Efficient spatial representation for Entity-Typing

Undergraduate Thesis

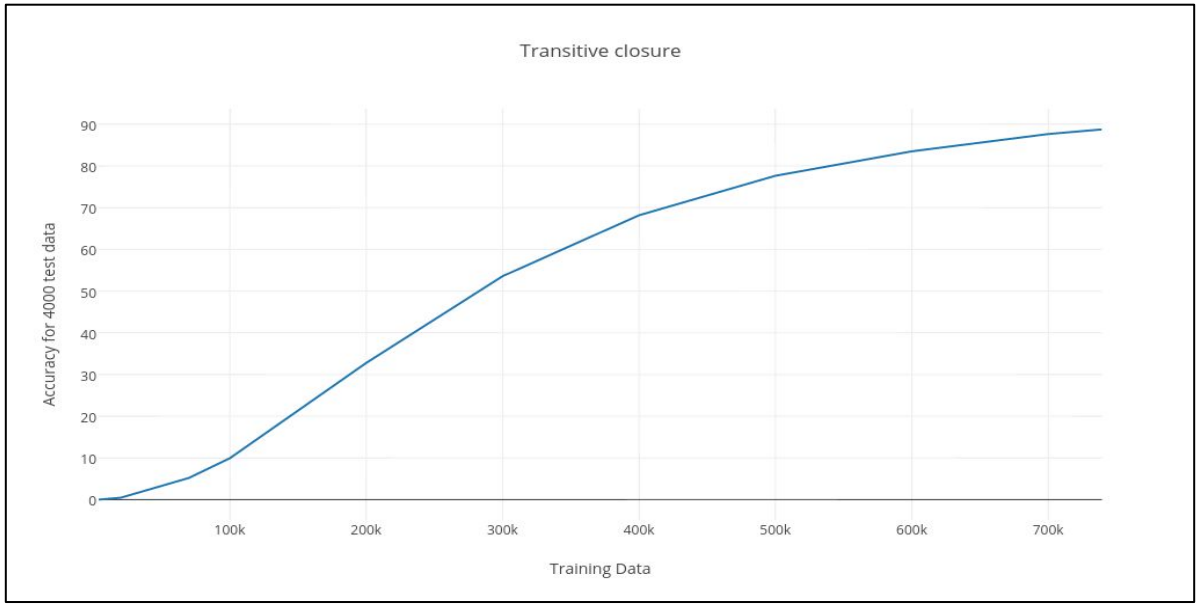
Anand Dhoot - 130070009
Aditya Kusupati - 130050054

Previous Semester

- Transitive closure in KBC evaluation
- Our evaluation protocols
- Order Embeddings (Vendrov+)
- Results
- Context Embeddings (Mikolov+, Pennington+)

Transitive Closure in KBC

- 82,192 relation triples in Wordnet
- Transitive closure gives 743,241 edges
- Small test set => high accuracy just by Transitive Closure



Evaluation Protocols

Protocol 1:

Let H be transitive closure of all hypernym pairs

1. (Randomly) split H into H-train and H-test
2. Set H-test ← (H-test - TC(H-train))

Protocol 2: Raw instances not transitively closed: ‘rawPosIsa’ & ‘rawPosSub’.
Sample ‘trainPosIsa’ and ‘trainPosSub’ from above and calculate closure.

if t1 sub t2: accept if not in ‘closedTrainPosSub’
if e isa t: if e isa ? is not in ‘trainPosIsa’, accept
Otherwise, for each e isa t1 in ‘trainPosIsa’,
if (t1, t2) in ‘closedTrainPosSub’, reject.
Otherwise, accept.

Order Embeddings (Vendrov+)

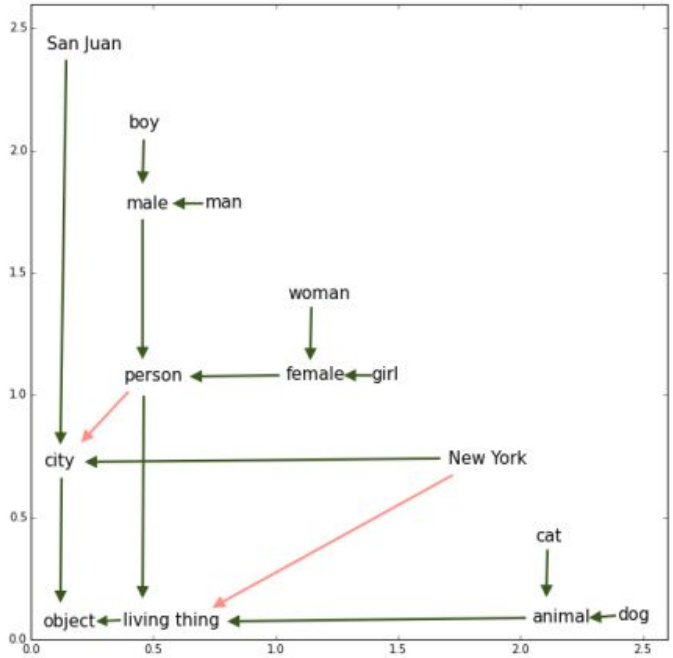
Spatial embeddings in positive orthant based on partial order.

