# Distributed Training Strategies for Knowledge Base Completion Model(ConvKB)

**Vikram Bhatt** [*1] **Kapil Pathak** [*2]

## Abstract

In this report, we present our work on Convolutional Neural Network based knowledge graph embedding model named ConvKB (Nguyen et al., 2017) for knowledge based completion.We trained our model in various distributed training scenarios such as i)Bulk Synchornous Parallel(BSP) Bridging Model ii)Asynchronous Parallel Execution(ASP) iii) DOWNPOUR SGD(Dean et al., 2012b)- an aysnchronous stochastic gradient procedure supporting large number of replicas in Distributed Tensorflow framework(Abadi et al., 2015).We initially performed experiments on benchmark dataset FB15K-237 and explored scalability issues in these distributed settings for link prediction tasks.We pre-processed entire Freebase(RDF triples) dump using Hadoop,for the purpose of demonstration of scalability,we constructed our own dataset(FB-1GB).Experiments shows that DOWNPOUR gives best trade-off between training time and convergence stability.Although,there doesn't exist explicit API's in Tensorflow at the moment, for Stale Synchronous Parallel(SSP) Bridging Model,we propose(one of many possible) simple method for implementation in the framework.Our vanilla version of distributed ConvKB model achieves competitive results on link prediction performance on benchmark dataset FB15k-237.

## 1. Introduction

In this report, the first section talks about what Freebase dataset, its ontology, general schema. Given that it's a huge dataset, this section talks about the preprocessing techniques we have implemented. The second section talks about ConvKB model. The third section talks about various distributed learning schemes we have tried and their analysis. The fourth section talks about results and analysis. The last section is dedicated for the further ideas developed during the project and are yet to be tried.

## 2. Literature Review

Knowledge graphs are graph structured bases, where facts are represented as triplets consisting subject entity (head), relation and object entity (tail) given as (h,r,t).However, knowledge bases tend to suffer from incompleteness and identifying missing links is referred to as link predcition. Considering the fact, these bases contain billions of dacts, link prediction should scale well with respect to parameters of our model and computational costs. Various embedding methods have been proposed previously(Bordes et al., 2013)(Trouillon et al., 2016)(Nguyen et al., 2016) using translational based characteristics on low dimensional embeddings of relations and entities.Although these models are simple and scalable, they learn less expressive features. A simple way to learn more expressive features capturing inter-relationships between entities and relations,is to use deep learning models.Convolutional neural networks tend to capture non-linear realtions between the embeddings with few number of parameters.(Dettmers et al., 2018) proposed ConvE model,a multilayer convolutional network which is paramter efficient and acheived state of the art results on various benchmark datasets.(Nguyen et al., 2017) in their article state that,

> ConvE doesn't capture global relationships among same dimensional entries of an embedding triple, so ConvE ignores the trasitional characteristic in transition-based models,which is one of the most useful intuitions for the task.ConvKB models relationships among same dimensional entries of the embeddings.

[*]Equal contribution [1]Department of Computational and Data Sciences,Indian Institute of Science, Bangalore [2]Department of Computer Science and Automation,Indian Institute of Science, Bangalore. Correspondence to: Vikram Bhatt <vikrambhatt@iisc.ac.in>, Kapil Pathak <kapilpathak@iisc.ac.in>.

*Proceedings of the $35^{th}$ International Conference on Machine Learning*, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

## 3. Model-ConvKB

We denote a valid factual triple in the form of $(head, relation, tail)$ as $(h, r, t)$ and the corresponding k-dimesional embeddings as $(v_h, v_r, v_t)$.We stack these embedding these horizontally in a matrix form $A = [v_h, v_r, v_t] \in R^{k\times 3}$.We use $\tau$ number of filters $\omega \in R^{1\times 3}$, which slides on each row of matrix of $A$,as illustrated in the Fig.1 and generate $k$ feature maps $v = [v_1, v_2, \cdot, v_k] \in R^k$ where $v_i = g(\omega * A_{i,:} + b)$, $b \in R$ is bias and $g$ is non-linear function(such as ReLU).These $\tau$ feature maps concatenated vertically and formed single vector of shape $R^{\tau k\times 1}$ ,then computed dot product with vector $w \in R^{k\times 1}$ of same size to give score for triple $(h, r, t)$.

$$f(h, r, t) = concat(g([v_h, v_r, v_t] * \Omega)).w$$

where f is score function.

