Efficient Spatial Representation for Entity-Typing

B.Tech Project Report

Aditya Kusupati 130050054

Anand Dhoot 130070009

Guide: Prof. Soumen Chakrabarti

May 25, 2017

Abstract

The project aims at creating a efficient spatial embeddings for entities and types which would be useful for various downstream tasks such as Knowledge Base Completion, Fine-Type Tagging and Question Answering.

1 Introduction

Entity Typing has been a research topic of interest in recent past. Many attempts have been made at obtaining a robust entity type system. They range from use of Knowledge graphs (KG) to a mixture of Corpus with KG. Nonetheless, visualizing Entities and Types as spatial embeddings can turn out to be a powerful tool for a better Entity Type system. We propose to use the signals from KG as well as the corpus to learn the embeddings as desired.

2 Knowledge Base Completion (KBC)

Incomplete Knowledge Bases have the necessity to be complete for the greater good of the research community. A lot of Knowledge Bases like Wordnet, DBPedia are not complete even after decades of usage. Being a tedious manual task, this has to be automated and the probable entries should be accepted

or rejected automatically. Research has been done and and was successful on various fronts.

KBC at its core is predicting whether a given (*e isa t*) or (*t1 sub t2*) belongs to the Knowledge Graph ie., whether they are true. This is done based on learning from various sources like existing KG, context information from corpus with the current entities.

3 Order Embeddings

Our primary investigation started with the paper on Order Embeddings. The approach was proposed by Vendrov, Ivan, et al.[1]. The paper aims to exploit existing hierarchy in language (KG) which could be extended to images as stated in the paper. We shall discuss only about the text/language part in this project.

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 of the entity-type and type-type pairs from the KG.

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

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

$$E(x, y) = 0 \iff x \preceq y$$

Figure 1: OE's loss function & Order penalty

P is the set of positives and N is the set of Negatives. E is our order-violation penalty, and (u', v') is a corrupted version of (u, v) (negative sample by Socher). Since we learn an independent embedding for each concept, the mapping f is simply a lookup table. f(x) $\rightarrow$ embedding of x in 50 dimensions. α is a threshold margin parameter for negative samples.

3.1 Paper's Evaluation

KG used was Wordnet with 82,192 relation triples. They had a primary baseline from simple transitive closure computation. With the transitive closure Wordnet spans over 743,241 edges. They chose 4,000 edges as test and rest

as train and by simple check in the Transitive closure of training set, test set was classified.

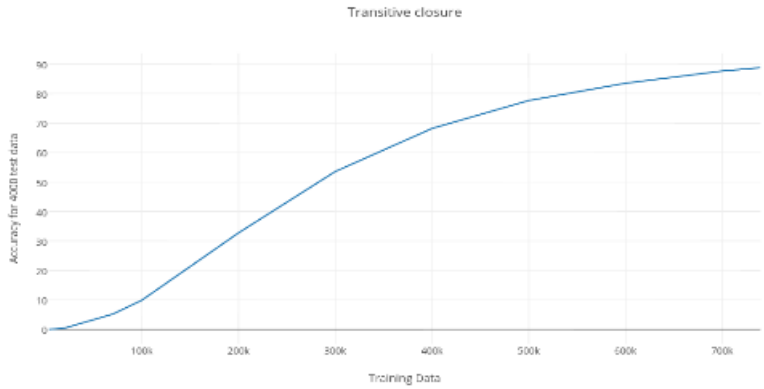

Figure 2: Accuracy when varying Train set size

While learning order embeddings their evaluation and training protocols do not remove the effect caused by transitive Closure. They effectively use Deep Learning to learn a straight forward and a powerful feature like transitive closure and are heavily benefited by it. We must use deep learning only to predict 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!

3.2 Proposed modifications

3.2.1 Two new protocols for evaluation to remove the contribution of Transitive Closure

Protocol 1: Let H be the transitive closure of all hypernym pairs, i.e., (hypo, hyper) instances

- (Randomly) split H into H-train and H-test
- H-test ← (H-test - TC(H-train)), the final test data now

This ensures that contribution of Transitive Closure for any test-train split would be 0% (by construction).

Protocol 2:

- 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* $\leftarrow$ 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*,
    - (t1, t2) in *closedTrainPosSub*, reject
    - Otherwise, accept

This gives *testPosIsa* and *testPosSub*. Even here TC is not being rewarded by construction. When the above protocol was run for 15% as test the obtained F1 score was 0.711 with an accuracy of 0.660 whereas the older protocol gave F1 score of 0.907 and accuracy of 0.902. This clearly shows that Transitive closure was playing a major role in the obtained accuracy.

