss (Slice-l), it generalizes well to other losses such as the hinge loss squared (Slice-s) that align with the log loss. Second, the importance of the proposed discriminative model can be judged from the fact that it could boost prediction accuracy by as much as 20% when added to the generative model (Slice-Generative). Third, Slice could be up to 8% more accurate as compared to kNN-HNSW even though both rely internally on HNSW based ANNS to cut costs from linear to logarithmic. Both these points highlight the advantage of Slice's proposed model over kNN classification.

Table 4: Slice's precision@k, nDCG@k, training time and prediction time on the large-scale related searches datasets.

Dataset	Method	P1=N1 (%)	P3 (%)	P5 (%)	N3 (%)	N5 (%)	Training time (min)			Test time/point (ms)		Model size (GB)
							T_{HNSW}	$T_{\mathcal{N}^{\mu}}$	$T_{\mathbf{w}^D}$	$t_{\mathcal{S}}$	$t_{\mathbf{y}}$	
RS-33M	Slice-Generative	11.80	8.82	7.19	16.7	19.4	150	-	-	0.84	-	35.30
	Slice-s	16.99	11.72	9.13	22.19	14.96	150	19.23	29.27	2.5	0.44	43.77
RS-101M	Slice-Generative	2.45	3.11	3.07	6.39	9.08	425	-	-	1.26	-	77.31
	Slice-s	5.08	4.94	4.40	10.90	13.96	425	40.05	84.10	2.95	1.05	103.17

Table 5: Slice makes significantly more accurate predictions as compared to leading related searches algorithms in Bing.

Dataset	Method	P1=N1 (%)	P3 (%)	P5 (%)	N3 (%)	N5 (%)
RS-2M	M1	42.14	27.44	20.66	40.29	40.17
	M2	38.15	27.35	21.87	38.16	39.35
	M3	22.62	17.84	14.75	25.46	27.44
	M4	23.09	12.96	8.85	19.98	19.05
	M5	15.05	7.59	4.93	11.90	11.00
	M6	12.73	7.04	4.54	11.20	10.57
	Slice-s	**43.08**	**31.10**	**24.81**	**45.40**	**47.60**
RS-33M	M1	35.26	24.43	18.89	32.53	31.89
	M2	25.44	21.07	18.06	26.99	28.83
	M3	16.18	13.72	12.02	17.35	18.73
	M4	31.30	22.19	16.86	29.76	29.34
	M5	21.73	13.57	9.69	18.60	17.55
	M6	16.34	14.63	11.69	18.94	19.77
	Slice-s	**39.21**	**29.67**	**24.15**	**39.08**	**40.19**

Performance on tail labels: Table 3 in the supplementary material shows that Slice's performance gains arose from the accurate prediction of many tail labels and not just a few head labels. Slice outperformed all other extreme classifiers according to propensity scored precision (PSP) which is a more suitable loss function [22] for extreme classification and recommendation as it is unbiased with respect to missing labels in the ground truth and assigns greater rewards for the accurate prediction of tail labels. Slice's PSP could be up to 3% and 4% higher than any other classifier's (including DiSMEC's) on Amazon-670K and Wikipedia-500K respectively. Figure 1 in the supplementary material also shows that Slice's (vanilla) precision per tail label could be higher than DiSMEC's.

Results on large datasets: Table 4 presents Slice-s and Slice-Generative's results on the RS-33M and RS-101M datasets which are well beyond the scaling capabilities of all other extreme classifiers. Slice could be trained efficiently on RS-101M in about 9 hours in a distributed fashion as discussed in Section 3 and could make predictions in under 5 milliseconds per test point. HNSW was trained (T_{HNSW}) in about 7 hours on 32 cores of an Intel Xeon 2.3 GHz machine. The negative training examples were identified ($T_{\mathcal{N}^{\mu}}$) and all 101 million discriminative classifiers were learnt ($T_{\mathbf{w}^D}$) in 40 and 84 minutes respectively on 10 cores each of 100 Intel Xeon 2.3 GHz machines. During prediction, the set of shortlisted labels for a given test point ($t_{\mathcal{S}}$) were determined in 2.95 milliseconds while the final predictions ($t_{\mathbf{y}}$) were made in another 1.95 milliseconds.

Table 6: Slice's coverage@k on the RS-101M dataset.