We use loss function $L$ with $L_2$ regularization on the weight vector $w$

$$L = \sum_{(h,r,t)\in\{G\cup G'\}} log(1+exp(l_{(h,r,t)}.f(h,r,t)))+\frac{\lambda}{2}||w||^2$$

where,

$$l_{(h,r,t)} = \begin{cases} 1 \text{ for } (h,r,t) \in G, \\ -1 \text{ for } (h,r,t) \in G' \end{cases}$$

G' is a collection of invalid tuples generated by corrupting valid tuples in G. We use the Bernoulli Trick to generate head or tail for sampling invalid triples.(Wang et al., 2014). As we need to scale up this model for larger dataset, we add 1 more layer of CNN layer on the top of initial layer.

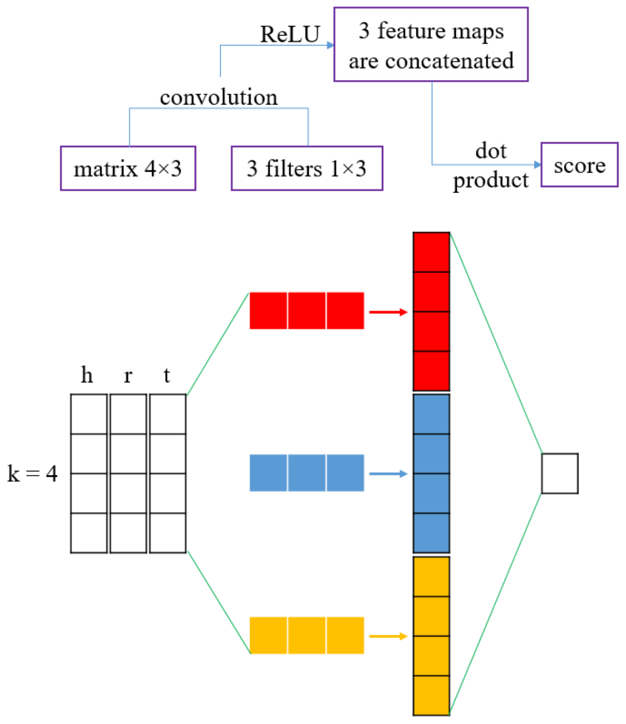

*Figure 1.* Graphical illustration of ConvKB model(Nguyen et al., 2017) with embedding size $k = 4$, the number of filters $\tau = 3$.

*Table 1.* Training times for various number of workers with oone(fixed) parameter server.

| No. of workers | Wallclock time |
|---|---|
| 2 | 7hr 12m |
| 3 | 6hr 4m |
| 4 | 3hr 12m |
| 5 | 3hr |

## 4. Experiments

### 4.1. Bulk Synchronous Parallel(BSP)

In Bulk Synchronous Parallel (BSP), the model alternate between computation phase and communication phase or also called synchronized barrier. Each worker accumulates the gradients for all parameters until the next superstep and the effect of the computations of one worker node is invisible to other worker nodes until the next superstep occurs. Beacuse of the cleaner separation of computation and communication phase, BSP scheme often enjoys correctness of parallel machine learning algorithms.

In (Abadi et al., 2016)Tensorflow API, *SyncReplicasOptimizer* performs similar job to BSP. Here all accumulators,each for individual parameter performs the job of gradients average. After each global step, the accumulator checks the staleness of all gradients and drops the gradients before those. A drawback of this scheme is that the throughput of individual worker is severely affected as each worker needs to wait until the next superstep occurs.Fig.2 and Fig.3 shows the learning curve for BSP training with 2 and 5 workers respectively.Table 1 shows training times on FB15k-237 dataset.

