Learning Semantics for Linked Data:

How to Evaluate Embedding Models in a View of Semantic Web

Yi Luo

Fall 2014, Independent Study
Instructor: Professor Jeff Heflin

The Department of Computer Science and Engineering, Lehigh University

LEHIGH CSE

Abstract

Linked data, knowledge graph and other large-scale structured datasets are growing at an incredible rate. Exploring strategies for effectively and efficiently exploiting these datasets is a long standing goal of the Semantic Web. Among proposed solutions towards this goal, machine learning and statistical learning approaches are regarded as promising attempts. In this report, we review two types of statistical embedding models – Tensor Factorization and Translating Embeddings. We explore their pros and cons by detailed evaluation of these embedding models. We observe that the approach of categorizing relations of the knowledge graph from one general group into 1-TO-1, 1-TO-MANY, MANY-TO-1 and MANY-TO-MANY categories poses limitations for a learning model. This notion triggers us to think over the importance of *semantics*. We, therefore, decide to evaluate embedding models in a view of Semantic Web. Based on our proposed hypothesis, which is that "a reasonable semantic categorization of relations helps us better evaluate embedding models", we categorize relations following the OWL/OWL2 standards. We perform experiments for two embedding models - TransE and TransM – and compare their link prediction results on both a real life dataset FB15K and a synthetic dataset Pizza8. Our evaluation process reveal many interesting results, for example, TransM, which has been claimed to be an improved model of TransE in general link prediction, is actually outperformed by TransE with respect to the learning of real semantics. In the end, we discuss our results and propose some future work.

1. Introduction

In traditional web 2.0, unstructured data such daily news, social network feeds and user blogs always compose the majority part of internet information. These contents are friendly to human users but not understandable for computers. A decade ago, the concept "Semantic Web" was proposed by Tim-Berners Lee [1] to build a more intelligent version (web 3.0) where machines are able to understand semantics of information on the World Wide Web so that the internet can offer better services by itself. To achieve it, knowledge in unstructured or semi-structured styles will be converted into structured formats and the World Wide Web will finally become a "web of data" [2].

As the development of Semantic Web, structured contents such as RDF triples, linked data and collaborative knowledge bases, grow dramatically. In recent years, many structured data repositories have become very popular and widely used in both industries and academics, including long-term funded projects led by institutes such as WordNet[1], DBPedia[2] and YAGO[3] or collaborative knowledge bases composed by community members such as Freebase[4]. They are now important resources to support other AI related applications such as information extraction and question answering.

In 2012, Google released its Knowledge Graph[5] and extended its search engine results by adding structured and detailed information-box for the query topic to its traditional ranking list of links. The incorporation of Knowledge Graph has improved the search experience for the end-users. This effort can be seen as a practical adoption of Semantic Web in industries and motivates the advantage of leveraging structured knowledge bases. However, it is far from enough and these additional information-box features are only preliminary applications of Semantic Web. Exploring strategies to fully exploit structured resources is still an interesting and open research topic.

On one hand, traditional researchers in Semantic Web community try to design standards for representing knowledge and build efficient reasoners for intelligent inference. Researchers integrate ideas from description logics and database systems to build a symbolic framework for Semantic Web structured-data. Generally, there are two levels of representations: concept-level knowledge, such as classes and properties that compose the controlled vocabulary, known as T-

[1] http://wordnet.princeton.edu/
[2] http://dbpedia.org/
[3] http://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/
[4] https://www.freebase.com/
[5] https://www.google.com/insidesearch/features/search/knowledge.html

Box statements; instance-level knowledge, such as assertions and facts that are compliant with the vocabulary, known as A-Box statements. As a whole, the integration of T-Box and A-Box is known as ontology. Accordingly, ontology languages such as OWL/OWL2 [3] are proposed to offer formal semantics, sufficient expressive power and efficient reasoning support. The state-of-the-art DL reasoners (e.g. Pellet[1], FACT++[2], Racer[3]) are capable of performing sound and complete reasoning for ontologies of reasonable size and complexity. This track of research for exploiting structured knowledge base is known as ontology research. Compared to other ideas, it emphasizes more on semantics of data.

