Efficient spatial representation of Entity-Type system

Undergraduate Thesis: Part I

Aditya Kusupati - 130050054
Anand Dhoot - 130070009

Guide: Prof. Soumen Chakrabarti

Knowledge Base Completion (KBC)

Knowledge Bases have never been robust and complete. As time progresses and due to the need, completeness of Knowledge Bases is a necessity. Given the importance of problem there had been multiple approaches towards it, trying to solve:

From the existent Knowledge Base and any other information pertaining to the entity types, we have to predict whether new relation/triple is valid and has to be a part of KB. So learning and discovering new (ent isa type) and (type1 subtype type2) triples is vital for KBC. The work done goes on the similar lines.

Transitive Closure (TC)

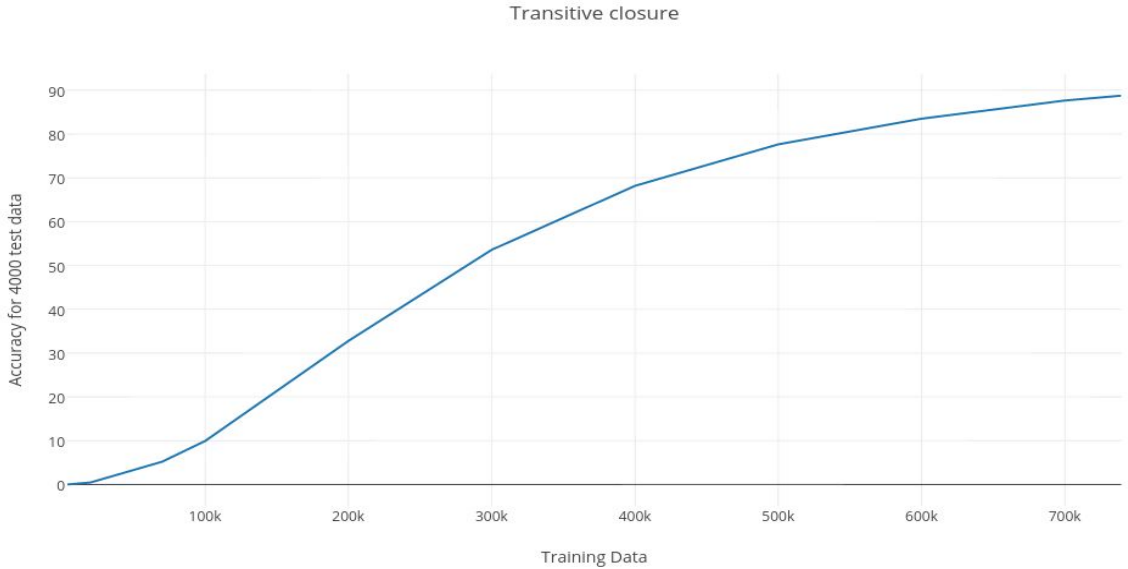
Transitive Closure is simple calculation of all the triples possible by using the transitive property of entailment of entities and types.

e1 isa t1 and t1 subtype t2 are part of KB => e1 isa t2.

This property is pretty powerful and can be used for basic KBC.

Transitive Closure (TC)

- Wordnet has 82192 relation triples. Transitive closure gives 743241 edges over entire wordnet.
- Vendrov+ split TC(wordnet) into test and train to predict effectiveness of TC (and OE)
- Negatives have no effect when evaluated by TC, as they are never covered
- For a test set of 4000 edges, TC(train) gives accuracy of 88.7 using all else for training



Deep Learning for TC?

KBC may involve

- Trivial TC computation for which (deep) learning is not needed
- Nontrivial entity and type embeddings based on other properties of KB or corpus

Evaluation scheme should separate reward for computing TC from reward for more nontrivial predictions.

We propose two new protocols for evaluation to remove the effect of TC on Deep Learning.

Baseline accuracy drops substantially on removing TC reward!