3.2.2 Negative sampling and pos:neg ratio

The paper uses Socher, et al. [2] sampling for negative generation and pos:neg ratio is set to default 1:1. But ideally by empirical experiments the pos:neg ratio is more likely to be around 1:5. As we proceed increasing pos:neg ration from 1:0.1 to 1:10 we had a drop of F1 score from 0.98 to 0.82. This trend shows that F1 score at a more realistic pos:neg ration can be improved.

3.2.3 Changing the KG

Given the limited nature of Wordnet we moved to DBpedia for a larger KG and with a backing of Wikipedia corpus. 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. DBpedia graph by virtue gives us disjoint relation from which we can generate 450 mn negative samples which can be used as part of our earlier protocols.

4 Dissecting Order Embeddings

4.1 What is the lift of Order Embeddings over Transitive Closure?

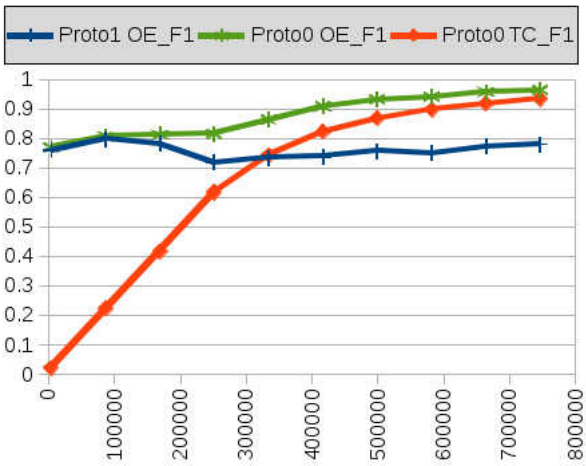

Figure 3: The dissection of OE

The graph in Figure 3 has some interesting inferences with respect to the algorithm. The inferences, along with our hypotheses for the same are mentioned below.

- Transitive closure has a significant contribution to the results of the base evaluation protocol.
- For small sizes of the training set, Order Embeddings perform way better, even without the help of Transitive Closure, We hypothesize this to the skewed sampling (from the transitive closure of all hypernyms, discussed below)
- The gain obtained by OE over TC isn't significant but is useful and we predict corpus can enhance it further.

4.2 Sampling: Problems

Sampling, both negative sampling for training and the sampling of data for testing, are skewed in most of the approaches we use. This is in contrast to the ideal sampling which asks for non-significant overlap and effect of training on test data. For example, Levy, et al.,[6] states that rather than

actual learning of the semantics there will be lexical memorization for a skewed training data which is from the standard sampling strategies. We are yet to investigate into ideal graph and spatial sampling techniques which can circumvent these problems. Backstrom, et al., [7] seems to have a good approach in this direction which can be investigated further.

5 Context-embeddings

The section above talked about learning embeddings of entities (and types) from the information present in a Knowledge Base(KB). While a KB does provide lot of information in a structured format, it may often be incomplete. Additionally, there may be entities in plain text such as 'bank' which have several semantic interpretations.

Plain-text can also be used for learning embeddings. We investigate two techniques to do this and evaluate the embeddings generated using the evaluation protocols specified above.

5.1 Previous Work

5.1.1 Word2vec - Continuous Bag of Words model (CBOW)

In this approach proposed by Mikolov, et al.,[3] the task of creating word-embeddings is performed using a feed-forward neural network.

Given a corpus of text, the technique involves iterating over all words, considering instances of 'focus' and 'context' (words appearing within a threshold distance of the focus word in a sentence) words and training their embeddings such that two-words co-occurring frequently would have a high cosine similarity between their embeddings.

5.1.2 GloVe

In a way very similar to the CBOW model above, GloVe[4] (short for Global Vectors) considers a focus word and some context words around the focus word. However, it creates a large co-occurrence counts for every such set of focus and context words.

Once the entire corpus is processed, this large matrix of co-occurence counts is factorized to train the word embeddings. The loss function used is presented below. w_i and w_j are the embeddings of the words i and j and X_{ij} is

their co-occurrence count. $f(x)$ is a squashing function that helps increase the contribution of pairs with small number of co-occurrences.

