Scalable Semi-Supervised Query Classification Using Matrix Sketching

Young-Bum Kim[†] **Karl Stratos**[‡] **Ruhi Sarikaya**[†]

[†]Microsoft Corporation, Redmond, WA
[‡]Columbia University, New York, NY
{ybkim, ruhi.sarikaya}@microsoft.com
stratos@cs.columbia.edu

Abstract

The enormous scale of unlabeled text available today necessitates scalable schemes for representation learning in language processing. For instance, in this paper we are interested in classifying the intent of a user query. While our labeled data is quite limited, we have access to virtually an unlimited amount of unlabeled queries, which could be used to induce useful representations: for instance by principal component analysis (PCA). However, it is prohibitive to even store the data in memory due to its sheer size, let alone apply conventional batch algorithms. In this work, we apply the recently proposed *matrix sketching* algorithm to entirely obviate the problem with scalability (Liberty, 2013). This algorithm approximates the data within a specified memory bound while preserving the covariance structure necessary for PCA. Using matrix sketching, we significantly improve the user intent classification accuracy by leveraging large amounts of unlabeled queries.

1 Introduction

The large amount of high quality unlabeled data available today provides an opportunity to improve performance in tasks with limited supervision through a semi-supervised framework: learn useful representations from the unlabeled data and use them to augment supervised models. Unfortunately, conventional exact methods are no longer feasible on such data due to scalability issues. Even algorithms that are considered relatively scalable (e.g., the Lanczos algorithm (Cullum and Willoughby, 2002) for computing eigenvalue decomposition of large sparse matrices) fall apart in this scenario, since the data cannot be stored in the memory of a single machine. Consequently, approximate methods are needed.

In this paper, we are interested in improving the performance for sentence classification task by leveraging unlabeled data. For this task, supervision is precious but the amount of unlabeled sentences is essentially unlimited. We aim to learn sentence representations from as many unlabeled queries as possible via principal component analysis (PCA): specifically, learn a projection matrix for embedding a bag-of-words vector into a low-dimensional dense feature vector. However, it is not clear how we can compute an effective PCA when we are unable to even store the data in the memory.

Recently, Liberty (2013) proposed a scheme, called *matrix sketching*, for approximating a matrix while preserving its covariance structure. This algorithm, given a memory budget, deterministically processes a stream of data points while never exceeding the memory bound. It does so by occasionally computing singular value decomposition (SVD) on a small matrix. Importantly, the algorithm has a theoretical guarantee on the accuracy of the approximated matrix in terms of its covariance structure, which is the key quantity in PCA calculation.

We propose to combine the matrix sketching algorithm with random hashing to completely remove limitations on data sizes. In experiments, we significantly improve the intent classification accuracy by learning sentence representations from huge amounts of unlabeled sentences, outperforming a strong baseline based on word embeddings trained on 840 billion tokens (Pennington et al., 2014).

2 Deterministic Matrix Sketching

PCA is typically performed to reduce the dimension of each data point. Let $X \in \mathbb{R}^{n \times d}$ be a data matrix whose n rows correspond to n data points in $\mathbb{R}^d$. For simplicity, assume that X is preprocessed to have zero column means. The key quantity in PCA is the empirical covariance matrix $X^\top X \in \mathbb{R}^{d \times d}$ (up to harmless scaling). It is well-known that the m length-normalized eigenvectors $u_1 \ldots u_m \in \mathbb{R}^d$ of $X^\top X$ corresponding to the largest eigenvalues are orthogonal directions along which the variance of the data is maximized. Then if $\Pi \in \mathbb{R}^{d \times m}$ be a matrix whose i-th column is u_i, the PCA representation of X is given by $X\Pi$. PCA has been a workhorse in representation learning, e.g., inducing features for face recognition (Turk et al., 1991).

Frequently, however, the number of samples n is simply too large to work with. As n tends to billions and trillions, storing the entire matrix X in memory is practically impossible[1]. One solution is to approximate the matrix with some $Y \in \mathbb{R}^{l \times d}$ where $l \ll n$. Many matrix approximation techniques have been proposed, such as random projection (Papadimitriou et al., 1998; Vempala, 2005), sampling (Drineas and Kannan, 2003; Rudelson and Vershynin, 2007), and hashing (Weinberger et al., 2009). Most of these techniques involve randomness, which can be undesirable in certain situations (e.g., when experiments need to be exactly reproducible). Moreover, many are not designed directly for the objective that we care about: namely, ensuring that the covariance matrices $X^\top X$ and $Y^\top Y$ remain "similar".

