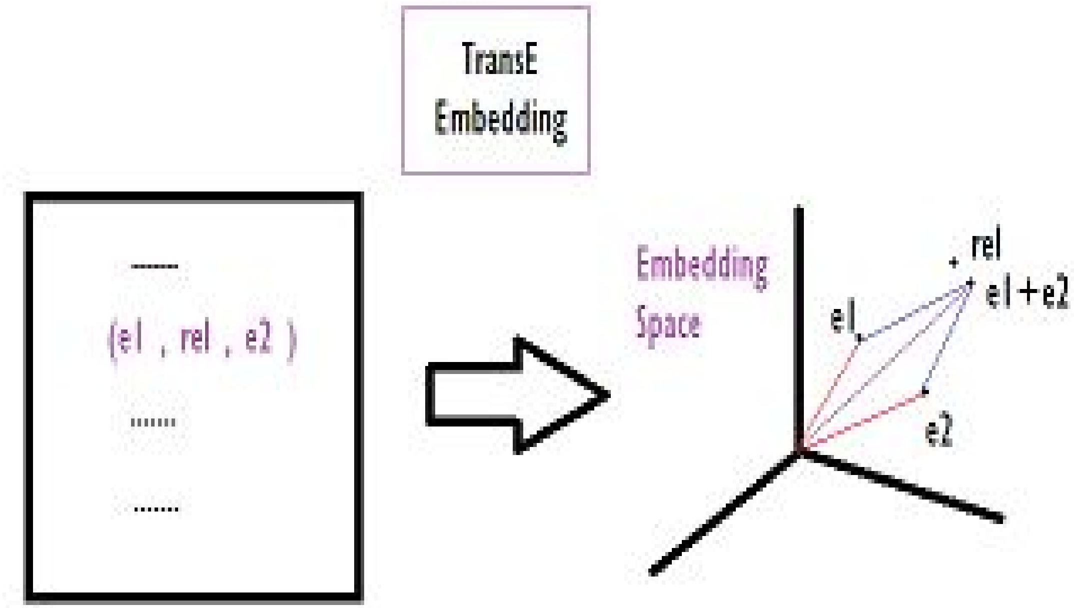
PARALLEL AND SCALABLE IMPLEMENTATIONS OF TRANS-E AND TRANS-H IN APACHE SPARK USING SPLASH

Krishna Manglani Prem and Srinivas

What is Trans-E ?

- One of the first/simplest algorithms for embedding knowledge graphs in a vector space.
- The basic idea behind Trans-E is that, the relationship between two entities corresponds to a translation between the embedding of entities in a vector space, that is, ***h + r = t*** when ***(h, r, t)*** holds.
- Hence, TransE assumes the score function **f_r(h, r,t) = norm((h + r - t),2)** is low if ***(h, r, t)*** holds, and high otherwise
- Number of parameters to learn : **O(d x (num_entities + num_relations))**, where '***d***' is the dimension of the embedding space.
- Training Method : SGD
- Challenging cases : 1-to-N, N-to-1 and N-to-N relations

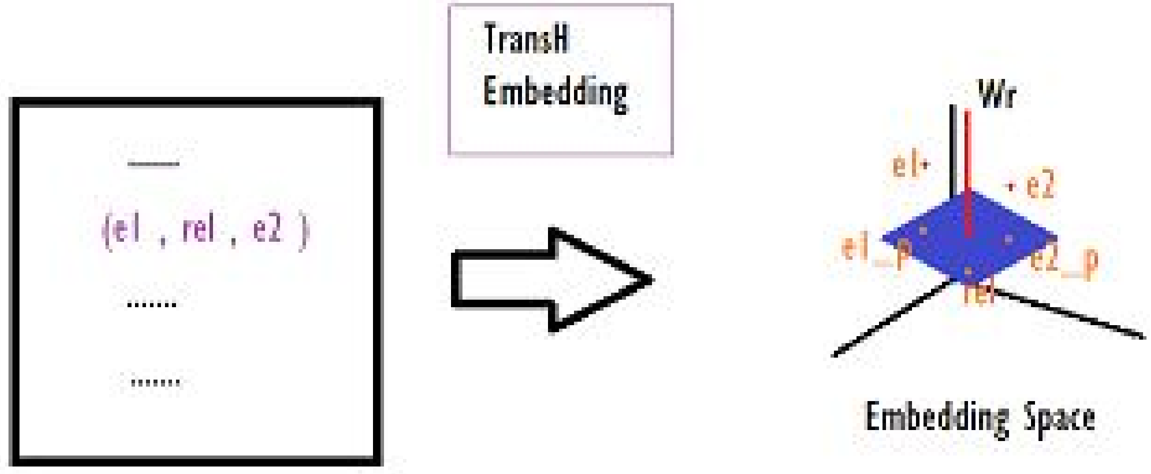
Algorithm : Trans-E



What is Trans-H ?