Evaluation Protocol 1 (EP1)

- Let H be all the hypernym pairs, i.e., (hypo, hyper) instances
- H ← TC(H)
- (Randomly) split H into H1 and H2
- H1 is the train data, now modify H2 such that TC(H1) has no effect on H2 while testing.
- H2 ←(H2 - TC(H1)), H2 is the final test data now.

Simple evaluation based out of TC gives out 0% accuracy for any test train split in this protocol. Thus TC is not rewarded here by construction.

- Targeted for WordNet; Wikipedia/Freebase has more asymmetry between the type layers and the final entity layer
- No test of the ability of an algorithm to reject false claims
- Next protocol …

Evaluation Protocol 2 (EP2)

Raw instances are pairs **rawIsa** = { e isa t } and **rawSub** = { t1 sub t2 }. Not (transitively) closed. No useful notion of closure associated with e isa t alone.
These are both positive, i.e. can call them **rawPosIsa** and **rawPosSub**.
Sample **trainPosIsa** and **trainPosSub** from above. trainPosSub likely to be a much larger subset of rawPosSub.
closedTrainPosSub ← TC(**trainPosSub**)
Now given a proposed test instance, which could be e isa t or t1 sub t2.

t1 sub t2: accept if not in **closedTrainPosSub** (this is for testPosSub)

e isa t2: if e isa ? is not in trainPosIsa, accept (this is for testPosIsa)
Otherwise, for each e isa t1 in trainPosIsa,
    if (t1, t2) in closedTrainPosSub, reject.
    Otherwise, accept.
This gives testPosIsa and testPosSub. Even here TC is not being rewarded by construction.

Negative Samples

We have looked at obtaining positive Samples till now.
We also need negative samples to check that algorithms can reject false claims.

Currently, we rely on two mechanisms for negative sampling:

1) Socher's sampling (Random perturbation of one of the entities/types in the triple)
2) Negative generation from disjoint sub trees in the KB/KG like DBPedia

We shall be using these to check the negative classification of our algorithm.

Order Embeddings

- Exploit existing hierarchy in language
- Learn from structured text (KGs)
- Visual ease in representation, understanding and extrapolation

Vendrov, Ivan, et al. "Order-embeddings of images and language." *arXiv preprint arXiv:1511.06361* (2015).

Algorithm

Order Embeddings relies on learning a 50 dimensional spatial embedding in positive orthant.

The spatial embeddings are learnt based on the partial order (entailment) and order violation.

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

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

E(x, y) = 0 => order is satisfied for (x, y) else E(x, y) > 0

E is our order-violation penalty, and (u' , v') is a corrupted version of (u, v) (neg sample by Socher) . Since we learn an independent embedding for each concept, the mapping f is simply a lookup table. f(x) -> embedding of x in 50 dimensions.

Representation and intuition

The origin contains the type which encloses everything and rest of the entities/types eventually entail to that type.