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2 \qquad f(x) = \begin{cases} (x/x_{\max})^\alpha & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases}.$$

Figure 4: Glove's loss & squashing function

The loss function can be interpreted to be trying to minimize the difference between the dot product of the two embeddings and their log co-occurrence counts, modulo a constant factor dependent on the pair of words being used.

We use GloVe to find the word-embeddings for all the experiments described below.

5.2 Implementation details

We use the Oct 2015 Wikipedia dump and preprocess it for creating the embeddings. This involves discarding everything except the plain text and annotations (to indicate the entities) from the dump downloaded, resolving redirects, removing stop-words & punctuation marks before passing on to the GloVe model.

5.3 Evaluation & Results

We create hypernyms using DBpedia types for the Wikipedia entities and split the set into train and test. Further, we use the type structure imposed by DBpedia to compute type bounding boxes for evaluation.

To ensure that our system performs an Order-Embeddings style evaluation, 'open-rectangles' corresponding to each type were inferred from the entity embeddings. This was done by finding a point (the apex of the open rectangle) in the embedding space for each type such that all entities of that type are considered to be belonging to that type.

For the evaluation, we find the 0/1 accuracy for the hypernyms in the test set and observe the results in Figure 4 as the test-train split is varied.

The numbers are much better than the previous result and are too good to be true. Therefore, we perform yet another evaluation - this time finding out the precision of entities getting captured by type. Note that every type 'contains' all entities of its type, meaning that recall is 1 by definition.

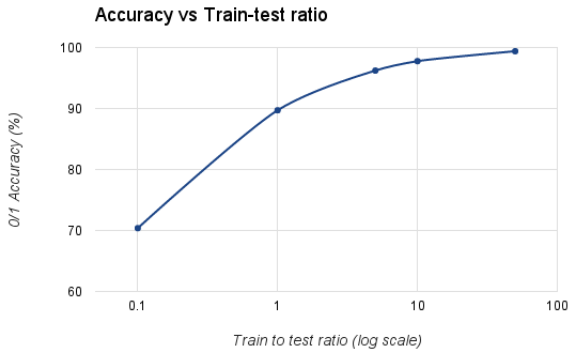

Figure 5: 0/1 accuracy vs test-train split for context embeddings

We observed that nearly every type, precision was very low (of the order of 10^{-3}). This means that for most types, only 1 in 1000 entities present actually belong to that type! Further investigation showed that nearly every type captures approximately 1.2 million entities out of a total of 1.3 million extracted from Wikipedia.

We hypothesize that this is because out of the several entities belonging to a type, there is always some entity that has a large negative value in every dimension. When calculating the apex point for the type open-rectangles, this value gets picked and hence, (nearly) all of 1.3 million entities satisfy the containment relation.

6 Type Representation

Having a type representation in the same spatial domain as the entity embeddings is required to perform the aimed tasks in an efficient fashion. The OE paper has the open (hyper-)rectangles with apex fixed at the learnt Type Embedding, these open-rectangles actually enclose the space and entail the partial order as in their hypernymy relation. We state that there are two plausible ways for the bounded Type representation.

- Learnt type boundaries based on distance metrics
- Robust Bounding boxes, in between "Complete" bounding boxes and learnt bounding boxes - deterministic in nature

6.1 Learnt type boundaries

This basically involves training the type bounding boxes as a variable during the training process and optimizing with the loss function. In order to do this, the type-embeddings (or the end-points of the type bounding box) must be trained along with embeddings of the entities.

This presents several challenges, the most important of which is scheduling & interleaving of the batches that perform updates to entity embeddings and those updating the type embeddings. More about this in the subsequent section(s).

6.2 Robust bounding boxes

Given the entity embeddings this aims to find the type-embeddings such that for each type, the F1 score of containment of entities in the type bounding box is maximized. However, since this maximization is expensive, a modified algorithm is used where the search for the optimal bounding box is done dimension wise.

This approach is different from the learnt bounding boxes as well as 'complete' bounding boxes which focus on ensuring recall = 1. The algorithm implemented to find the robust bounding boxes of every type was as below:

- Start from any one of the n dimensions - say d1
- Consider top 5 candidate bounding boxes that optimize the F1 score only in dimension d1
- While another dimension remains, consider another dimension - say d2
- Find the top 5 candidate boxes with the candidate end-points of the bounding box in the 'explored' dimensions limited to the top 5 found already.
- Repeat till you run out of dimensions

The above steps ensure that we try to capture the best possible (resulting in the highest F1 score) bounding box as we progress over dimensions, while not remaining myopic enough to pick just the best one in each dimension.