- To address the issue of TransE when modeling N-to-1, 1-to-N and N-to-N relations, TransH is proposed to enable an entity to have distinct distributed representations when involved in different relations.
- For a relation '*r*', TransH models the relation as a vector '***r***' on a hyperplane with '***Wr***' as the normal vector. For a triple ***(h, r, t)***, the entity embeddings '***h***' and '***t***' are first projected to the hyperplane of '***Wr***', denoted as '***hp***' and '***tp***'.
- The score function is defined as **fr(h, r, t) = norm(hp + r - tp , 2)** .
- Number of parameters to learn : **O(d x (num_entities + 2 x num_relations))**, where '***d***' is the dimension of the embedding space.
- Training Method : SGD

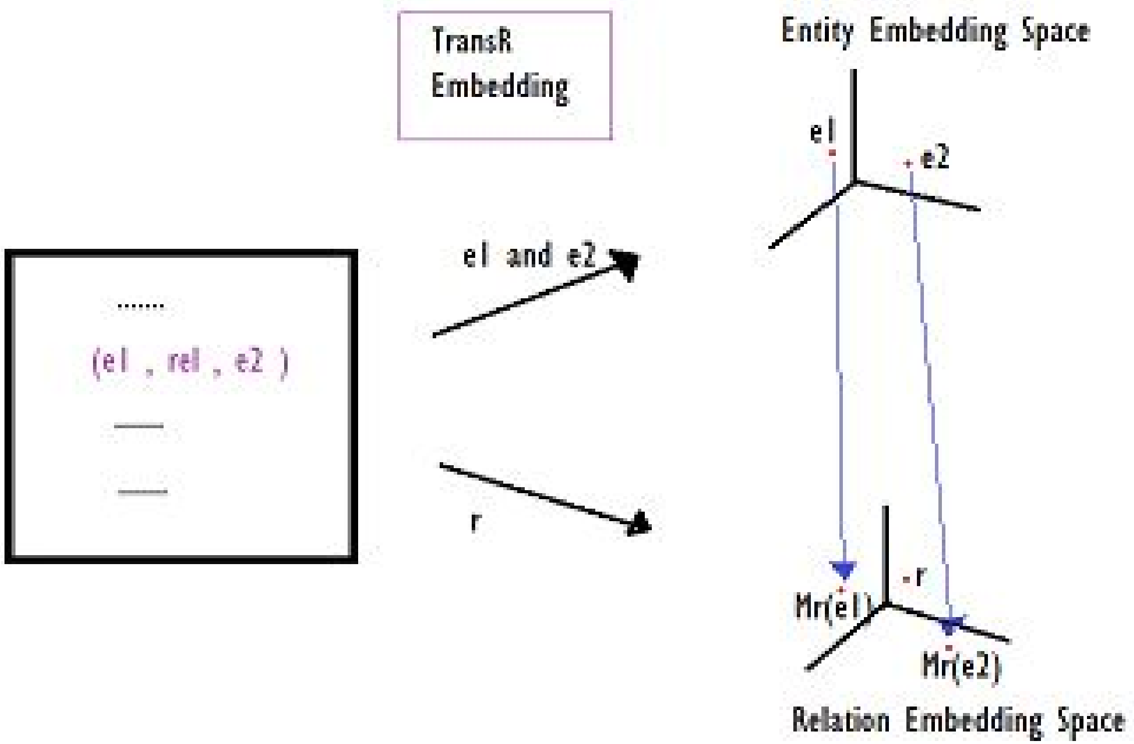
Algorithm : Trans-H



What is Trans-R ?

- In TransR, for each triple ***(h, r, t)***, entities embeddings are set as ***h, t*** $\in$ ***Rk*** and relation embedding is set as ***r in Rd*** . Note that, the dimensions of entity embeddings and relation embeddings are not necessarily identical, i.e., ***k != d***.
- For each relation ***r***, we set a projection matrix ***Mr*** , which projects entities from entity space to relation space. With the mapping matrix, we define the projected vectors of entities as ***hr = h x Mr, tr = t x Mr***.
- The score function is correspondingly defined as: ***fr(h, r, t) = norm(hr + r - tr,2)*** .
- Number of parameters to learn : **O**
  (d x (num_entities + num_relations^2)), where '***d***' is the max of dimensions of the embedding spaces.
- Training Method : SGD

Algorithm : Trans-R



Problem Statement :

A Parallel and Scalable implementation of
Trans-E , Trans-H and Trans-R.

