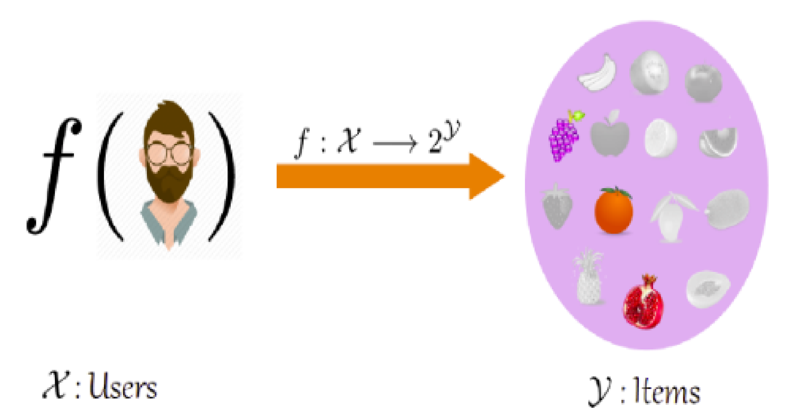
Distributional Semantics meets Multi Label Learning

Vivek Gupta (1,2), Rahul Wadbude (3), Nagarajan Natarajan (2), Harish Karnick (3), Prateek Jain (2) and Piyush Rai(3)

(1) School of Computing, University of Utah, (2) Microsoft Research Lab, India, (3) Indian Institute of Technology, Kanpur

Extreme Multi-Label Learning

- Learning with millions of labels
- Learning with heavy tail distribution of labels
- Learning with missing labels
- Learning to promote diverse recommendations



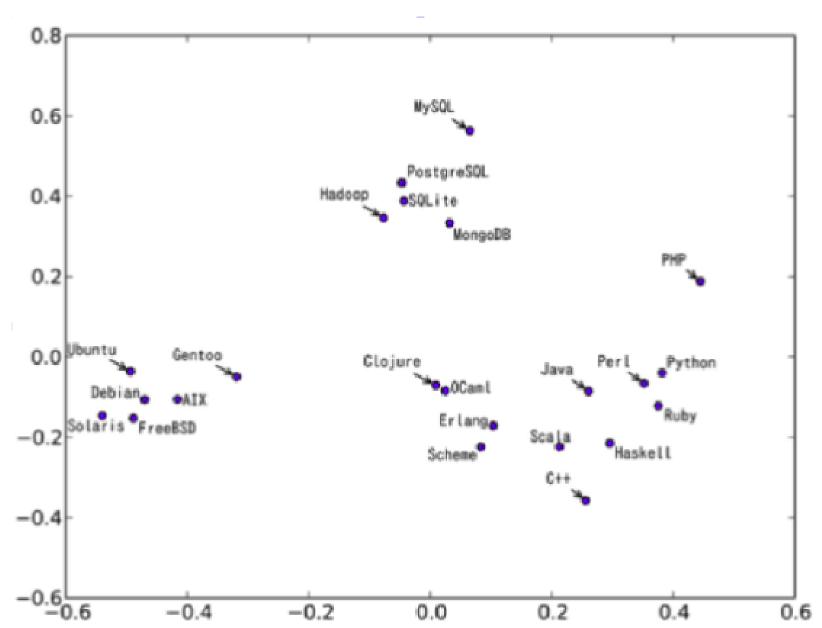
Methods for Extreme Learning

- Tree based : split examples by labels
- Embedding based : embed labels or examples
- One-vs-all: one classifier per label

Method	Accuracy	Scalable	Predict Cost	Model Size	Theory
1-vs-All	☺	☹	☹	☹	☺
Embedding	☺	☺	☺	☺	☺
Tree	☺	☺	☺	☹	☹