On the other hand, in the community of machine learning and data mining, researchers have a different view about how to use structured knowledge bases. Since most statements of existing knowledge bases are A-Box assertions, they are rich datasets for all kinds of statistical learning models. In a knowledge graph, nodes are entities and edges are predicates (aka relations) and they are expressed in the triple form *(subject, predicate, object)*. Such a graph model and triple representations are suitable for many learning algorithms. For example, with data mining techniques, association rule learning may be applied to learn frequent patterns and inference rules or abnormal detection algorithms may be applied to detect conflicting facts within the graph. Recently, inspired by the Word Vector [5, 6] idea in natural language processing, some researchers try to convert entities and predicates of the knowledge graph into vectors. By doing this, it is more convenient to integrate the graph with applications that involve numerical computing in continuous vector spaces. It also provides an approach to completing the graph by calculating the compatibility of a candidate triple (s, p, o). This track of research for exploiting structured knowledge bases is known as statistical learning or machine learning. As opposed to ontology researches and rule-based methods, these learning approaches care more about the size and type of datasets.

[1] http://clarkparsia.com/pellet/
[2] https://code.google.com/p/factplusplus/
[3] https://www.ifis.uni-luebeck.de/index.php?id=385

1.1. SEMANTIC Web or Semantic WEB?

Therefore, we may ask a question "SEMANTIC Web or Semantic WEB?" and discuss which aspect of the Semantic Web we should invest more energy. As summarized by James Hendler in his talk [23], there are both pros and cons for these two directions: "Ontology research is turning OWL into a usable KR standard but the linking story is unclear, linked-data-based applications are growing in size, number and importance on the web but the 'vocabulary' story is unclear."

To be more specific, rule-based strategies guarantee clear semantics and offer explicit evidence for reasoning. Reasoners can produce sound and complete inferred results. However, rule-based approaches highly rely on the quality of ontologies and are very sensitive to noises and inconsistencies in the data. On the contrary, statistics-based strategies are more flexible and typically robust to noise and data inconsistencies. They don't require explicit or detailed definitions of semantics; even a shallow ontology can be helpful. Currently, most web-scale structured knowledge bases such as Google's Freebase and CMU's NELL[1] still have A-Box facts as their majority statements and lack ontologies. Therefore, statistical methods can be a good start to explore these knowledge bases. However, the well-known drawback of statistical approaches is that the learning process is sort of a black box to human beings and many parameters are difficult even impossible to interpret.

In this report, we mainly research statistical learning approaches. In particular, we concentrate on embedding ideas to convert entities and predicates of the knowledge graph into continuous vector spaces. There are many embedding models; in this project we review two major branches: Tensor Factorization and Translating Embedding.

2. Embedding the Knowledge Graph

The embedding idea is inspired by the Word Vector [5, 6] work in natural language processing. With sufficient training documents, machine learning models convert a keyword into a k-

[1] http://rtw.ml.cmu.edu/rtw/

dimensional vector. This gives opportunities to execute lots of interesting tasks. For example, people can compute similarities of pairs of words to detect synonyms since synonyms always distribute close to each other in vector spaces. Also, words from different languages but conveying similar semantics may have similar geometric arrangements in each language space. In Figure 1, we can see that number words from English (left) and Spanish (right) have similar geometric arrangements in their vector spaces. This kind of results can be helpful for machine translation [4].

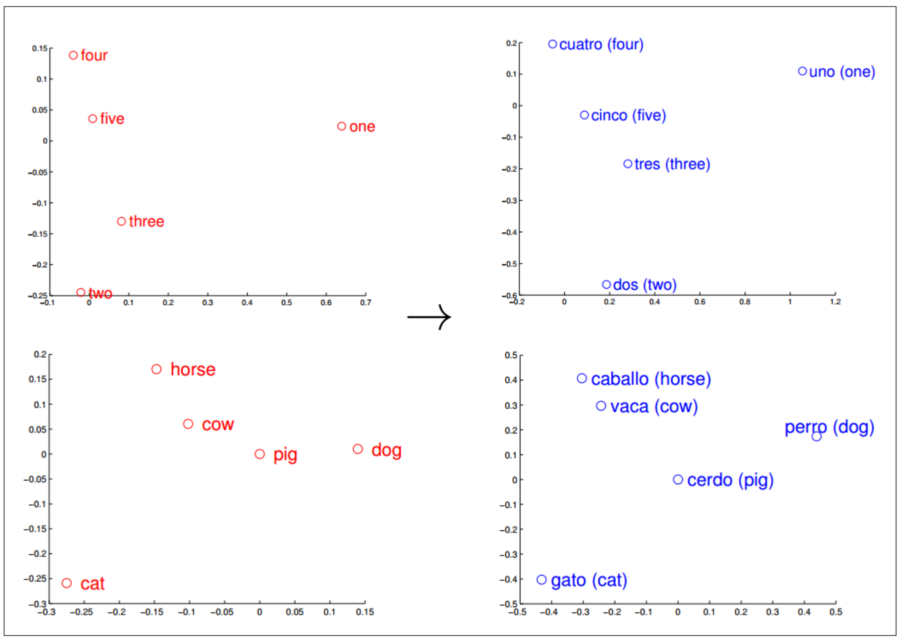

Figure 1. Distributed word vector representations of numbers and animals in English (left) and Spanish (right). The five vectors in each language were projected down to two dimensions using PCA, and then manually rotated to accentuate their similarity [4].