### 4.2. Asynchronous SGD

Unlike BSP, in asynchronous SGD, any worker never waits for any superstep or other worker to catch synchronize the computations. Each worker computes gradients and communicates with parameter server separately throughout the course of each iteration. This scheme achieves near-ideal P-fold throughput in each iteration. But as there is no synchronization among the co-workers,the staleness of the gradients causes slower con- vergence rate per iteration than BSP scheme. If the staleness is not bounded, it may even cause incorrect gradient calcula- tions and wrong results. This is a drawback of asynchronous SGD.

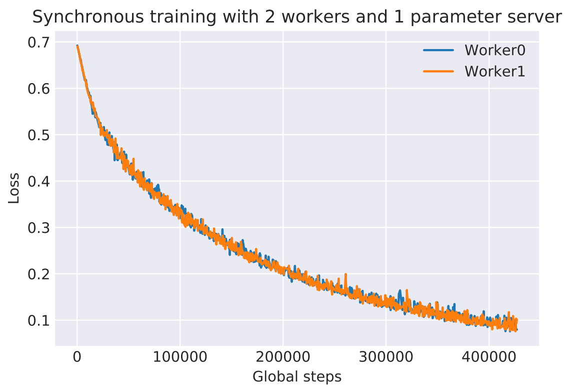

*Figure 2.* Learning curve for synchronous SGD with 2 workers and 1 parameter server.

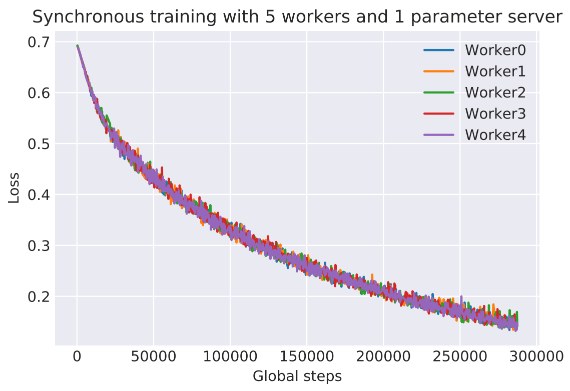

*Figure 3.* Learning curve for synchronous SGD with 5 workers and 1 parameter server

## 4.3. Stale Synchronous Parallel (SSP) Bridging model

In case of asynchronous SGD, as unbounded staleness may cause incorrect results, Stale Synchronous Parallel (SSP) allows us to overcome this issue by bounding the staleness for particular s. For given staleness s, SSP allows a faster worker go ahead in terms of iterations than its other co-workers until it is s iterations apart. The local model parameters become stale as each worker receives partial updates.The advantage of SSP model is that it behaves like Asyncrhonous model most of the time.It can also stop workers as needed to ensure correct update of parameters. SSP bridging model find advantages of both BSP and asynchronous SGD as it increases through put per iteration compared to BSP and avoid unbounded staleness enhancing convergence rate.(Xing et al., 2016)

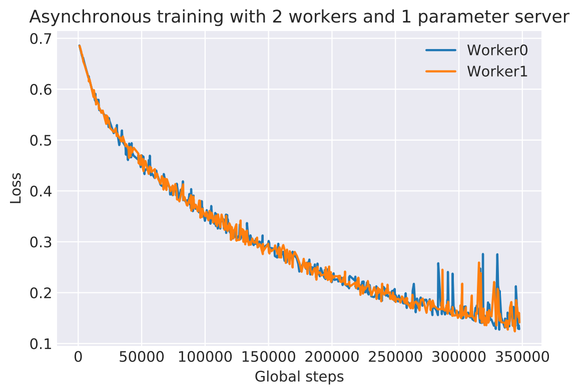

*Figure 4.* Learning curve for asynchronous SGD with 2 workers and 1 parameter server.