Loss function:

$$\sum_{(u,v)\in WordNet} E(f(u), f(v)) + \max\{0, \alpha - E(f(u'), f(v'))\}$$

$$E(x, y) = ||\max(0, y - x)||^2$$

The space covered by each type is an open hyper-rectangle starting at the embedding itself.



Results (Wordnet)

Dataset: Transitive Closure(Wordnet)

Negatives: Socher's sampling

% test data Eval Protocol	1% test data OE Eval	15% test data OE Eval	15% test data Protocol 1 Eval
Precision	0.867	0.863	0.618
Recall	0.971	0.955	0.837
F1 score	0.916	0.907	0.711
0/1 accuracy	0.911	0.902	0.660

Results (DBpedia)

Dataset: Transitive Closure(DBpedia)
Negatives: Socher's sampling
15% test data

pos:neg split Dataset	1:1 split Wordnet	1:0.1 split DBpedia	1:1 split DBpedia	1:10 split DBpedia
Precision	0.618	0.972	0.919	0.844
Recall	0.837	0.989	0.921	0.796
F1 score	0.711	0.980	0.920	0.819
0/1 accuracy	0.660	0.964	0.920	0.968

Context embeddings (Mikolov+, Pennington+)

Word2Vec (CBOW)

- Predict focus given context
- Minimize ‘distance’ between embeddings for better prediction
- Uses Feed-forward neural network

“Efficient Estimation of Word Representations in Vector Space”
Tomas Mikolov, et al.,
CoRR 2013

GloVe

- Creates co-occurrence counts
- Factorize to get embeddings

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2$$

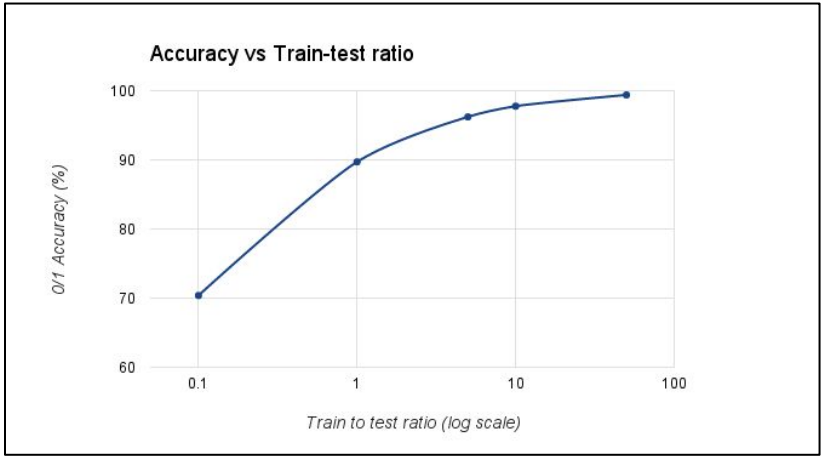
$$f(x) = \begin{cases} (x/x_{\max})^{\alpha} & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases}.$$

“GloVe: Global Vectors for Word Representation”,
Jeffrey Pennington, et al.,
EMNLP 2014

Evaluation - Baselines

First baseline (bounding box):