A recent result by Liberty (2013) gives a *deterministic* matrix sketching algorithm that tightly preserves the covariance structure needed for PCA. Specifically, given a sketch size l, the algorithm computes $Y \in \mathbb{R}^{l \times d}$ such that

$$\left|\left|X^\top X - Y^\top Y\right|\right|_2 \leq 2\,||X||_F^2\,/l \qquad (1)$$

This result guarantees that the error decreases in $O(1/l)$; in contrast, other approximation techniques have a significantly worse convergence bound of $O(1/\sqrt{l})$.

```
Input: data stream $x_1 \ldots x_n \in \mathbb{R}^d$, sketch size $l$
  1. Initialize zero-valued $Y \in 0^{l \times d}$.
  2. For $i = 1 \ldots n$,
     (a) Insert $x_i$ to the first zero-valued row of $Y$.
     (b) If $Y$ has no zero-valued row,
         i. Compute SVD of $Y = U\Sigma V^\top$ where $\Sigma = \mathrm{diag}(\sigma_1 \ldots \sigma_l)$ with $\sigma_1 \geq \cdots \geq \sigma_l$.
         ii. Compute a diagonal matrix $\bar{\Sigma}$ with at least $\lceil l/2 \rceil$ zeros by setting
             $\bar{\Sigma}_{j,j} = \sqrt{\max\left(\Sigma_{j,j}^2 - \sigma_{\lfloor l/2 \rfloor}^2, 0\right)}$
         iii. Set $Y = \bar{\Sigma} V^\top$.
Output: $Y \in \mathbb{R}^{l \times d}$ s.t. $\left|\left|X^\top X - Y^\top Y\right|\right|_2 \leq 2\,||X||_F^2\,/l$
```

Figure 1: Matrix sketching algorithm by Liberty (2013). In the output, $X \in \mathbb{R}^{n \times d}$ denotes the data matrix with rows $x_1 \ldots x_n$.

The algorithm is pleasantly simple and is given in Figure 1 for completeness. It processes one data point at a time to update the sketch Y in an online fashion. Once the sketch is "full", its SVD is computed and the rows that fall below a threshold given by the median singular value are eliminated. This operation ensures that every time SVD is performed at least a half of the rows are discarded. Consequently, we perform no more than $O(2n/l)$ SVDs on a small matrix $Y \in \mathbb{R}^{l \times d}$. The analysis of the bound (1) is an extension of the "median trick" for count sketching and is also surprisingly elementary; we refer to Liberty (2013) for details.

3 Matrix Sketching for Sentence Representations

Our goal is to leverage enormous quantities of unlabeled sentences to augment supervised training for intent classification. We do so by learning a PCA projection matrix Π from the unlabeled data and applying it on both training and test sentences. The matrix sketching algorithm in Figure 1 enables us to compute Π on arbitrarily large data.

There are many design considerations for using the sketching algorithm for our task.

[1]Processing large datasets often require larger memory than the capacity of a typical single enterprise server. Clusters may enable a aggregating many boxes of memory on different machines, to build distributed memory systems achieving large memory capacity. However, building and maintaining these industry grade clusters is not trivial and thus not accessible to everyone. It is critical to have techniques that can process large data within a limited memory budget available in most typical enterprise servers.

3.1 Original sentence representations

We use a bag-of-words vector to represent a sentence. Specifically, each sentence is a d-dimensional vector $x \in \mathbb{R}^d$ where d is the size of the vocabulary and x_i is the count of an n-gram i in the sentence (we use up to $n = 3$ in experiments); we denote this representation by SENT. In experiments, we also use a modification of this representation, denoted by SENT+, in which we explicitly define features over the first two words in a query and also use intent predictions made by a supervised model.

3.2 Random hashing

When we process an enormous corpus, it can be computationally expensive just to obtain the vocabulary size d in the corpus. We propose using random hashing to avoid this problem. Specifically, we pre-define the hash size H we want, and then on encountering any word w we map $w \rightarrow \{1 \dots H\}$ using a fixed hash function. This allows us to compute a bag-of-words vector for any sentence without knowing the vocabulary size. See Weinberger et al. (2009) for a justification of the hashing trick for kernel methods (applicable in our setting since PCA has a kernel (dual) interpretation).

3.3 Parallelization

The sketching algorithm works in a sequential manner, processing each sentence at a time. While it leaves a small memory footprint, it can take prohibitively long time to process a large corpus. Liberty (2013) shows it is trivial to parallelize the algorithm: one can compute several sketches in parallel and then sketch the conjoined sketches. The theory guarantees that such layered sketches does not degrade the bound (1). We implement this parallelization to obtain an order of magnitude speedup.