However, the F1 scores obtained on the standard GloVe/Word2vec embeddings using the above technique were of the order of 10^{-2}. This was an indication that probably, standard Word2vec and GloVe won't help us out in our case and we would have to move to L2 similarity based embeddings, explained in the subsequent section.

7 L2 Embeddings

As Order Embedding tries to exploit partial oder by enclosure and entailment, it is reliant on L2 distance. We need to get contextual similarity to couple with the entailment, thus the closeness metric for the contextual embeddings should be L2 rather than cosine as in many standard implementations like Word2Vec and GloVe. We thus propose mechanisms to get the contextual embedding align with our L2 closeness requirement

7.1 Overview of Word2Vec: Skip-Gram Model

Given a sentence and a window k, we define the context of a word as its 2k neighbor words. Word2vec defines the positive dataset D from the above method. Each of these pair did appear in the dataset, so we associate them with a label 1. We now define the negative word pairs dataset D' and label these pair as 0, the negative sampling is based on unigram sampling with a tunable distortion parameter. The loss function for the standard word2vec is as follows

$$argmax_\theta \sum_{(w,c)\in D} log\sigma(v_w.v_c) + \sum_{(w,c)\in D'} log\sigma(-v_w.v_c)$$

The first term represents the Pr(w, c) which is the co-occurrence probability, to be high in case of $(w, c) \in D$ and to be low when $(w, c) \in D'$ and the loss function does the required. The loss function can be emulated using cross entropy and Noise Contrastive Estimation Losses which are coded up in the tensor flow implementation of the code.

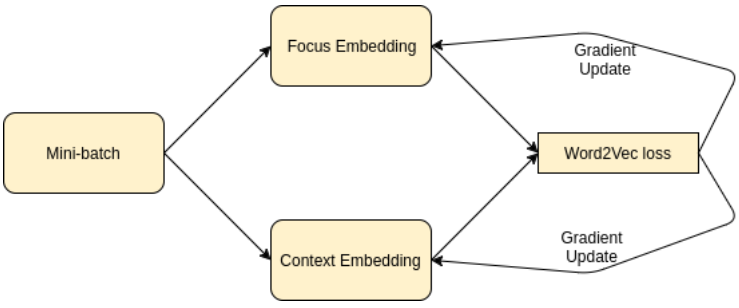

Figure 6: Flow Graph of Word2Vec

NLP tasks clearly prove that word when used as Focus (when used as the center for the window) and as Context (when in the 2k window of any given focus word) have different implications, thus there is a need to learn them separately.

7.2 How to get to L2 Embeddings?

Currently we seem to have found two viable options

- A transformation from Cosine Domain to L2 domain capturing the closeness metric accordingly
- Bottom-up building and learning of L2 based embeddings on parallels lines to the standard Word2Vec

We chose the second one given that we have a head start in the form of some existent code base in C++ and the we haven't completely understood or worked out the theory for the first option

7.2.1 Transformation?

$$v.w \longrightarrow r - ||v - w||_2$$

The above expression states that any cosine based similarity has a counter part in L2 domain which implies that dot and L2 are inversely proportional in terms of magnitude of closeness. r is a radius parameter which describes the entire structure of the learnt L2 embeddings. So the question is that whether there exists a transformation from cosine to L2.
Approximation allows us to find transform $r-||v - w||_2$ to $v.w$ in some higher dimensions

- Possible using Kernel Tricks and Mercer Kernels
- Given a forward transform, there does exist an exact/approximate inverse transform from dot domain to L2 domain

7.2.2 L2 Word2Vec

This is the train the model from scratch using a loss function on the similar lines of standard Word2Vec to get to L2 Word2Vec. Let us have a look at analogy between Standard Word2Vec and L2 Word2Vec.

$$Pr(w, c)_{(w,c)\in D} = log\sigma(v_w.v_c) \longrightarrow log\sigma(r - ||v_w - v_c||_2)$$

Using the above analogy and the loss function discussed earlier we can write the loss in terms of cross entropy and Noise Contrastive Estimation and will be ready to be trained. Radius, r, is a trainable parameter which will show the movement of embedding during training and help in estimating final structure.