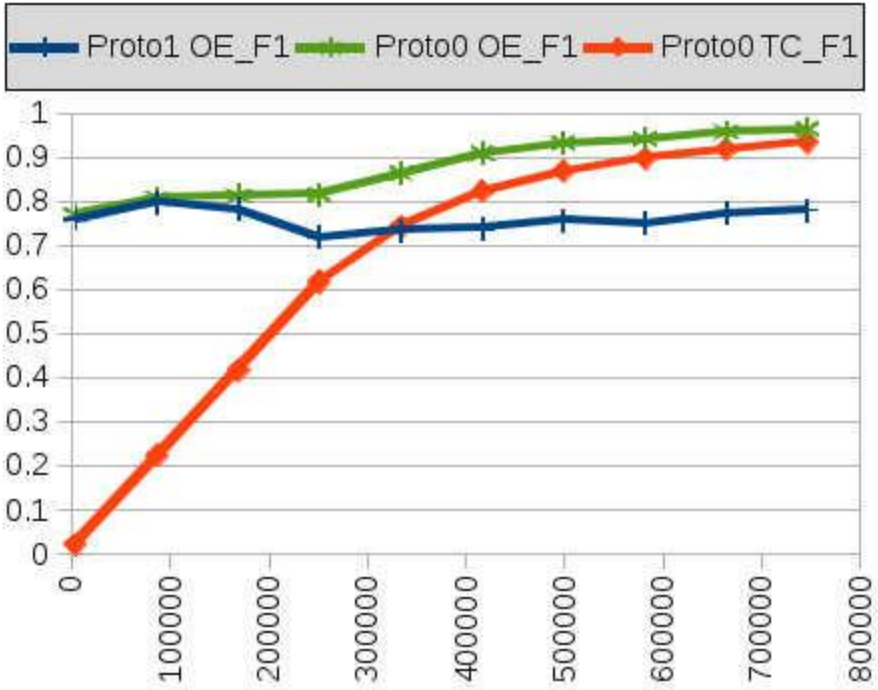
- ➢ Clean and process Wikipedia dump
- ➢ Train embeddings for all entities
- ➢ Split into train and test
- ➢ Learn types from the train set
- ➢ OE style evaluation



Second baseline (optimizing F1):

- ➢ precision typically very low (~10^{-3})
- ➢ recall is 1 (by definition)
- ➢ types with >500 entities have ~1.2M entities out of total 1.3M

Final inferences from OE



Other types of bounding boxes?

- Learning the type boundaries (as variables of the optimization process) using training loss.
- Robust bounding: In between ‘Complete’ bounding boxes and optimizing for F1 for each type F1 is maximised

Other types of bounding boxes?

Middle ground between ‘complete’ bounding boxes and leaving out entities to optimize on other metrics such as F1 at a global level.

Algorithm:

- Start from any one of the n dimensions - say d1.
- Consider top 5 candidate boxes that optimize the F1 score in d1
- Consider another dimension - say d2
- Find the top 5 candidate boxes with the candidate end-points of the box in the first dimension limited to the top 5 found already.
- Repeat over the remaining dimensions to find the end-points of the bounding box in all the remaining dimensions

Other types of bounding boxes?

Essentially tries to optimize for F1 score on individual dimensions

- F1 scores ~ 10^{-2} on standard GloVe/Word2vec embeddings
- Indication that standard word2vec and glove won't help us out in our case
- This led us to move to L2 similarity based embedding
  - Would help to capture the corpus into OE scenario when evaluated with F1 maximising Robust Bounding

Learning type boundaries

- Would be dependent on the position of the positive and negative entities from boundary of types

L2 embeddings

- Intuitive for Entity-Typing Tasks
- Not Standard/Established as word2vec/GloVe

Word2vec -> dot to L2

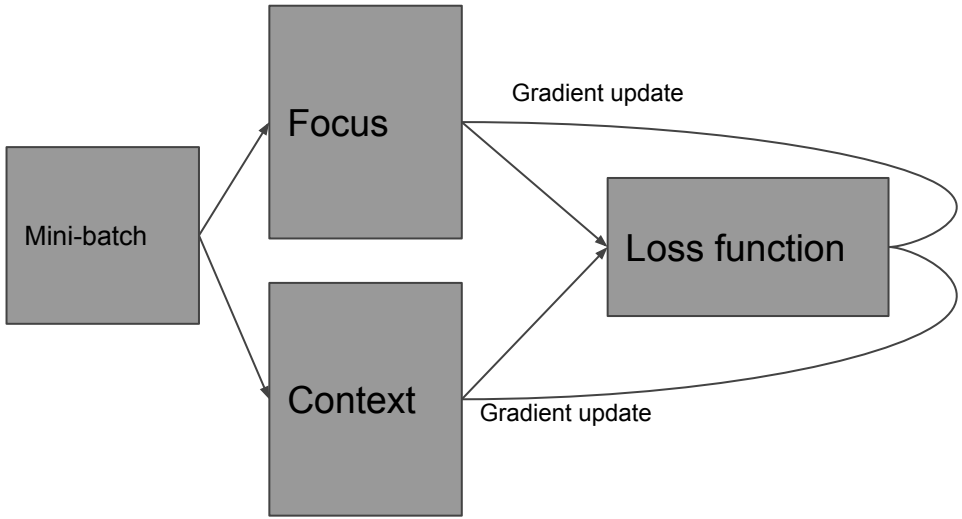
- Standard Word2vec (Mikolov et al) is a cosine distance based approach
  - Training
  - Evaluation
    - Analogy
    - Spot Checks
- We are going to use standard skip-gram negative sampling model as our base