3.4 Final sentence representation:

Once we learn a PCA projection matrix Π, we use it in both training and test times to obtain a dense feature vector of a bag-of-words sentence representation. Specifically, if x is the original bag-of-words sentence vector, the new representation is given by

$$x_{\text{new}} = \frac{x}{||x||} \oplus \frac{x\Pi}{||x\Pi||} \qquad (2)$$

where $\oplus$ is the vector concatenation operation. This representional scheme is shown to be effective in previous work (e.g., see Stratos and Collins (2015)).

3.5 Experiment

To test our proposed method, we conduct intent classification experiments[2] (Celikyilmaz et al., 2011; Ji et al., 2014; El-Kahky et al., 2014) across a suite of 22 domains shown in Table 1[3]. To learn a PCA projection matrix from the unlabeled data, we collected around 17 billion unlabeled queries from search logs, which give the original data matrix whose columns are bag-of-n-grams vector (up to trigrams) and has dimensions approximately 17 billions by 41 billions, more specifically,

$$X \in \mathbb{R}^{17,032,086,719 \times 40,986,835,008}$$

We use a much smaller sketching matrix $Y \in \mathbb{R}^{1,000,000 \times 1,000,000}$ to approximate X. Note that column size is hashing size. We parallelized the sketching computation over 1,000 machines; we will call the number of machines parallelized over "batch". In all our experiments, we train a linear multi-class SVM (Crammer and Singer, 2002).

Results of Intent Classification Task: Table 1 shows the performance of intent classification across domains. For the baseline, SVM without embedding (w/o Embed) achieved 91.99% accuracy, which is already very competitive. However, the models with word embedding trained on 6 billion tokens (6B-50d) and 840 billion tokens (840B-300d) (Pennington et al., 2014) achieved 92.89% and 93.00%, respectively[4]. To use word embeddings as a sentence representation, we simply use averaged word vectors over a sentence, normalized and conjoined with the original reprsentation as in (2). Surprisingly, when we use sentence representation (SENT) induced from the sketching method with our data set, we can boost the performance up to 93.49%, corresponding to a 18.78% decrease in error relative to a SVM without representation. Also, we see that the exten-

[2]An intent is defined as the type of content the user is seeking. This task is part of the spoken language understanding problem (Kim et al., 2015a; Kim et al., 2015b).

[3]The amount of training data we used ranges from 12k to 120k (in number of queries) across different domains, the test data was from 2k to 20k. The number of intents ranges from 5 to 39 per domains.

[4]50d and 300d denote size of embedding dimension

	w/o Embed	6B-50d	840B-300d	SENT	SENT+
alarm	97.25	97.68	97.5	97.68	97.74
apps	89.16	91.07	92.52	94.24	94.3
calendar	91.34	92.43	92.32	92.53	92.43
communication	99.1	99.13	99.08	99.08	99.12
finance	90.44	90.84	90.72	90.76	90.82
flights	94.19	92.99	93.99	94.59	94.59
games	90.16	91.79	92.09	93.08	92.92
hotel	93.23	94.21	93.97	94.7	94.78
livemovie	90.88	92.64	92.8	93.28	93.37
livetv	83.14	85.02	84.67	85.41	85.86
movies	93.27	94.01	93.97	94.75	95.16
music	87.87	90.37	90.9	91.75	91.33
mystuff	94.2	94.4	94.51	94.51	94.95
note	97.62	98.36	98.36	98.49	98.52
ondevice	97.51	97.77	97.6	97.77	97.84
places	97.29	97.68	97.68	98.01	97.75
reminder	98.72	98.96	98.94	98.96	98.96
sports	76.96	78.53	78.38	78.7	79.44
timer	91.1	91.79	91.33	92.33	92.61
travel	81.58	82.57	82.43	83.64	82.81
tv	91.42	94.11	94.91	95.19	95.47
weather	97.31	97.33	97.4	97.4	97.47
Average	91.99	92.89	93.00	93.49	93.56

Table 1: Performance comparison between different embeddings style.

dend sentence representation SENT+ can get additional gains.

As in Table 2 , we also measured performance of our method (SENT+) as a function of the percentage of unlabeled data we used from total unlabeled sentences. The overall trend is clear: as the number of sentences are added to the data for inducing sentence representation, the test performance improves because of both better coverage and better quality of embedding. We believe that if we consume more data, we can boost up the performance even more.