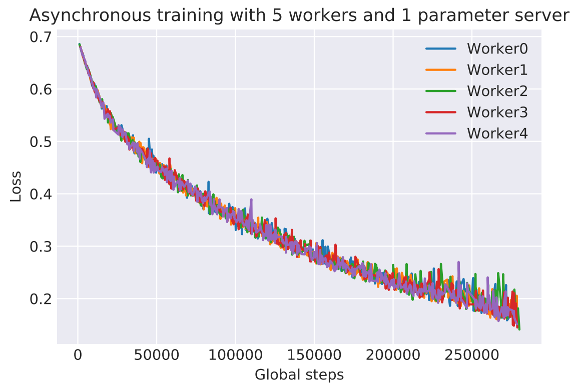

*Figure 5.* Learning curve for asynchronous SGD with 5 workers and 1 parameter server.

## 4.4. Downpour SGD

Another variant of asynchronous stochastic gradient descent that can be applied on large dataset is Downpour SGD. Here too, the model replicas run independently along with independent parameter server shards. In Downpour SGD, the communication overhead can be reduced by limiting parameter requests from each replica to parameter server after n_push steps and n_fetch steps vice versa. In Downpour SGD, a separate Adagrad optimizer is applied on the local copy of parameters on each worker. Here each model replica is calculating gradients on the set of parameters which are stale depending upon the n_fetch and n_push steps. This introduces stochasticity into the model. (Dean et al., 2012a) suggest that Adagrad optimizer along with ”warmstarting” strategy allows to keep more number of replicas that can work productively and serve the purpose of distributed train-

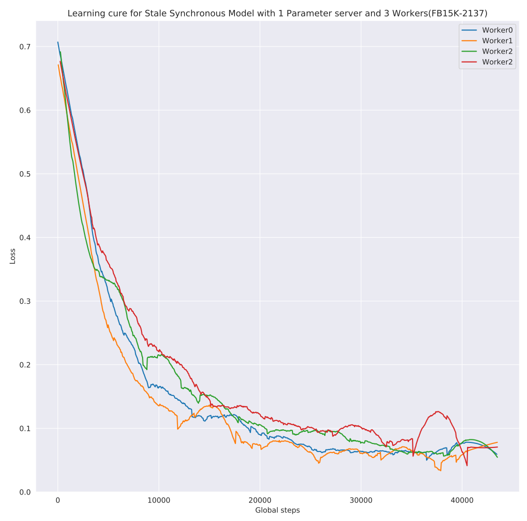

*Figure 6.* Learning curve for SSP-SGD with 4 workers and 1 parameter server.

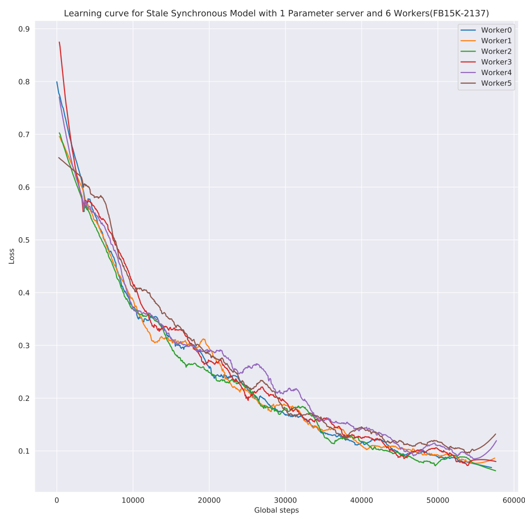

*Figure 7.* Learning curve for SSP-SGD with 6 workers and 1 parameter server.

ing. This induces more stability into the training of deep networks. Published ConvKB model uses single layer of CNN. But if we need to scale up the knowledge graph embeddings further, we can add more number of CNN layers to the model which induces more nonlinearity into the model. Hence, apart from Stale Synchrnous Parallel (SSP), Downpour SGD is an effective distributed learning strategy, especially for ConvKB model.