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

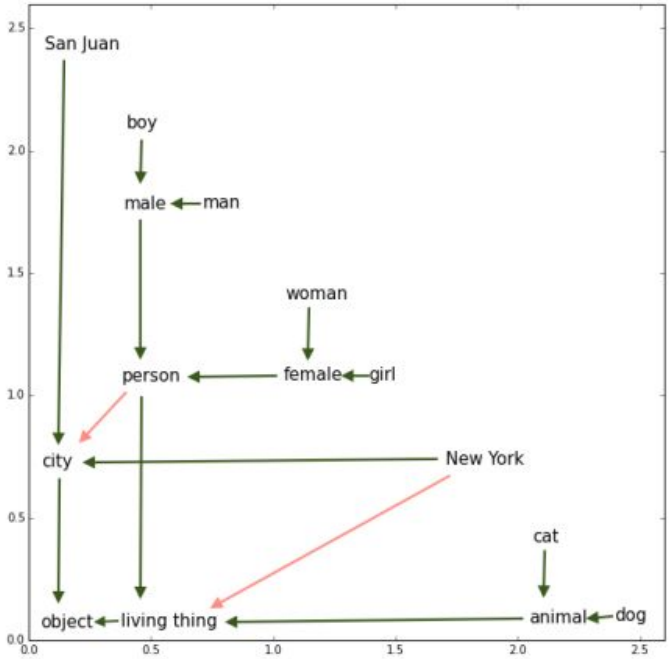
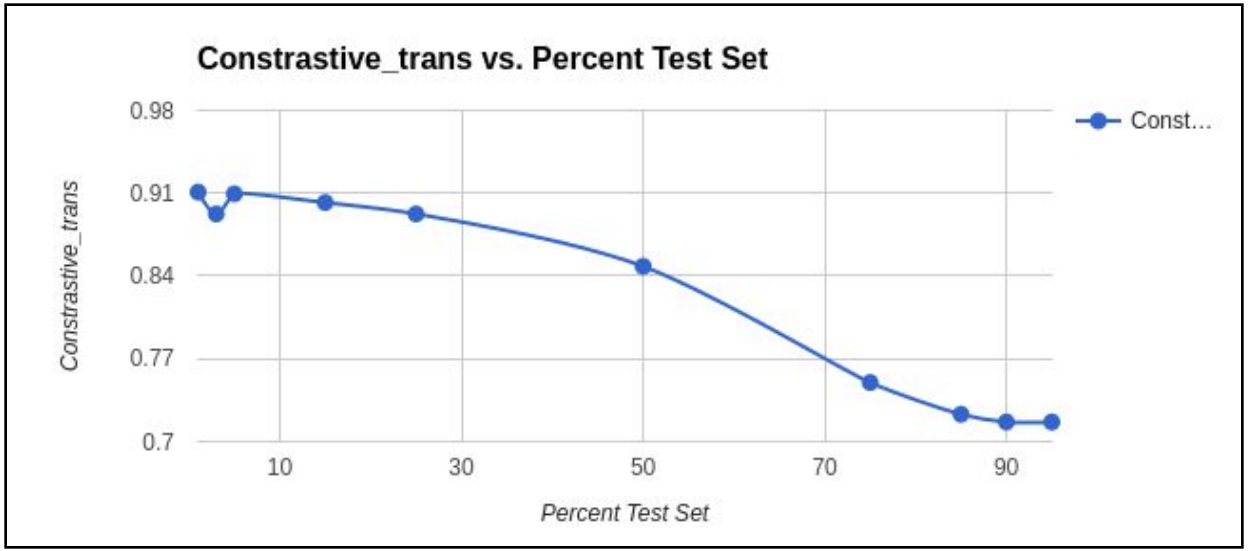


Figure 2: 2-dim order-embedding of a small subset of the WordNet hypernym relation. All the true hypernym pairs (green arrows) are correctly embedded, but two spurious pairs (pink arrows), are introduced. Only direct hypernyms are shown.

Results


Constrastive_trans vs. Percent Test Set
0.98
0.91
0.84
0.77
0.7
Constrastive_trans
10
30
50
70
90
Percent Test Set
Const…

Results

On Wordnet (transitively closed)

With 1% test data and rest as training (Negative sampling as per Socher)

Precision	0.867
Recall	0.971
F1 score	0.916
0/1 accuracy	0.911

Results

On Wordnet (transitively closed)

With 15% test data and rest as training (Negative sampling as per Socher)

Precision	0.863
Recall	0.955
F1 score	0.907
0/1 accuracy	0.902

Results

On Wordnet (transitively closed) using EP1 and EP2 (EP2 mimics EP1 in terms of results)
With 15% test data and rest as training (Negative sampling as per Socher)

Precision	0.618
Recall	0.837
F1 score	0.711
0/1 accuracy	0.660

This clearly shows that TC is affecting order embedding accuracy drastically.

More Results on DBPedia

DBPedia is huge, with 15 mn positives the training will make the model fit most of the embeddings to get the best possible accuracies, but there shall be a dent in F1 scores.

The OE paper has assumed a 1:1 pos:neg scenario in test and train. In the real world it is likely to be 1:x {x>1} and we generated values for 1:0.1, 1.05, 1:1, 1:2, 1:5, 1:10

We can clearly see a trend in 0/1 accuracy and F1 in the results obtained. For simplicity we have used EP1 everywhere for comparison.