C1 (%)	C3 (%)	C5 (%)	C10 (%)	C15 (%)	C20 (%)
24.70	45.25	59.41	82.11	94.42	99.31

The importance of the proposed discriminative model can again be judged from the fact that adding it to the generative model yielded relative improvements of 107%, 59% and 43% in precision@1, 3 and 5 respectively. Further inspection of Slice's top 5 and top 15 predictions per point on the RS-101M dataset revealed that they covered more than 59 and 95 million unique labels respectively (see Table 6). Almost all of the 101 million labels were covered in the top 20 predictions per point. Slice's strong performance is therefore achieved not by recommending just a few frequently occurring, easy to predict, head labels but by recommending rare, hard to predict, tail labels. Scaling to large datasets is therefore critical for making recommendations from diverse suggestions for tail triggers.

Comparison to related searches methods: Table 5 compares Slice's offline prediction accuracy to that of leading related searches methods anonymized as M1-M6. Note that the accuracy of some of these methods might be low as they were unable to make recommendations for previously unseen triggers. Such triggers were therefore excluded from the test sets and only those triggers for which all methods M1-M6 could recommend at least one suggestion were retained. This allows the comparison to focus on recommendation accuracy without worrying about trigger coverage. Unfortunately, the prediction accuracies of M1-M6 were still low as reported in Table 4 in the supplementary material. Many irrelevant recommendations could be eliminated by restricting their predictions to the set of 2 or 33 million labels that Slice was trained on. Even in this restricted setting, Table 5 shows that Slice's accuracy could be up to 8% higher than all other related searches methods.

Table 7: Relative improvements in online metrics when Slice was added to the ensemble serving related searches on Bing.

Trigger coverage (%)	Suggestion density (%)	CCR (%)	Success rate (%)	Quick back rate (%)
52.01	33.0	2.88	2.62	-0.93

Live deployment on Bing: Table 7 reports the relative improvements in online metrics when Slice was added to the related searches ensemble in production in Bing. Slice was able to increase the number of triggers for which at least 8 suggestions were recommended

Table 8: Slice's accuracy does not vary much on Amazon-670K if HNSW was replaced by LSH or exact NN search.

	Exact NN	HNSW	LSH	Random
Precision@1 (%)	37.85	37.77	35.21	24.67
Precision@3 (%)	33.83	33.76	32.22	22.32
Precision@5 (%)	30.73	30.70	29.53	20.41
Training time (hr)	58.79	1.92	1.93	0.47
Test time/point (ms)	429	3.49	11.55	429

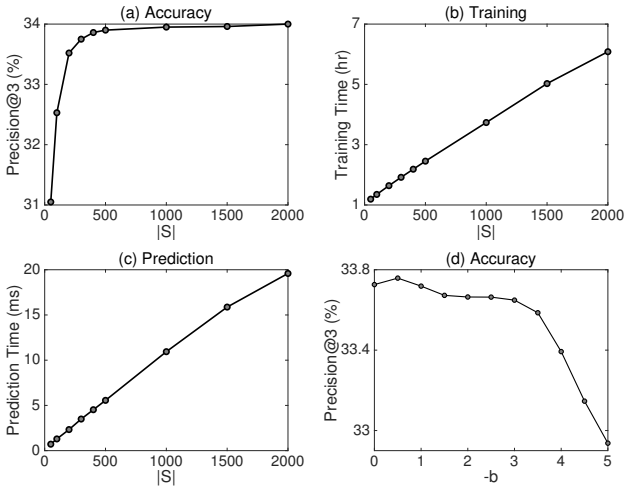

Figure 1: The variation in Slice's (a) accuracy, (b) training time & (c) prediction time on Amazon-670K with the size of the label shortlist $|\mathcal{S}|$. The variation in Slice's accuracy with the bias parameter b is shown in (d).

(trigger coverage) by 52%. Slice also increased the number of suggestions being recommended per trigger (suggestion density) by 33%. A relative improvement of 2.88% was observed in the conditional click-through rate (CCR) defined as the ratio of the total number of clicks on suggestions to the total number of pages on which suggestions were displayed. At the same time, the number of triggers for which the user clicked on an irrelevant suggestion and then quickly hit the back button on the browser (quick back rate) was reduced by 0.93%. Finally, Slice was found to improve task-completion with a 2.62% increase in the overall success rate defined as the ratio of the total number of related searches clicks which led to a subsequent organic search click with long dwell time to the total number of pages on which suggestions were displayed. The success rate for the most infrequently occurring triggers went up by 12.62%. This is particularly noteworthy as almost all traditional related searches methods fail in the tail and getting performance improvements for tail triggers is known to be notoriously hard. These results demonstrate Slice's effectiveness and the advantages of reformulating related searches as an extreme classification task.

