Extreme Classification in Log Memory using Count-Min Sketch: A Case Study of Amazon Search with 50M Products

Tharun Medini[1], Qixuan Huang[1], Yiqiu Wang[2], Vijai Mohan[3], Anshumali Shrivastava[1]

[1]Rice University, [2]MIT, [3]Amazon Search

Problem Statement

Predict semantically related products among millions in Amazon Product Catalog for a given query to maximize the purchase likelihood; using classification instead of regression

- Currently, Deep Semantic Search Model (DSSM) works on the principle of projecting queries and product titles to a small dimensional subspace (by embedding the sentence tokens in to 256-d vectors).
- These embeddings can be used to extract closest neighbors for each query
- Our Hypothesis: Learning a classifier with each product as a separate class is better than learning token embeddings.
- Why so?: Classifier intrinsically imposes all the irrelevant products as negatives. For embedding, we need to sample negatives manually. Cross-entropy loss is more structured than any regression loss.
- Also: Learning embeddings needs pairwise training which is very slow compared to having one sample per query.

Challenges:

- 256-d penultimate layer to 50M final layer takes 12.8 billion parameters.
- **Modest feed-forward network cannot be trained on an expensive p3.16x machine! We need smart Extreme Classification methods**

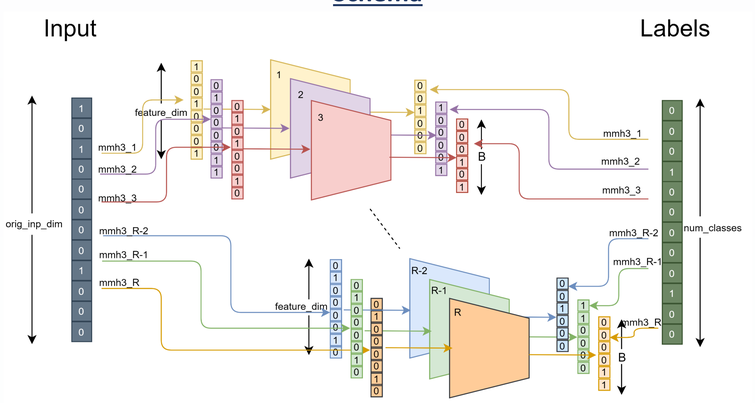
Our Proposal: Merged Average Classifiers via Hashing (MACH)

- Assign all classes to a small number of buckets B using a 2-universal random hash function
- Reduce large output classification to small output space problem
- Using only R = O(logK) such classifiers, MACH can discriminate any pair of classes with high probability. (K = |ASINs|)
- We predict R B-dimensional vectors using parallel and independent models where R*B << K. Hence
- Theoretically, we can prove that for multiclass, the following holds

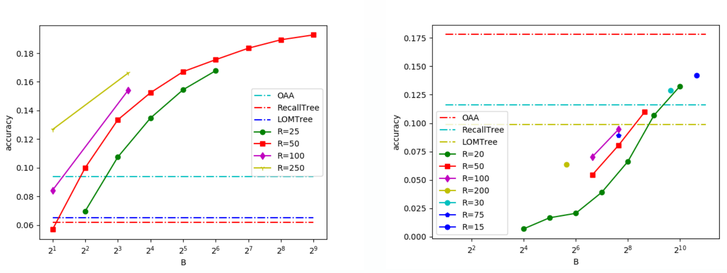
$$p_i = \frac{B}{B-1}\left[\frac{1}{R}\sum_{j=1}^{R} P^{j}_{h_j(i)}(x) - \frac{1}{B}\right]$$

- Order or predictions is preserved in expectation by the sum of $P_{hj}^{j}(x)$

Schema



Multiclass Datasets



Accuracy-Resource tradeoff with MACH with varying settings of R and B.
Left: ODP Dataset. Right: Imagenet Dataset

Multilabel Datasets

Dataset	Precision@K	MACH	Parabel	DisMEC	PfastreXML	FastXML
Wiki10-31K	P@1	**0.8544**	0.8431	0.8520	0.8357	0.8303
	P@3	0.7142	0.7257	**0.7460**	0.6861	0.6747
	P@5	0.6151	0.6339	**0.6590**	0.5910	0.5776
Delicious-200K	P@1	0.4366	**0.4697**	0.4550	0.4172	0.4307
	P@3	**0.4018**	0.4008	0.3870	0.3783	0.3866
	P@5	**0.3816**	0.3663	0.3550	0.3558	0.3619
Amazon-670K	P@1	0.4141	**0.4489**	0.4470	0.3946	0.3699
	P@3	**0.3971**	**0.3980**	**0.3970**	0.3581	0.3328
	P@5	**0.3632**	0.3600	0.3610	0.3305	0.3053