7.3 Quality of Embeddings

Quality of Embedding is defined as the effectiveness/accuracy in performing designated tasks which are expected to be of semantical importance. They include

- Analogy Tests
- Top K nearest neighbors spot checks

Analogy test cases are basically of the type $a1 : b1 :: a2 : b2$ given 3 of the four, we should be able to predict the fourth element. Eg., Rome:Italy::Paris:?, our test need to give out France as the answer.

The top KNN spot checks are to ensure that the co-occurrence is being accounted for and similar(contextual) elements do come together after training.

7.3.1 Analogy Tests

We shall compare the evaluation expressions in both Dot and L2 word2vec and get to their results. Expressions for Standard Word2Vec analogy are from Levy, et al.,[8]:

- $3CosAdd \longrightarrow argmax_{b2}(cos(b2, b1) - cos(b2, a1) + cos(b2, a2))$
- $3CosMul \longrightarrow argmax_{b2}(cos(b2, b1) * cos(b2, a2))/cos(b2, a1)$

3CosMul is a better analogy expression than 3CosAdd, 3CosAdd is trying to simply maximize the dot product with the aligned vector, as 3CosMul is basically a *log* squashed version of 3CosAdd where the high frequencies are toned down such that they don't affect the entire expression and change the expected result. It is shown that 3CosMul outperforms 3CosAdd in all the datasets available currently for analogy tests.

Analogous to the above expressions, L2 Word2Vec tries to capture the analogy in the following fashion as proposed by us:

- $Base \longrightarrow argmin_{b2}||(b2 - a2) - (b1 - a1)||_2$
- $3DistAdd \longrightarrow argmin_{b2}||b2 - b1||_2 - ||b2 - a1||_2 + ||b2 - a2||_2$
- $3DistMul \longrightarrow argmin_{b2}(||b2 - b1||_2 * ||b2 - a2||_2)/||b2 - a1||_2$

Currently, none of the above 3 approaches are able to capture the analogy as good as the dot based word2vec. This empirically proved that the analogy relationship might be a higher order function which is not as simple as in earlier case and we need to find that to exploit the required feature out of L2 Embeddings

7.3.2 KNN Spot Checks

Spot Checks for L2 were not as satisfactory as expected when compared to dot based implementation and the C++ implementation of L2 word2vec

- 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/right results

7.4 Implementation of L2 Embeddings

We currently have two working and scalable versions of L2 Word2Vec.

- C++ implementation - Prof. Soumen Chakrabarti
- TensorFlow implementation as part of the project

Both of them have slightly different but equivalent losses and have a major difference in the value of radius after convergence. TensorFlow code give out a negative/small radius and C++ code gives a positive radius. Apart from that KNN Spot Checks are better in the case of C++ code when compared to TensorFlow code. We are looking into and are in the process of getting things right.

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

8 Joint Training

- Intertwined training of OE and L2 Word2Vec
- Tight(Efficient) Type boxes
- Further Use Cases as discussed above

As stated earlier, the synergy between these two settings needs to be exploited and one of doing them is an intertwined training of both the models within a common platform. We went ahead with this approach and have an working correct implementation of the same.

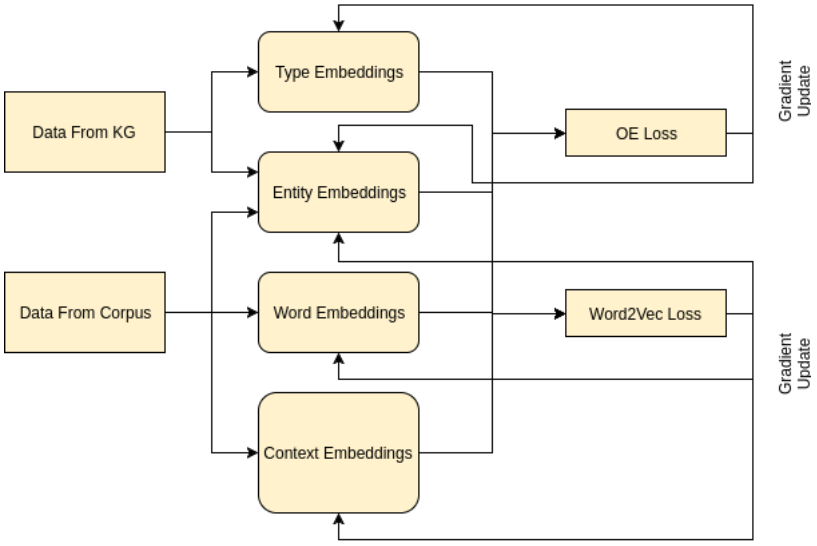

Figure 7: Flow Graph of Joint Training

8.1 Implementation of Joint Training

8.1.1 Assumptions

- KG and Corpus are duly matched with respect to entities (ie, the same entities are present in both; done in pre-processing codes)
- Types are solely obtained from KG, no relation with the corpus as usch
- Words are the rest of the vocabulary apart from entities present in the corpus
- All the 4 blocks have same embedding dimensions and also belong to the same space