CHALLENGES :
1) PARALLELIZATION
2) SCALABILITY

Choice of Parallelization Strategy

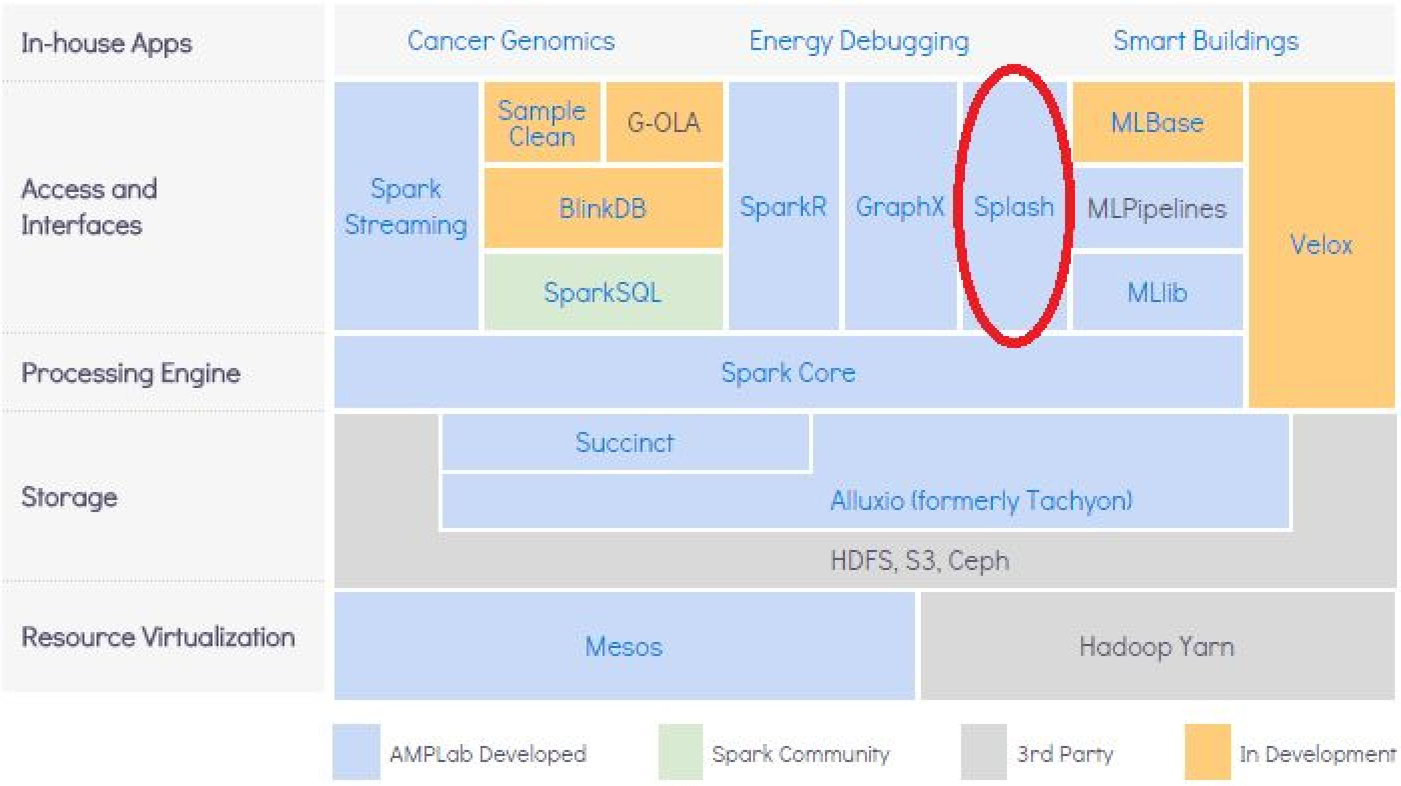
Parallel Algorithm :

1. Bounded Delayed Updates – Distributed Delayed Stochastic Optimization
2. Asynchronous Updates – Hogwild !
3. Weighted Average – Splash

Parallel Implementation :

1. Parameter Server – parameterserver.org, petuum, Naiad
2. Graph Lab -
3. Splash - Parameter Server + Reweighting + Programming Interface