Qualitative results: Table 5 in the supplementary material compares Slice's suggestions for a few triggers to those recommended by the ensemble currently in production in Bing. Slice recommended more suggestions than Bing and with greater relevance and diversity. For instance, as already discussed in the Introduction, Bing recommended the single irrelevant, but lexically similar, suggestion "cam newton shoulder surgery" in response to the trigger "cam procedure shoulder". On the other hand, all of Slice's recommendations were relevant for a user looking to learn more about the Comprehensive Arthroscopic Management (CAM) shoulder surgery procedure. Slice's suggestions included "shoulder surgery procedures", "cost of arthroscopic shoulder surgery", "shoulder replacement surgery success rate", "recovery from arthroscopic shoulder surgery", *etc.* which covered diverse facets of the trigger such as surgery cost, recovery expectations, alternatives and their success rates. Similar trends were observed for other triggers.

Ablations: Table 8 shows that Slice's accuracy was not very sensitive to the choice of the ANNS method as it did not vary significantly if HNSW was replaced by Latent Semantic Hashing (LSH) [2] or even exact NN search. This validates the use of ANNS for low-dimensional dense features. Slice therefore relies on HNSW as it speeded up training and prediction by 30x and 123x respectively over exact NN search and also speeded up prediction by 3x over LSH. Table 8 also reconfirms the importance of Slice's negative sampling procedure as replacing it by random sampling reduced accuracy by more than 10%. Figure 1 (a)-(c) shows that $|\mathcal{S}| = 300$ leads to a good trade-off between accuracy, training time and prediction time as precision@3 saturates soon after that while training and prediction time continue to grow linearly. Finally, Figure 1 (d) shows that Slice's precision@3 varied by less than a percent with the bias hyper-parameter and was therefore robust to the choice of b.

5 CONCLUSIONS

This paper reformulated the problem of recommending related queries on a search engine as an extreme classification task. It developed the Slice algorithm specifically designed for low-dimensional, dense, deep learning features popularly used to represent queries. Slice cut down training and prediction time from linear to logarithmic in the number of labels based on a novel negative sub-sampling technique. The proposed technique was demonstrated to be just as effective as training on all the negative examples and significantly more accurate than those proposed in Parabel and PPDSparse. This allowed Slice to match DiSMEC's accuracies while being many orders of magnitude faster at training and prediction. It also allowed Slice to efficiently train on related searches datasets having 240 million points and 100 million labels which are well beyond the scaling capabilities of all other extreme classifiers.

The proposed reformulation was shown to alleviate some of the problems of traditional related searches methods in recommending suggestions for tail triggers. When added to the ensemble of leading related searches algorithms currently in production in Bing, Slice led to a relative increase of 52% in the trigger coverage, 33% in the suggestion density, 2.88% in the conditional click-through rate, 2.62% in the overall success rate and a decrease of 0.93% in the quick back rate. The gains for tail triggers were particularly noteworthy as a 12.62% improvement was observed in the success rate.

ACKNOWLEDGEMENTS

We are grateful to Manoj Agrawal, Kunal Dahiya, Prateek Jain and Yashoteja Prabhu for helpful discussions and feedback. Himanshu Jain is supported by a Google PhD Fellowship at IIT Delhi.

REFERENCES

[1] R. Agrawal, A. Gupta, Y. Prabhu, and M. Varma. 2013. Multi-label Learning with Millions of Labels: Recommending Advertiser Bid Phrases for Web Pages. In *WWW*.

[2] A. Andoni, P. Indyk, T. Laarhoven, I. Razenshteyn, and L. Schmidt. 2015. Practical and optimal LSH for angular distance. In *NIPS*.

[3] R. Babbar and B. Shoelkopf. 2017. DiSMEC-Distributed Sparse Machines for Extreme Multi-label Classification. In *WSDM*.

[4] R. Baeza-Yates, C. Hurtado, and M. Mendoza. 2004. Query recommendation using query logs in search engines. In *ICDT*.

[5] K. Bhatia, K. Dahiya, H. Jain, Y. Prabhu, and M. Varma. 2014. The Extreme Classification Repository. http://manikvarma.org/downloads/XC/XMLRepository.html.

[6] K. Bhatia, H. Jain, P. Kar, M. Varma, and P. Jain. 2015. Sparse Local Embeddings for Extreme Multi-label Classification. In *NIPS*.

[7] P. Boldi, F. Bonchi, C. Castillo, D. Donato, A. Gionis, and S. Vigna. 2008. The query-flow graph: model and applications. In *CIKM*.

[8] F. Bonchi, R. Perego, F. Silvestri, H. Vahabi, and R. Venturini. 2012. Efficient query recommendations in the long tail via center-piece subgraphs. In *SIGIR*.

[9] L. Boytsov. 2016. Code for HNSW. https://github.com/searchivarius/nmslib.