Comparison of MACH and popular extreme classification algorithms on few public datasets. We notice that MACH mostly preserves the precision and slightly betters the best algorithms on half of the cases. These numbers also establish the limitations of pure tree based approaches FastXML and PfastreXML

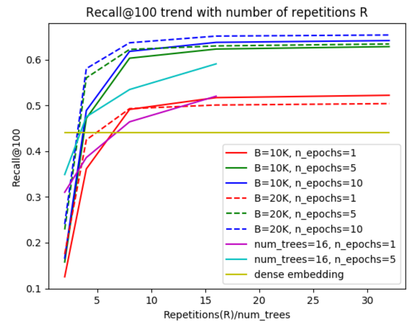
EKTPP-50M (Electronics, Kitchen, Toys & Games, PC Hardware and Photo)

Model	Epochs	wRecall@100	Training time	Peak Memory-Training	Peak Memory-Eval
DSSM – 256d	5	0.441	316.6 hrs	40 GB	286 GB
Parabel, 16 trees	5	0.5810	232.4 hrs	350 GB	426 GB
MACH, B=10K, R=32	10	0.6419	31.8 hrs	150 GB	80 GB
MACH, B=20K, R=32	10	0.6541	34.2 hrs	180 GB	90 GB

Softlines-85M

Model	Epochs	wRecall@100	Training time	Peak Memory-Training	Peak Memory-Eval
DSSM – 256d	5	0.2489	160 hrs	40 GB	486 GB
Parabel, 16 trees	5	0.2753	119 hrs	-	-
MACH, B=20K, R=32	10	0.3216	15.83 hrs	200 GB	100 GB
MACH, B=40K, R=32	10	0.3384	16.67 hrs	233.3 GB	116.67 GB

Recall Trend (EKTPP-50M)



Matching Metrics (EKTPP-50M)

Metric	Embedding	Parabel	MACH, B=10K, R=32	MACH, B=20K, R=32
map_weighted	0.6419	0.6335	0.6864	**0.7081**
map_unweighted	0.4802	**0.5210**	0.4913	0.5182
mrr_weighted	0.4439	**0.5596**	0.5393	0.5307
mrr_unweighted	0.4658	**0.5066**	0.4765	0.5015
ndcg_weighted	0.7792	0.7567	0.7211	**0.7830**
ndcg_unweighted	0.5925	0.6058	0.5828	**0.6081**
recall_weighted	0.8391	0.7509	0.8344	**0.8486**
recall_unweighted	**0.8968**	0.7717	0.7883	0.8206

Ranking Metrics (EKTPP-50M)

Metric	Embedding	Parabel	MACH, B=10K, R=32	MACH, B=20K, R=32
ndcg_weighted	0.7456	0.7374	0.7769	0.7749
ndcg_unweighted	0.6076	0.6167	0.6072	0.6144
mrr_weighted	0.9196	0.9180	0.9414	0.9419
mrr_unweighted	0.5160	0.5200	0.5200	0.5293
mrr_most_rel_weighted	0.5091	0.5037	0.5146	0.5108
mrr_most_rel_unweighted	0.4671	0.4693	0.4681	0.4767
prec@1_weighted	0.8744	0.8788	0.9109	0.9102
prec@1_unweighted	0.3521	0.3573	0.3667	0.3702
prec@1_most_rel_weighted	0.3776	0.3741	0.3989	0.3989
prec@1_most_rel_unweighted	0.3246	0.3221	0.3365	0.3460

References

[1] Medini, Tharun, et al. " Simultaneous Matching and Ranking as end-to-end Deep Classification: A Case study of Information Retrieval with 50M Documents", NeurIPS 2019 .

[2] Nigam, Priyanka, et al. "Semantic Product Search“ , *KDD 2019* .

Contact

Tharun Medini: tharun.medini@rice.edu

Anshumali Shrivastava: anshumali@rice.edu

RUSH-LAB: rush.rice.edu