Results of Parallelization: Table 3 shows the sketching results for various batch size. To evaluate parallelization, we first randomly generate a matrix $\mathbb{R}^{1,000,000\times 100}$ and it is sketched to $\mathbb{R}^{100\times 100}$. And then we sketch run with different batch size. The results show that as the number of batch increases, we can speed up dramatically, keeping residual value $||X^\top X - Y^\top Y||_2$. It indeed satisfies the bound value, $||X||_F^2/l$, which was 100014503.16.

Batch Size	$\|\|X^\top X - Y^\top Y\|\|_2$	time
1	1019779.69	100.21
2	1019758.22	50.31
4	1019714.19	26.50
5	1019713.43	21.67
8	1019679.67	14.53
10	1019692.67	12.13
16	1019686.35	8.53
20	1019709.03	7.35
25	1019650.51	6.40
40	1019703.24	4.97
50	1019689.33	4.48

Table 3: Results for corresponding batch size. Second column indicates the norm of gap between original and sketching matrix. Time represents the running time for sketching methods, measured in seconds.

4 Conclusion

We introduced how to use *matrix sketching* algorithm of (Liberty, 2013) for scalable semi-supervised sentence classification. This algorithm approximates the data within a specified mem-

	0	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
apps	89.16	89.83	90.04	90.26	90.88	91.9	92.41	92.41	92.95	93.72	94.3
music	87.87	89.12	89.61	90.4	90.83	91.26	91.31	91.33	91.38	91.33	91.33
tv	91.42	92.28	92.83	93.61	93.96	94.67	94.91	95.12	95.34	95.44	95.47

Table 2: Performance for selected domains as the number of unlabeled data increases.

ory bound while preserving the covariance structure necessary for PCA. Using matrix sketching, we significantly improved the classification accuracy by leveraging very large amounts of unlabeled sentences.

References

Asli Celikyilmaz, Dilek Hakkani-Tür, and Gokhan Tür. 2011. Leveraging web query logs to learn user intent via bayesian discrete latent variable model. ICML.

Koby Crammer and Yoram Singer. 2002. On the learnability and design of output codes for multiclass problems. *Machine Learning*, 47(2-3):201–233.

Jane K Cullum and Ralph A Willoughby. 2002. *Lanczos Algorithms for Large Symmetric Eigenvalue Computations: Vol. 1: Theory*, volume 41. SIAM.

Petros Drineas and Ravi Kannan. 2003. Pass efficient algorithms for approximating large matrices. In *SODA*, volume 3, pages 223–232.

Ali El-Kahky, Xiaohu Liu, Ruhi Sarikaya, Gokhan Tur, Dilek Hakkani-Tur, and Larry Heck. 2014. Extending domain coverage of language understanding systems via intent transfer between domains using knowledge graphs and search query click logs. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 4067–4071. IEEE.

Yangfeng Ji, Dilek Hakkani-Tur, Asli Celikyilmaz, Larry Heck, and Gokhan Tur. 2014. A variational bayesian model for user intent detection. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 4072–4076. IEEE.

Young-Bum Kim, Minwoo Jeong, Karl Stratos, and Ruhi Sarikaya. 2015a. Weakly supervised slot tagging with partially labeled sequences from web search click logs. In *Proc. of the Conference on the North American Chapter of the Association for Computational Linguistics - Human Language Technologies*, pages 84–92.

Young-Bum Kim, Karl Stratos, Ruhi Sarikaya, and Minwoo Jeong. 2015b. New transfer learning techniques for disparate label sets. In *Proc. of Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

Edo Liberty. 2013. Simple and deterministic matrix sketching. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 581–588. ACM.

Christos H Papadimitriou, Hisao Tamaki, Prabhakar Raghavan, and Santosh Vempala. 1998. Latent semantic indexing: A probabilistic analysis. In *Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 159–168. ACM.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12:1532–1543.

Mark Rudelson and Roman Vershynin. 2007. Sampling from large matrices: An approach through geometric functional analysis. *Journal of the ACM (JACM)*, 54(4):21.

Karl Stratos and Michael Collins. 2015. Simple semi-supervised pos tagging. In *Proceedings of NAACL-HLT*, pages 79–87.

Matthew Turk, Alex P Pentland, et al. 1991. Face recognition using eigenfaces. In *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference on*, pages 586–591. IEEE.

Santosh S Vempala. 2005. *The random projection method*, volume 65. American Mathematical Soc.

Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. 2009. Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1113–1120. ACM.