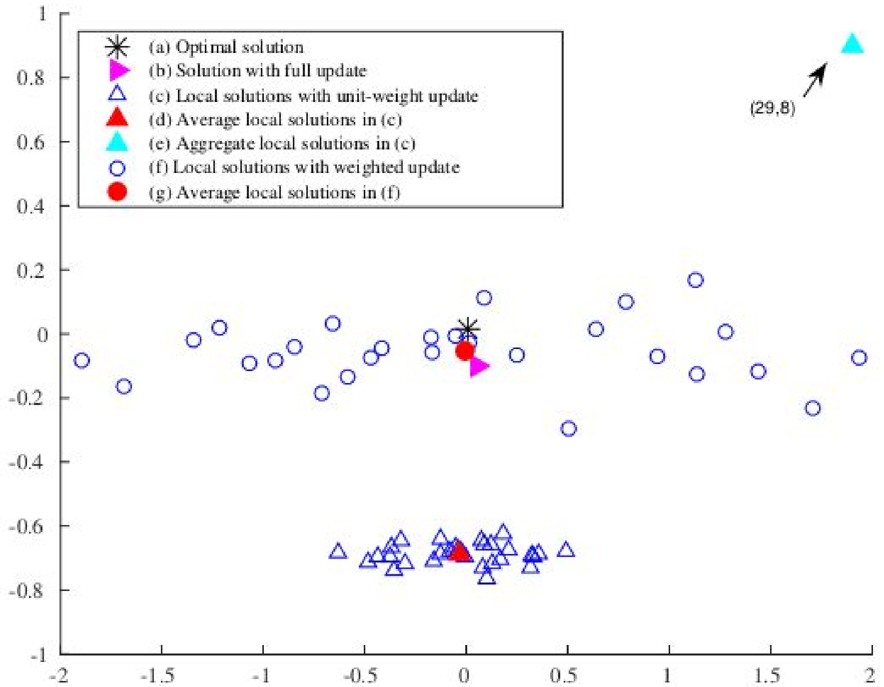
Brief Overview of SPLASH



SPLASH Features

- SPLASH = All SPARK features +
  Shared Variables +
  Reweighting scheme.
- Communication for Synchronization = shared variables
- Theoretical requirement for faster convergence = reweighting.

How does Splash Work??



$$v_{\text{new}} = \frac{1}{m}\sum_{i=1}^{m}\Big(\Gamma(mG_i)\cdot v_{\text{old}} + \Delta(mS_i)\Big) + \sum_{i=1}^{m} T(mS_i).$$

Implementation Stages

Pre-Processing → Initialization → Training → Testing

- Pre-Processing – Create graph with negative samples
- Initialization – Random Initialization for TransE/TransH and output of TransE for TransR
- Training – TransE / TransH / TransR
- Testing – Calculate mean rank and Hits@10

Scalability Challenges

1. Sampling –

For every triplet **(e1, rel, e2)** in the graph , we need to create a negative samples of the form **(e1', rel, e2)** or **(e1, rel, e2')** such that the negative samples are not part of the knowledge graph.

For small graphs, this can be done easily by storing the following steps :

- Store all the triplets in a hash table.
- Given a triplet **(e1, rel, e2)** randomly create an alternative sample **(e1, rel, e2')** .
- Search the hash table if **(e1, rel, e2')** is available.
- If so, sample again and if not add it as a negative example. Repeat this process till enough negative samples have been created.

2. Synchronization of large number of shared variables -

Taken care by the SPLASH framework.

3. Computation of filtered MeanRank and Hits@10-

Need to have entire training dataset available at time of generating rank of triplet.

4. **Normalization of vectors after every iteration:**

Tweaked Splash library in order to achieve this

Proposed Solution for Sampling

- ➔ PreProcess the negative samples before hand
- ➔ Create multiple Random corrupt triples (= Replication factor) for each true triple

Concerns:

- ❖ High Density elements do not need high replication factors
- ❖ Need to restrict maximum replication factor in order to limit data duplication
- ❖ Distribution of density could be uniform which has very less probability in real world knowledge graphs

Algorithm 3 corrupt_right_elem($\langle (e1, rel), \{e2\} \rangle$)

m = replication_factor(e1)
Declare Corrupted List = {1 2 .. entity_num }
Remove all e2 from corrupted list
for each e2 **do**
  repeat m times **do**
    take random t_corrupt from corrupted list
    emit $\langle (e1, e2, rel, e1, t_{corrupt}, rel) \rangle$

Current Status of the Implementation

- Pre-Processing code- complete
- Initialization code – complete
- Training code

1. TransE - complete
2. TransH - complete
3. TransR - almost complete

- Testing code - complete

Experiments and Results for TransE

Running Time in seconds	FB15k
Serial	3258
Parallel	2707

Mean Rank	FB15k
Serial	243
Parallel	673

Hits@10	FB15k
Serial	34.9 %
Parallel	14 %

Work Remaining :

1. Run pre-processing code on large graphs and then run the algorithms to check weak scaling.
2. Normalization issue - Normalization of the weight vector affects performance. Need to understand when to normalize and when not to normalize.
3. The number of dimensions also affects the performance. Need to estimate dimension based on the dataset.