- Loss function:

$$\text{argmax}_\theta \sum_{(w,c)\in P} \log\sigma(v_c \cdot v_w) + \sum_{(w,c)\in N} \log\sigma(-v_c \cdot v_w)$$

P is the set of positive examples, N is the set of negative examples

Word2vec -> dot to L2



- Unigram Negative-Sampling
- An equivalent loss can be emulated using cross_entropy and noise contrastive estimation loss

Word2vec -> dot to L2

- Need for L2:
  - Entity Typing Tasks rely heavily on a good representation of a given type which can cover all the entities belonging to the particular type
  - OE uses partial order violation in KG to obtain their current Entity-Typing system
  - Partial Ordering gives an ordering relying on L2 distance - evident from their loss
  - Can cosine based embeddings go hand in hand with OE for a better type system representation?
- We need to have a transformation of the cosine based embeddings to L2 based similarity in order to exploit the joint advantage of KG and Corpus
- Do we have a transformation?

Word2vec -> dot to L2

- dot(v, w) -> radius - l2(v, w)
  - High dot -> close -> low l2
  - Radius is a param, v, w are embeddings
- Transformation?
  - There do exist transformations for the task
  - Approximation allows us to transform radius-l2(v, w) to dot(v, w) in higher dimensions
    - Using Kernel Tricks and Mercer Kernels
    - Approximate inverse transforms do exist
- We went ahead with actually training L2 based embeddings instead of transformation

Word2vec -> dot to L2

- Engineering of L2 embeddings:
  - We use the same batching and sampling strategies as dot based approach

    $$Pr(v, w) = \log\sigma(v \cdot w) \rightarrow \log\sigma(radius - l2(v, w))$$

  - Loss changes accordingly and an equivalent loss can be emulated using cross_entropy and noise contrastive estimation loss
  - Radius is a trainable parameter which will show the movement of embedding during training and help in estimating final structure
- Evaluation:
  - Not as simple as dot based evaluation
  - Haven’t been able to capture the analogy function as in the case of dot

Word2vec -> dot to L2

- Analogy of L2: a1:b1 :: a2:?(b2)
  - Dot has the analogy as
    argmax_{b2} b2.(a2+(b1-a1)) -> argmax_{b2} cos(b2, b1) - cos(b2, a1) + cos(b2, a2). - >3CosAdd
  - Levy and Goldberg proposed a new analogy function for dot as
    3CosMul -> argmax_{b}(cos(b2, b1)*cos(b2,a2))/cos(b2, a1)
  - We tried to obtain a function which might capture analogy (similar to b2 = a2+(b1-a1))
- Our Proposals:
  - argmin_{b2} |(b2 - a2) - (b1 - a1)| - Base Approach
  - argmin_{b2} |b2 - b1| - |b2 - a1| + |b2-a2| - 3DistAdd
  - argmin_{b2} (|b2 - b1|*|b2-a2|) / (|b2 - a1|) - 3DistMul
- None of the above were able to capture the analogy for L2 embeddings as in case of dot

Word2vec -> dot to L2

- Spot Checks for L2 were not as satisfactory as expected:
  - These capture the closeness of similar entities which we intend to exploit for Entity-Typing
  - Still working on understanding the dynamics of L2 to get the required results
- Currently have 2 implementations:
  - C++ implementation - Prof. Soumen Chakrabarti
  - TensorFlow implementation
- But still L2 embeddings have the characteristics which can enhance Entity-Typing when coupled with OE
- We do a joint training of OE and L2 word2vec to capture corpus information for KG based entity-typing system