[10] L. Cayton. 2008. Fast nearest neighbor retrieval for bregman divergences. In *ICML*.

[11] Y. N. Chen and H. T. Lin. 2012. Feature-aware Label Space Dimension Reduction for Multi-label Classification. In *NIPS*.

[12] M. Cissé, N. Usunier, T. Artières, and P. Gallinari. 2013. Robust Bloom Filters for Large MultiLabel Classification Tasks. In *NIPS*.

[13] M. Dehghani, S. Rothe, E. Alfonseca, and P. Fleury. 2017. Learning to attend, copy, and generate for session-based query suggestion. In *CIKM*.

[14] F. Diaz, B. Mitra, and N. Craswell. 2016. Query expansion with locally-trained word embeddings. *CoRR* (2016). https://arxiv.org/abs/1605.07891

[15] R. E. Fan, K. W. Chang, C. J. Hsieh, X. R. Wang, and C. J. Lin. 2008. LIBLINEAR: A library for large linear classification. *JMLR* (2008).

[16] H. B. Hashemi, A. Asiaee, and R. Kraft. 2016. Query intent detection using convolutional neural networks. In *WSDM*.

[17] D. Hsu, S. Kakade, J. Langford, and T. Zhang. 2009. Multi-Label Prediction via Compressed Sensing. In *NIPS*.

[18] P. S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *CIKM*.

[19] P. Indyk and R. Motwani. 1998. Approximate nearest neighbors: towards removing the curse of dimensionality.

[20] A. Jain, U. Ozertem, and E. Velipasaoglu. 2011. Synthesizing high utility suggestions for rare web search queries. In *SIGIR*.

[21] H. Jain, V. Balasubramanian, B. Chunduri, and M. Varma. 2019. Code for Slice. http://manikvarma.org/code/Slice/download.html.

[22] H. Jain, Y. Prabhu, and M. Varma. 2016. Extreme Multi-label Loss Functions for Recommendation, Tagging, Ranking & Other Missing Label Applications. In *KDD*.

[23] K. Jasinska, K. Dembczynski, R. Busa-Fekete, K. Pfannschmidt, T. Klerx, and E. Hüllermeier. 2016. Extreme F-measure Maximization Using Sparse Probability Estimates. In *ICML*. 1435–1444.

[24] R. Jones, B. Rey, O. Madani, and W. Greiner. 2006. Generating query substitutions. In *WWW*.

[25] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov. 2017. Bag of tricks for efficient text classification. In *EACL*.

[26] E. Kushilevitz, R. Ostrovsky, and Y. Rabani. 1998. Approximate nearest neighbors: towards removing the curse of dimensionality.

[27] W. Li, Y. Zhang, Y. Sun, W. Wang, W. Zhang, and X. Lin. 2016. Approximate Nearest Neighbor Search on High Dimensional Data—Experiments, Analyses, and Improvement. *CoRR* (2016). https://arxiv.org/pdf/1610.02455.pdf

[28] Z. Lin, G. Ding, M. Hu, and J. Wang. 2014. Multi-label Classification via Feature-aware Implicit Label Space Encoding. In *ICML*.

[29] J. Liu, W. C. Chang, Y. Wu, and Y. Yang. 2017. Deep Learning for Extreme Multi-label Text Classification. In *SIGIR*.

[30] Z. Lu, B. Savas, W. Tang, and I. S. Dhillon. 2010. Supervised link prediction using multiple sources.

[31] Y. Malkov and D. A. Yashunin. 2016. Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. *CoRR* (2016).

[32] J. McAuley and J. Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *RecSys*.

[33] Q. Mei, D. Zhou, and K. Church. 2008. Query Suggestion Using Hitting Time. In *CIKM*.

[34] E. L. Mencia and J. Fürnkranz. 2008. Efficient pairwise multilabel classification for large-scale problems in the legal domain. In *SIGIR*.

[35] A. K. Menon and C. Elkan. 2011. Link prediction via matrix factorization. In *ECML-PKDD*.

[36] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

[37] P. Mineiro and N. Karampatziakis. 2015. Fast Label Embeddings for Extremely Large Output Spaces. In *ECML*.

[38] J. P. Mordelet, F. amd Vert. 2014. A bagging SVM to learn from positive and unlabeled examples. (2014).

[39] G. Navarro. 2002. Searching in metric spaces by spatial approximation.

[40] A. Ng and M. Jordan. 2002. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes.. In *NIPS*.

[41] A. Niculescu-Mizil and E. Abbasnejad. 2017. Label Filters for Large Scale Multilabel Classification. In *AISTATS*.