8.1.2 Constraints

- TensorFlow implementation had many conditional evaluations based on mini-batching
- A complicated conditional TensorFlow 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 was required to get things fall into right place

8.1.3 Scheduling

We intertwine both the schedules of OE and L2 Word2Vec, one after another, after one single Epoch/parse through the entire KG/Corpus respectively. The major reasons for doing the scheduling in terms of intertwined epochs instead of intertwined mini-batching is that

- Mini-batching needs to be intelligent to cover the entire spatial distribution of all the embeddings, types, words
- Such an algorithm is expensive and choosing random mini-batching will result in attraction and repulsion of the embeddings in a haphazard and localized fashion which will affect the quality of the final Joint Embeddings Trained
- An entire Epoch assures a decent global update with respect to one model before we optimize for the other and so on
- Earlier we discussed about Learnt Type Boundaries which are trained along with the entities and type embeddings, this was avoided because of exactly the same reason as above as the global structure not being captured in mini-batches will result in localized optimization leading to heavy losses once the random batching hits the unnoticed points

Initially, we warm up the system with a word2vec epoch so that words, contexts, entites are initialized decently.Then run the following pattern "OE followed by Word2vec" and finally end things with OE so as to minimize the partial order violations. This gives us the required closeness based similarity from L2 over and above the power of OE

8.2 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 around 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 for evaluation are recall, precision and F1 of each Type box after training and ensuring that we have maximum F1

9 Other Investigations

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. Extracting the relevant attributes, numerics of each MID in freebase and encoding them appropriately in a higher dimensional space can help us learn the relation between attributes and the dimensions of the embedding space. On reducing dimensionality we might be able to prove that every embedding dimension represents a unique attribute and linear combination of these attributes can lead to a definition of a required type. Thus a new type system can be proposed in this lines.

A new evaluation protocol for most of the KBC tasks such that hierarchically top level types get lower weight in the calculation of accuracy ie., types with huge population of entities end up getting less score for their hypernym pairs in test and train and thus reducing the effect of humongous types which are repeated in the data many times resulting in the memorization as we discussed earlier. This needs to be taken forward and investigated further.

10 What next?

- Get good L2 Embedding and results corresponding to them, ensure the sync between C++ and TensorFlow Code, radius is ideally positive, need to look at the minute differences in the loss function of both the implementations and work on them to make the algorithm robust
- Scale Joint Training to get results by fixing the necessary threading or Loss Graph in TensorFLow
- Prove that corpus enhances and helps OE to perform far better under ideal/correct evaluation protocols
- Work on Subspace problem for disambiguation and Fine-type Tagging, by learning the relation betwween the subspaces in the type boxes to the use-cases mention before

11 Acknowledgements

We would like to thank our guide Prof. Soumen Chakrabarti for his constant support and thoughtful insights through out the project. We are indebted to him for the research discussions which have widened the hozizon of the field for us.

References

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

[2] Socher, Richard, Chen, Danqi, Manning, Christopher D, and Ng, Andrew. Reasoning with neural tensor networks for knowledge base completion. In NIPS, 2013.

[3] Tomas Mikolov, et al., Efficient Estimation of Word Representations in Vector Space, CoRR, 2013.

[4] Jeffrey Pennington, et al., GloVe: Global Vectors for Word Representation, Empirical Methods in Natural Language Processing (EMNLP), 2014.

[5] H. Poor, An Introduction to Signal Detection and Estimation. New York: Springer-Verlag, 1985, ch. 4.

[6] Levy, Omer, et al. "Do Supervised Distributional Methods Really Learn Lexical Inference Relations?." HLT-NAACL. 2015.

[7] Backstrom, Lars, and Jon Kleinberg. "Network bucket testing." Proceedings of the 20th international conference on World wide web. ACM, 2011.

[8] Levy, Omer, Yoav Goldberg, and Israel Ramat-Gan. "Linguistic Regularities in Sparse and Explicit Word Representations." CoNLL. 2014.