## 5. Freebase

The Freebase knowledge graph is an important semantic web and linked data technology. It was acquired by Google in 2010 and shut down in 2016. This entire data dumb is of size 33 GB in compressed(gZip) format. This data is useful for various research tasks such as information retrieval, knowledge based question answering etc.

### 5.1. Freebase Schema

In Freebase, each entity is represented by unique machineId identifier. Each machineId starts with "/m/" followed by alphanumeric characters. Each relation is represented by domain followed by type followed by property and each of them separated by "/". For example, for machineId "/m/xyz789" the place of birth given by machineId "/m/cincinn123" can be incorporated into triplet as follows

$$\texttt{/m/xyz789}, \texttt{/people/person/place\_of\_birth},$$

$$\texttt{/m/cincinn123}$$

Here machineId /m/xyz789 is of domain people, of type person and a machineId /m/cincinn123 is related to /m/xyz789 with property place_of_birth. Such kind of triplets are called facts.Within each type of domain, there are further fineraccruedgradients 0 details given by $/property$.The following examples states the fact $/m/syz789$ entity is part of topic people and has type property person.

$$\texttt{/m/xyz789}, \texttt{/type/object/type}, \texttt{/people/person}$$

### 5.2. Preprocessing of Freebase Dataset

This dataset is in the form of N-Triples RDF format, each of the entities of the format "/m/xyz789" has been converted to "/m.xyz789" while each of the relations of the format "/people/person" has been converted to "/people.person". These changes have reverted to original form by replacing "." by "/". Apart from this change, each object in the data set is encoded with full URL path such as

$$\texttt{http}: \texttt{//rdf.freebase.com/ns/m.xyz789}$$

In the preprocessing step, these URLs are removed to reduce the size of the dataset as these URLs are not required to incorporate into the model.Apart from machineIds, the triplets consisting of *mediators* starting with "/g" are also present in the data dumps. These triplets consisting of *mediators* only convey properties of the head entity. We reduced the entire

**Algorithm 1** DOWNPOURSGDCLIENT(Dean et al., 2012a)($\alpha, n\_fetch, n\_push$)

```
procedure
StartAsynchronouslyFetchingParameters(parameters):
parameters=GetParametersFromParamServer()

procedure
StartAsynchronouslyPushingGradients(accruedgradients
SendGradientStopParamServer(accruedgradients)
accruedgradients = 0

global parameters, accruedgradients
step = 0
accruedgradients = 0

while true do
  if (step mod n_fetch) == 0 then
    StartAsynchronouslyFetchingParameters(parameter
  end if
  data =GetNextMinibatch()
  gradient = ComputeGradient(parameters, data)
  accruedgradients = accruedgradients + gradient
  parameters = parameters - $\alpha\times$ gradient
end while
if (step mod n_push) == 0 then
  StartAsynchronouslyPushingParameters(parameters)
end if
```

dataset of Freebase to only facts in the form of

$$/m/070xg/sports/sports\_team/colors/m/01g5v$$

with Hadoop map-reduce job.

### 5.3. Characteristics of Freebase Dataset

Fig.12 shows the distribution of number of triples topics obtained in the reduced dataset obtained after mapreduce job.We observe the distribution is very skewed,if we want to sample using these proportions one might not sample any triplets from the tail at all. We want to create a new train,test and valid sets from the dataset we obtained from mapreduce job,one can take two ways.

1. Determinstic Splitting
   (a) Since our dataset is huge(17GB) shuffle,sort and split in memory is costly,we decided to go for probabilistic splitting.
2. Probabilistic Splitting
   (a) We stream through the entire dataset and generate a random number uniformly between 0 and 1.If the generated random number is greater than specifed threshold (threshold sepcifies the percentage of triples in original dataset we want to retain) we add that to our new dataset.This kind of splitting is cheap and takes care of shuffling by the randomness and we never run out of memory.