[42] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell. 2000. Text classification from labeled and unlabeled documents using EM. (2000).

[43] U. Ozertem, O. Chapelle, P. Donmez, and E. Velipasaoglu. 2012. Learning to suggest: a machine learning framework for ranking query suggestions. In *SIGIR*.

[44] J. Pennington, R. Socher, and C. Manning. 2014. Glove: Global vectors for word representation.

[45] J. Platt. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in large margin classifiers*.

[46] Y. Prabhu, A. Kag, S. Gopinath, K. Dahia, S. Harsola, R. Agrawal, and M. Varma. 2018. Extreme multi-label learning with label features for warm-start tagging, ranking and recommendation. In *WSDM*.

[47] Y. Prabhu, A. Kag, S. Harsola, R. Agrawal, and M. Varma. 2018. Parabel: Partitioned Label Trees for Extreme Classification with Application to Dynamic Search Advertising. In *WWW*.

[48] Y. Prabhu and M. Varma. 2014. FastXML: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *KDD*.

[49] R. Ramanath, G. Polatkan, L. Xu, H. Lee, B. Hu, and S. Zhou. 2018. Deploying Deep Ranking Models for Search Verticals. *CoRR* (2018). https://arxiv.org/abs/1806.02281

[50] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback.. In *AUAI*.

[51] E. Sadikov, J. Madhavan, L. Wang, and A. Halevy. 2010. Clustering query refinements by user intent. In *WWW*.

[52] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil. 2014. Learning semantic representations using convolutional neural networks for web search. In *WWW*.

[53] S. Si, H. Zhang, S. S. Keerthi, D. Mahajan, I. S. Dhillon, and C. J. Hsieh. 2017. Gradient Boosted Decision Trees for High Dimensional Sparse Output. In *ICML*. 3182–3190.

[54] W. Siblini, F. Meyer, and P. Kuntz. 2018. CRAFTML, an Efficient Clustering-based Random Forest for Extreme Multi-label Learning. In International Conference on Machine Learning. In *ICML*.

[55] A. Sordoni, Y. Bengio, H. Vahabi, C. Lioma, J. Grue Simonsen, and J. Y. Nie. 2015. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *CIKM*.

[56] S. Sra. 2012. A short note on parameter approximation for von Mises-Fisher distributions: and a fast implementation of Is(x). (2012).

[57] Y. Tagami. 2017. AnnexML: Approximate Nearest Neighbor Search for Extreme Multi-label Classification. In *KDD*.

[58] J. Uhlmann. 1991. Satisfying general proximity similarity queries with metric trees. (1991).

[59] H. Vahabi, M. Ackerman, D. Loker, R. Baeza-Yates, and A. Lopez-Ortiz. 2013. Orthogonal query recommendation. In *RecSys*.

[60] J. Weston, S. Bengio, and N. Usunier. 2011. Wsabie: Scaling Up To Large Vocabulary Image Annotation. In *IJCAI*.

[61] J. Weston, A. Makadia, and H. Yee. 2013. Label Partitioning For Sublinear Ranking. In *ICML*.

[62] S. H. Yang, B. Long, A. Smola, N. Sadagopan, Z. Zheng, and H. Zha. 2011. Like like alike: joint friendship and interest propagation in social networks.. In *WWW*.

[63] I. E. H. Yen, X. Huang, W. Dai, P. Ravikumar, I. Dhillon, and E. Xing. 2017. PPDsparse: A Parallel Primal-Dual Sparse Method for Extreme Classification. In *KDD*. 545–553.

[64] I. E. H. Yen, X. Huang, P. Ravikumar, K. Zhong, and I. S. Dhillon. 2016. PD-Sparse: A primal and dual sparse approach to extreme multiclass and multilabel classification. In *ICML*.

[65] I. E. H. Yen, S. Kale, F. Yu, D. Holtmann-Rice, S. Kumar, and P. Ravikumar. 2018. Loss Decomposition for Fast Learning in Large Output Spaces. In International Conference on Machine Learning. In *ICML*.

[66] P. N. Yianilos. 1993. Data structures and algorithms for nearest neighbor search in general metric spaces.

[67] H. F. Yu, P. Jain, P. Kar, and I. S. Dhillon. 2014. Large-scale Multi-label Learning with Missing Labels. In *ICML*.

[68] W. Zhang, L. Wang, J. Yan, X. Wang, and H. Zha. 2017. Deep Extreme Multi-label Learning. *CoRR* (2017).

[69] Y. Zhang and J. G. Schneider. 2011. Multi-Label Output Codes using Canonical Correlation Analysis. In *AISTATS*.