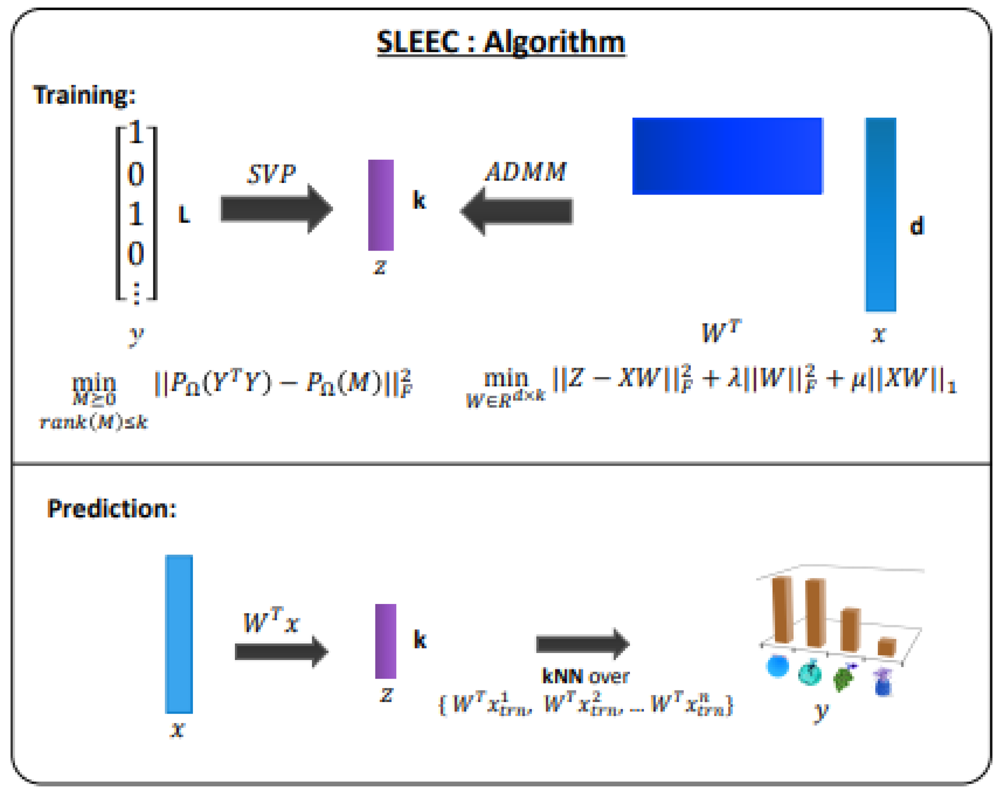
Distributional Semantics

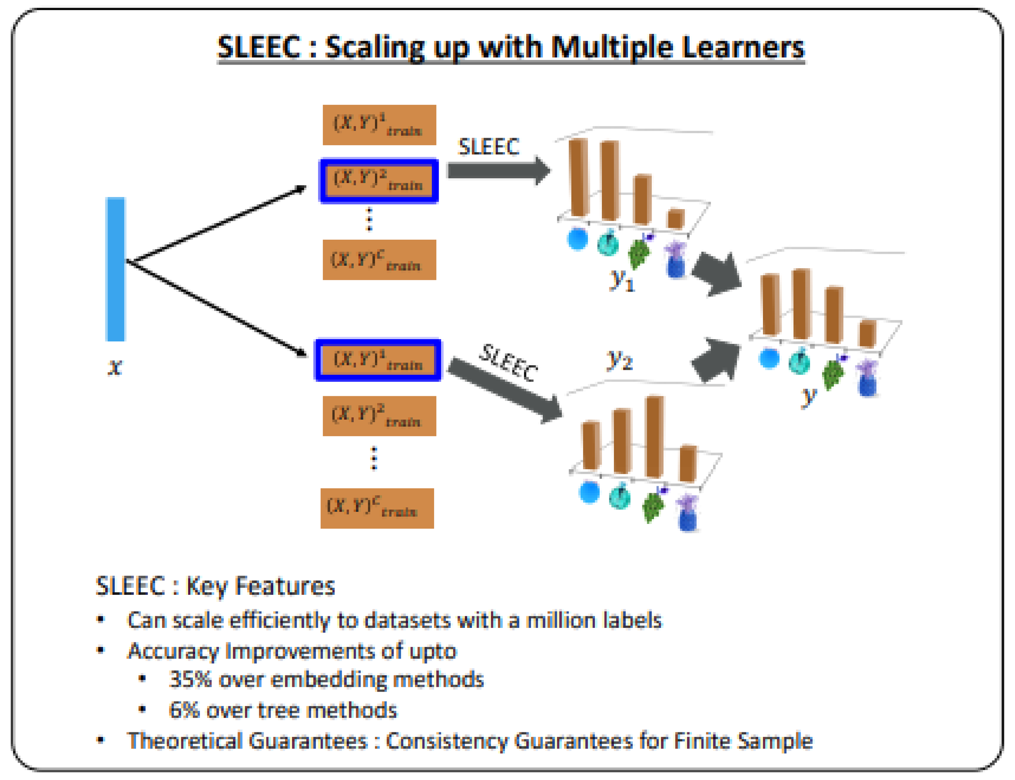
- Each word (w) or sentence (s) is represented using a vector $\vec{v} \in \mathrm{R}^d$
- Semantically similar words or sentences occur closer in the vector space



- Various methods word2vec (SGNS, CBOW) and Doc2vec (PV-DM, PV- DBOW) by Mikolov et al.