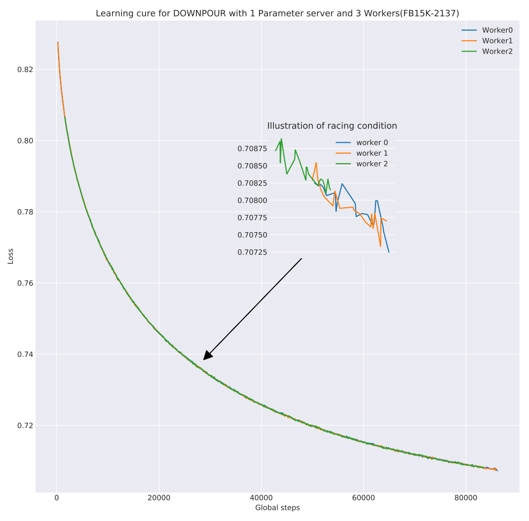

*Figure 8.* Learning curve for Downpour SGD with 3 workers and 1 parameter server.

*Table 2.* Comparison for various Benchmarks with and without distributed frameworks on FB15k-237

| VARIANTS | MR | MRR | HITS@10 |
|---|---|---|---|
| CONVE | 246 | 0.316 | 49.1 |
| CONVKB | 257 | 0.396 | 51.7 |
| BSP CONVKB | 278.16 | 0.332 | 46.7 |
| ASP CONVKB | 344.46 | 0.288 | 43.4 |
| SSP CONVKB | 342.43 | 0.307 | 44.2 |
| DOWNPOUR CONVKB | 312 | 0.301 | 42.6 |

## 6. Test Protocol

We used mean rank(MR), mean reciprocal rank(MRR) and Hits@10 as evaluation metrics for our model.Table.2 shows our results for various model we trained. We are using FB15k dataset for these evaluations.

## Acknowledgements

**Do not** include acknowledgements in the initial version of the paper submitted for blind review.

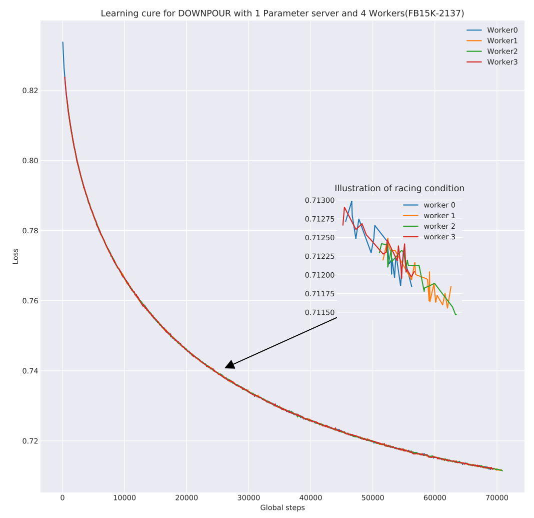

*Figure 9.* Learning curve for Downpour SGD with 4 workers and 1 parameter server.

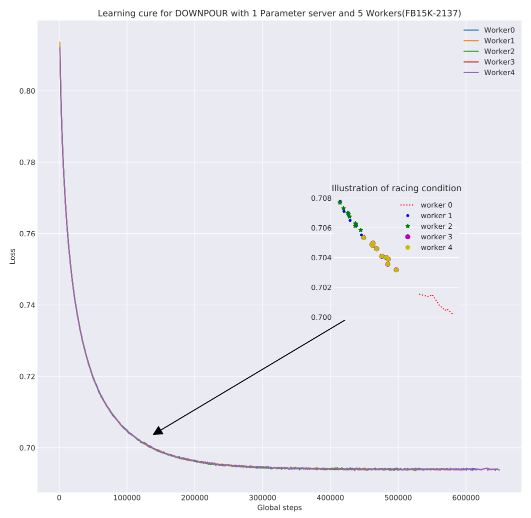

*Figure 10.* Learning curve for Downpour SGD with 5 workers and 1 parameter server.