In addition, people can even directly apply linear operations upon vector representations. For instance, vector operations "king" – "man" + "woman" results in a vector that is close to "queen" [4]. This Word Vector work soon inspires several other embedding ideas such as Paragraph Vector [5] and the Embedding of the Knowledge Graph.

As we can see, embedding methods and results are not as sound and complete as rule-based methods. Instead of being competitors to rule-based approaches, embedding models act more as complimentary roles for effectively and efficiently exploiting structured knowledge graphs. In existing embedding algorithms, two of them are widely researched: Tensor Factorization [8, 9, 10, 11] and Translating Embedding [15, 16, 17, 18, 20, 21, 28, 29, 30].

2.1. Tensor Factorization

Tensor (three-dimensional) factorization is a derived work from matrix (two-dimensional) factorization [7] which has been practically adopted in today's recommender systems. Basically, matrix factorization is to find out two or more sub-matrices such that when multiplying them back you will get the original matrix. Factorization helps to disclose latent features underlying the interactions between different types of entities. In traditional recommender systems, there are two types of entities: users and items (movies or books).

	D1	D2	D3	D4
U1	5	3	-	1
U2	4	-	-	1
U3	1	1	-	5
U4	1	-	-	4
U5	-	1	5	4

Figure 2. A simple example of ratings by 5 users to 4 items [26]

Given a matrix like Figure 2, recommender systems need to figure out values of missing slots in the matrix so that to recommend interesting items to potential users. The matrix factorization method assumes there are latent features underlying the interactions between users and items. A user may be more interested in scientific and natural movies while another user may be more interested in romantic and love stories. Assuming the original matrix is $\boldsymbol{R}$ (with size $|M| \times |N|$) and we would like to find k features. Then our task is to discover two matrices $\boldsymbol{P}$ (with size $|M| \times k$) and $\boldsymbol{Q}$ (with size $|N| \times k$) such that:

$$\boldsymbol{R} \approx \boldsymbol{P} \times \boldsymbol{Q}^T$$

In this way, each row of $\boldsymbol{P}$ is a feature vector representing a user and each row of $\boldsymbol{R}$ is a feature vector representing an item. Therefore, to predict the rating of an item D_j by a user U_i we can calculate the dot product of two feature vectors as:

$$r_{ij} = p_i q_j^T$$

We now extend the same idea to three dimensions where there are three types of entities interacting with each other: a subject entity, a relation (predicate) and an object entity. Similar to matrix $\boldsymbol{R}$, we use a three-way tensor $\boldsymbol{X}$ with size $n \times n \times m$ to represent a knowledge graph, where n is the number of entities and m is the number of predicates.

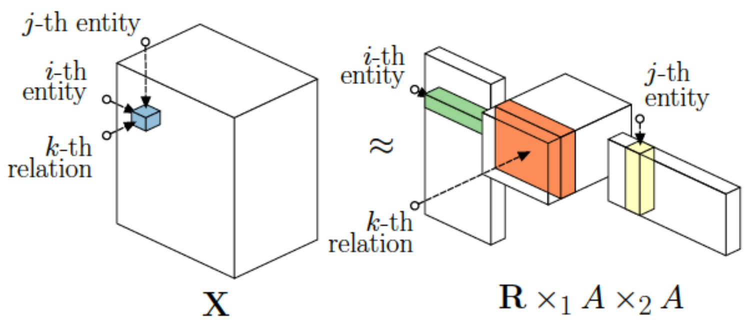

Figure 3. Factorization of a tenser $\boldsymbol{X}$ [9]

As shown in Figure 3, each element in the tensor χ_{ijk} represents a triple (i -th entity, k -th relation, j -th entity). If this triple appears in the knowledge graph, that element is set to 1; if not (not satisfied or unknown), the element is set to 0. Therefore, it is easy to build this three-way tensor. The next step is to factorize this tensor into sub-components: a factor matrix $A \in \mathbb{R}^{n \times r}$ and a core tensor $R \in \mathbb{R}^{r \times r \times n}$ such that:

$$X \approx R \times_1 A \times_2 A \,,$$

$$\text{i.e. } X_k \approx A R_k A^T, for\ k = 1, \dots, m$$

In this way, the factor matrix A turns to be the embeddings of all entities and the k -th relation (predicate) is represented as R_k. According to this tensor factorization approach, M. Nickel et al. [9] implemented a system named RESCAL and carried experiments [10] on the YAGO 2 core ontology which consists of millions of entities and facts at that time. Their system was able to predict unknown triples on a large-scale knowledge graph.