SLEEC: Embedding based Algorithm





SGNS meets Label Embedding

- $\mathbb{S}^i = \{j; j \in NN_i\}_{j=1}^K$, here NN_i denote nearest neighbour of y_i
- $K_{ij} = cos(z_i, z_j) = \frac{z_i . z_j}{\|z_i\|\|z_j\|}$, where z_i,z_j label embedding of y_i, y_j
- $z_i = Vx_i$, where $V \in \mathbb{R}^{l \times D}$

Optimization Objective

$$P_i(j \in S^i) = \sigma(\gamma K_{ij})$$

$$\mathbb{J}_i = \sum_{j \in S_i} \log(P_i(j \in S^i)) + \sum_{k \notin S_i}^{K} \log(P_i(k \notin S^i))$$

$$\mathbb{J}_i = \sum_{j \in S_i} \log(\sigma(\gamma K_{ij})) + \sum_{k \notin S_i}^{K} \log(\sigma(-\gamma K_{ik}))$$

Optimization by Matrix Factorization

Theorem (levy et. al. 2014) : SGNS objective is equivalent to weighted matrix factorization of SPPMI (shifted PMI) matrix

$$PMI_{ij}(M) = log\left(\frac{M_{ij} * |M|}{\sum_k M_{(i,k)} * \sum_k M_{(k,j)}}\right)$$

$$SPPMI_{ij}(M) = \max(PMI_{ij}(M) - log(k), 0)$$

Here, PMI(M) is point wise mutual information matrix, |M| represent sum of all element in matrix M

ExMLDS Algorithm

- Multi-iter SVP algorithm replaced with single step SVD on SPPMI
- Regression and Prediction algorithm are exactly same to the SLEEC
- ExMLDS is 10x faster than the SLEEC with similar performance

Incorporating Label Correlation

- Learn embedding of labels as well as instances jointly
- Overall Idea: think of labels as individual words, whereas instances as a sentence
- PV-DBoW maximize similarity between embedded sentence and words of the sentence.
- Can incorporate auxiliary label-label correlation information

Joint Learning of Embedding and Regressor

$$\nabla_V \mathbb{J}_i = \gamma \sum_{j \in S_i} \sigma(-\gamma K_{ij}) \nabla_V K_{ij} - \gamma \sum_{k \notin S_i} \sigma(\gamma K_{ik}) \nabla_V K_{ik}$$

$$\nabla_V K_{ij} = -ab^3 c z_i (x_i)^T - abc^3 z_j (x_j)^T + bc(z_i x_j^T + z_j x_i^T)$$

$$a = z_i^T z_j, b = \frac{1}{\|z_i\|}, c = \frac{1}{\|z_j\|}$$

Experiments

- We compared our method with several state of art extreme classification algorithms on several datasets
- We used the two most popular metrics Prec@k and nDCG@k for evaluation

Results: ExMLDS1 training time

Method	Bibtex	Delicious	Eurlex	Media mill	Delicious 200K
ExMLDS1	**23**	**259**	**580.9**	**1200**	**1937**
ExMLDS2	143.19	781.94	880.64	12000	13000
SLEEC	313	1351	4660	8912	10000

Results: Missing 80% Labels

Dataset	Prec@k	ExMLDS3	SLEEC	LEML	LEML-IMC
Bibtex	P@1	**48.51**	30.5	35.98	41.23
	P@3	**28.43**	14.9	21.02	25.25
	P@5	**20.7**	9.81	15.50	18.56
Eurlex	P@1	**60.28**	51.4	26.22	39.24
	P@3	**44.87**	37.64	22.94	32.66
	P@5	**35.31**	29.62	19.02	26.54
rcv1v2	P@1	**81.67**	41.8	64.83	73.68
	P@3	**52.82**	17.48	42.56	48.56
	P@5	**37.74**	10.63	31.68	34.82

Results: Joint Learning

Dataset	Prec@k	ExMLDS4	AnnexML	SLEEC
Delicious-200K	P@1	47.70	46.66	**47.85**
	P@3	41.22	40.79	**42.21**
	P@5	37.98	37.64	**39.43**
Wikipedia-500K	P@1	62.27	**63.86**	58.39
	P@3	41.43	**42.69**	37.88
	P@5	31.42	**32.37**	28.21
Amazon-670K	P@1	41.47	**42.08**	35.05
	P@3	36.35	**36.65**	31.25
	P@5	32.43	**32.76**	28.56

References

For dataset details refer to Extreme Classification Repository by Manik Varma (`https://goo.gl/3LvVa6`)