If a paper is accepted, the final camera-ready version can (and probably should) include acknowledgements. In this case, please place such acknowledgements in an unnumbered section at the end of the paper. Typically, this will include thanks to reviewers who gave useful comments, to colleagues who contributed to the ideas, and to funding agencies and corporate sponsors that provided financial support.

## References

Abadi, Martín, Agarwal, Ashish, Barham, Paul, Brevdo, Eugene, Chen, Zhifeng, Citro, Craig, Corrado, Greg S., Davis, Andy, Dean, Jeffrey, Devin, Matthieu, Ghemawat, Sanjay, Goodfellow, Ian, Harp, Andrew, Irving, Geoffrey, Isard, Michael, Jia, Yangqing, Jozefowicz, Rafal, Kaiser, Lukasz, Kudlur, Manjunath, Levenberg, Josh, Mané, Dan, Monga, Rajat, Moore, Sherry, Murray, Derek, Olah, Chris, Schuster, Mike, Shlens, Jonathon, Steiner, Benoit, Sutskever, Ilya, Talwar, Kunal, Tucker, Paul, Vanhoucke, Vincent, Vasudevan, Vijay, Viégas, Fernanda, Vinyals, Oriol, Warden, Pete, Wattenberg, Martin, Wicke, Martin, Yu, Yuan, and Zheng, Xiaoqiang. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`. Software available from tensorflow.org.

Abadi, Martín, Agarwal, Ashish, Barham, Paul, Brevdo, Eugene, Chen, Zhifeng, Citro, Craig, Corrado, Gregory S., Davis, Andy, Dean, Jeffrey, Devin, Matthieu, Ghemawat, Sanjay, Goodfellow, Ian J., Harp, Andrew, Irving, Geoffrey, Isard, Michael, Jia, Yangqing, Józefowicz, Rafal, Kaiser, Lukasz, Kudlur, Manjunath, Levenberg, Josh, Mané, Dan, Monga, Rajat, Moore, Sherry, Murray, Derek Gordon, Olah, Chris, Schuster, Mike, Shlens, Jonathon, Steiner, Benoit, Sutskever, Ilya, Talwar, Kunal, Tucker, Paul A., Vanhoucke, Vincent, Vasudevan, Vijay, Viégas, Fernanda B., Vinyals, Oriol, Warden, Pete, Wattenberg, Martin, Wicke, Martin, Yu, Yuan, and Zheng, Xiaoqiang. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *CoRR*, abs/1603.04467, 2016. URL `http://arxiv.org/abs/1603.04467`.

Bordes, Antoine, Usunier, Nicolas, Garcia-Duran, Alberto, Weston, Jason, and Yakhnenko, Oksana. Translating embeddings for modeling multi-relational data. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 26*, pp. 2787–2795. Curran Associates, Inc., 2013. URL `http://papers.nips.cc/paper/5071-translating-embeddings-for-modeling-multi-re` `pdf`.

Dean, Jeffrey, Corrado, Greg, Monga, Rajat, Chen, Kai, Devin, Matthieu, Mao, Mark, aurelio Ranzato, Marc, Senior, Andrew, Tucker, Paul, Yang, Ke, Le, Quoc V., and Ng, Andrew Y. Large scale distributed deep networks. In