Joint Training

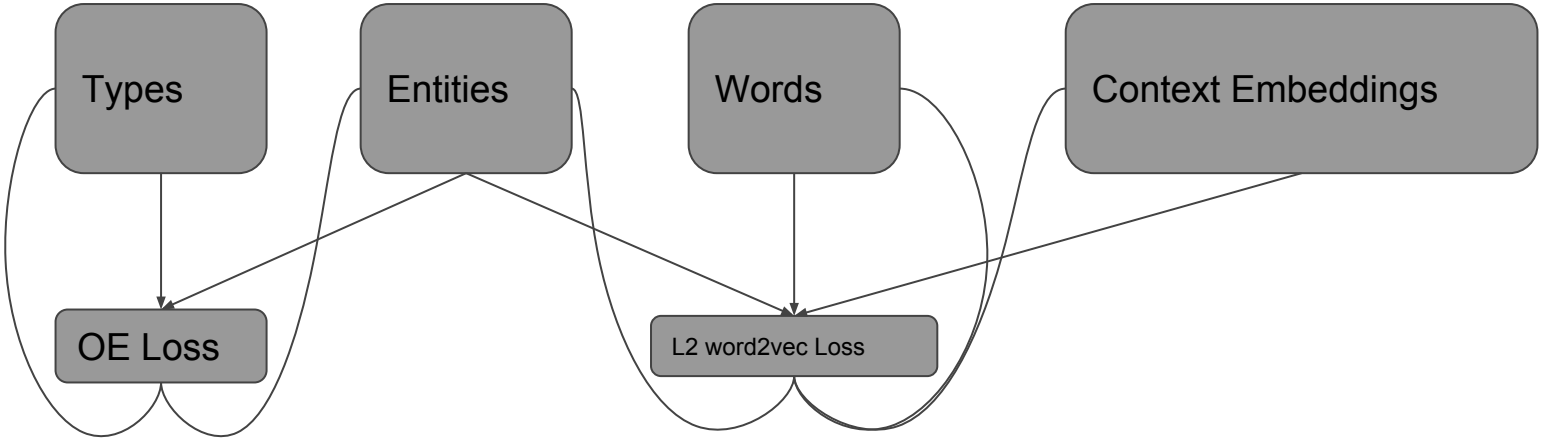
- Intertwined learning of OE and l2 word2vec
- Tight(Efficient) Type boxes
- Further use cases

Joint Training

- Intertwined training using OE and l2 word2vec
- We feed DBPedia as KG and Wikipedia as Corpus:
  - DBPedia has 4.58 m entities and 716 types
  - Most of the wikipedia entities are present in DBPedia
  - 17GB Wikipedia corpus
- Framework:
  - Coded entirely in Tensorflow
  - OE has 2 implementation:
    - In Lua by Ivan Vendrov
    - In Tensorflow by us (Fully Scalable)

      Both of them produce exactly same results
  - We use L2 word2vec in tensorflow

Joint Training - Framework



- All the curved lines are gradient updates
- Entities, Types, Words and Context all have same embedding size
- In our implementation we safely assume that KG has all the relevant Entities and rest or the corpus i then parsed to form words

Joint Training - Implementation and Constraints

- TensorFlow implementation had many conditional evaluations based on hyper batching
- A complication conditional tf graph was created and the learning goes on using the inherent properties of mini-batches
- TensorFlow has core problems when it comes to Conditional Graphs with respect to various aspects and it is non-trivial
- Learning TensorFlow and getting to the low level engineering wwas required to get things fall into right place

Joint Training - Scheduling

- We intertwine both the schedules
  - OE
  - Word2vec

  One after another after one single epoch/parse through KG/corpus
- Initially, we warm up the system with a word2vec epoch so that words, contexts, entites are initialised decently
- Then run the following pattern OE followed by Word2vec and finally end things with OE so as to minimise the partial order violations
- This gives us the required closeness based similarity from L2 over and above the power of OE

Joint Training - Current State and Evaluation

- We have a complete working implementation of the above framework
- Word2vec scales in great fashion, but the OE schedules are not scaling currently. We are trying to get it to scale in-order to get the final results as it will take ~6hrs to get results with scaling for word2vec and same goes with OE, when scaled across 20 cores. Thus it is tough to obtain results without scaling
- The evaluation is based on same strategy as OE, hide some amount of KG and check the accuracy of them with the Type Bound Boxes
- Other metrics are recall, precision and F1 of each Type box after training and ensure we have maximum F1

Joint Training - Evaluation

- Type Bounding boxes can be done in two ways as discussed earlier
  - Robust Bounding
  - Learning of Bounding using certain training loss based on distance from the type box for each positive/negative entity
- As we are currently working on correctness of L2 embeddings and scaling in a parallel fashion, one both of them have given clear positive indications, we shall go ahead with getting out results and evaluation

Apart from the above things, we tried to get explicit embedding based on attributes from Freebase and tried to prove that we can assign literal meaning to dimensions in Embeddings

Further and Future Work

- Get good L2 Embedding and results corresponding to them
- Scale Joint Training to get results
- Prove Corpus allows to better OE results
- Work on Subspace problem for disambiguation and Fine-type Tagging

Thank you