More Results on DBPedia

pos:neg	Pre	Recall	F1	0/1 acc
1:0.1	0.971757829	0.989291932	0.9804464927	0.96412739
1:0.5	0.9367525612	0.9532707132	0.9449394557	0.9259386407
1:1	0.9194504466	0.9215423468	0.9204952082	0.920404765
1:2	0.9086531992	0.8773589332	0.8927318992	0.9297193804
1:5	0.87448542	0.8331983796	0.853342797	0.9522682856
1:10	0.8440309864	0.7961075571	0.8193691312	0.9680903949

F1 score gradually decreases as pos:neg ratio decreases.

Drawbacks

- Use of TC in deep learning ie., flawed evaluation protocol
- 1:1 pos: neg ratio
- Trains on high amount of train data and testing is done on very less test data. Ideally we intend to use minimum train (small in size) and try to extrapolate the KG by running the model on test data (large in size)
- No use of context in whatsoever scenario to get better embeddings for the entity-type system
- Many disjoint or irrelevant hyper rectangles/cones intersect in space giving an illusion of wrong entity-type relationship

Context Embeddings

Need for context embeddings -

- Incomplete knowledge-bases
- Uses unstructured text
- Polysemantic words & entities

Word embeddings

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

Word embeddings

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

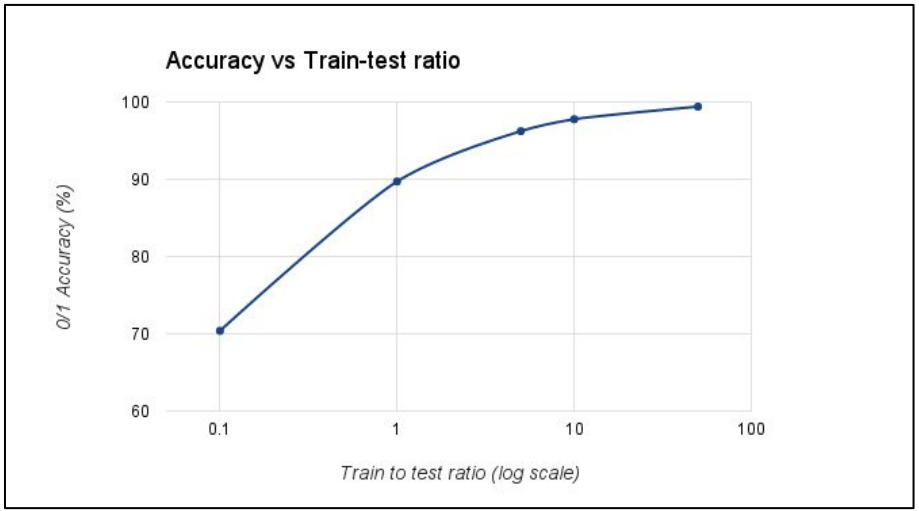
$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2$$

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

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

Evaluation - First Baseline

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



Evaluation - Second Baseline

For every type present,

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

Intricacies & Possible Improvements

When to do the Train-test split? Just before creating

- Type-space over embeddings
- Word embeddings?
- Corpus-space?

Intricacies & Possible Improvements

When to do the Train-test split? Just before creating

- Type-space over embeddings
- Word embeddings?
- Corpus-space?

Learning types for entities

- Open rectangles
- Non-negative embeddings?
- Open cones?
- Bounded rectangles?

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

GloVe's loss function

Intricacies & Possible Improvements

When to do the Train-test split? Just before creating

- Type-space over embeddings
- Word embeddings?
- Corpus-space?

Learning types for entities

- Open rectangles
- Non-negative embeddings?
- Open cones?
- Bounded rectangles?

Finding rectangle/cone vertex

- Strict bounding
- Robust bounding?

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

GloVe's loss function

During next semester!

- Joint training on KG and corpus
- Fine type tagging

Thank you