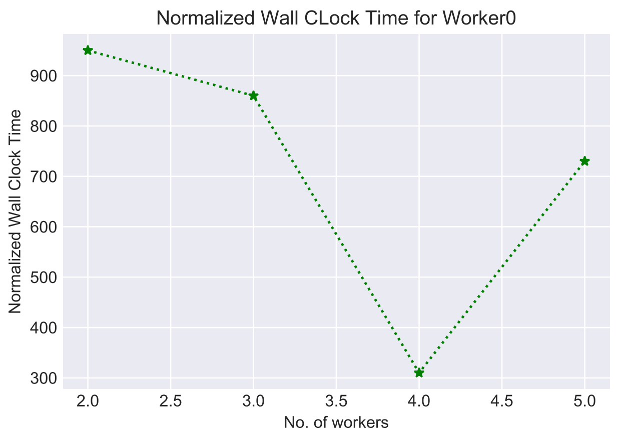

*Figure 11.* Normalized Wall Clock Time for worker0 with different number of worker nodes

Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 25*, pp. 1223–1231. Curran Associates, Inc., 2012a. URL http://papers.nips.cc/paper/4687-large-scale-distributed-deep-networks.pdf.

Dean, Jeffrey, Corrado, Greg, Monga, Rajat, Chen, Kai, Devin, Matthieu, Mao, Mark, Senior, Andrew, Tucker, Paul, Yang, Ke, Le, Quoc V, et al. Large scale distributed deep networks. In *Advances in neural information processing systems*, pp. 1223–1231, 2012b.

Dettmers, Tim, Pasquale, Minervini, Pontus, Stenetorp, and Riedel, Sebastian. Convolutional 2d knowledge graph embeddings. In *Proceedings of the 32th AAAI Conference on Artificial Intelligence*, pp. 1811–1818, February 2018. URL https://arxiv.org/abs/1707.01476.

Nguyen, Dai Quoc, Nguyen, Tu Dinh, Nguyen, Dat Quoc, and Phung, Dinh. A novel embedding model for knowledge base completion based on convolutional neural network. *arXiv preprint arXiv:1712.02121*, 2017.

Nguyen, Dat Quoc, Sirts, Kairit, Qu, Lizhen, and Johnson, Mark. Stranse: a novel embedding model of entities and relationships in knowledge bases. *CoRR*, abs/1606.08140, 2016. URL http://arxiv.org/abs/1606.08140.

Trouillon, Théo, Welbl, Johannes, Riedel, Sebastian, Gaussier, Éric, and Bouchard, Guillaume. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, pp. 2071–2080, 2016.

Wang, Zhen, Zhang, Jianwen, Feng, Jianlin, and Chen,

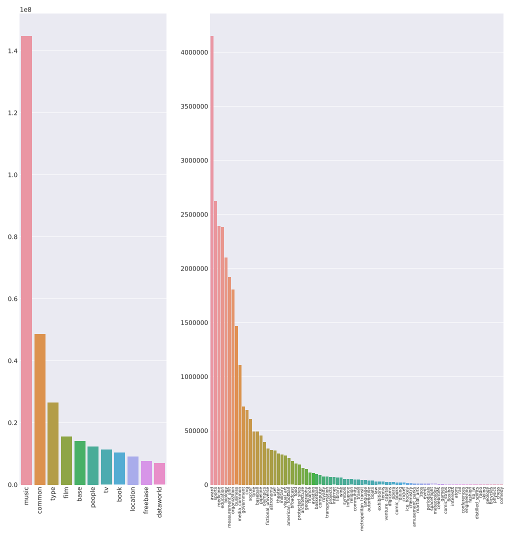

*Figure 12.* Frequency of topics processed from the entire Freebase dump shows very skew distribution,dominated by few topics weighing heavily compared to the rest.

Zheng. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, volume 14, pp. 1112–1119, 2014.

Xing, Eric P, Ho, Qirong, Xie, Pengtao, and Wei, Dai. Strategies and principles of distributed machine learning on big data. *Engineering*, 2(2):179–195, 2016.

## A. Acknowledgements

1. We would like to thank Dai Quoc Nguyen for providing baseline model and helper functions. (https://github.com/daiquocnguyen/ConvKB)

2. We would like to appreciate Tommy Mulc(https://github.com/tmulc18/Distributed-TensorFlow-Guide) for their awesome tutorials on distributed tensorflow frameworks.