2.2. Translating Embedding

One deficiency of tensor factorization is that the original three-way $\boldsymbol{X}$ could be very huge and sparse, then the factorization would be extremely complicated and time-consuming because it involves a large number of parameters and many iterations of learning processes.

An alternative embedding method, translating embedding, is proposed as a promising one. It has a smaller number of parameters to learn so it is more efficient compared to tensor factorization. Moreover, it can still achieve state-of-the-art predictive performance. The simplest translating embedding model proposed by A. Bordes et al. [16] was named TransE. This method models relations (predicates) as translations operating on the embeddings of entities. If a triple (h, l, t) holds, the embedding of the tail entity t should be close to the embedding of the head entity h plus the vector representation of predicate l in the same vector space.

TransE defines a dissimilarity measure d as $d(h + l, t)$, which is always *L1-norm* (Least Absolute Deviations) or *L2-norm* (Least Squares). And the learning objective loss function is defined as

$$\mathcal{L} = \sum_{(h,l,t)\in S} \sum_{(h',l,t')\in S'_{(h,l,t)}} [\gamma + d(h + l, t) - d(h' + l, t')]_+$$

where $[x]_+$ denotes the positive part of x, $\gamma > 0$ is a margin hyperparameter, (h, l, t) refers to correct (positive) triples from the training set and (h', l, t') refers to corresponding corrupted (negative) triples. The learning algorithm favors to decrease the value of dissimilarity measure d

for correct training triples while increase the value of corrupted triples, which is intuitive and desired. The optimization process is carried out by mini-batch stochastic gradient descent (SGD).

Table 1. Number of parameters of different models. n_e and n_r are the number of unique entities and relations respectively. It is the often case that $n_r \ll n_e$. k is the dimension of embedding space. s is the number of hidden nodes of a neural network or the number of slices of a tensor.

Model	Number of Parameters
Unstructured [25]	$O(n_e k)$
SE [15]	$O(n_e k + 2n_r k^2)$
SME (Linear) [25]	$O(n_e k + n_r k + 4k^2)$
SME (Bilinear) [25]	$O(n_e k + n_r k + 2k^3)$
NTN [13]	$O(n_e k + n_r(sk^2 + 2sk + 2s))$
RESCAL [9]	$O(n_e k + n_r k^2)$
TransE [16]	$O(n_e k + n_r k)$

As shown in Table 1, the most important advantage of the translating model (TransE) is that it has a small number of parameters compared to other embedding algorithms, making it a scalable system for large size datasets.

3. Evaluation of Embedding Models

Both tensor factorization and translating embedding models can be trained in the preprocessing stage in only one run and the resulting embedding vectors can be permanently stored on disks for repeated use. So the learning time is not an essential factor as long as it is acceptable. Instead, we emphasize more on the quality of the embeddings vectors because we will use them again and again for different applications.

To evaluate the quality of embedding results, A. Bordes et al. [15, 16, 25] designed experiments on two tasks: link prediction and triple classification, from different viewpoints and application context.

3.1. Link Prediction

This task is to predict the subject entity given a triple $(?, p, o)$ or the object entity given $(s, p, ?)$. Basically, it is impossible to predict an entity outside of the knowledge base. Rather than a single best answer, this task emphasizes more in the ranking of candidate entities from the knowledge graph. A potential application is to answer questions like (?, presidentOf, USA) and (WALL-E, genre, ?) for question answering systems.

A. Bordes et al. [16] described the experiment protocol. For each testing triple (s, p, o), the object entity is replaced by every entity e in the knowledge graph vocabulary to get a corresponding corrupted triple (s, p, e). Then dissimilarity scores of all triples, including the testing triple and all corrupted ones, are calculated. Ranking these scores in ascending order, we get the rank of the testing triple. Similarly, we can get another rank for the same testing triple by corrupting the subject entity. Eventually, we need to report two metrics: the averaged rank (denoted as Mean) and the proportion of ranks not larger than 10 (denoted as Hits@10). This is regarded as the "raw" setting. However, if a corrupted triple exists in the knowledge graph and is correct, we should not count it as wrong if it ranks before the testing triple. Therefore we need to remove them from the ranking list. This setting is denoted as "filter". Obviously, we expect a lower Mean while a higher Hits@10 as good performance.

3.2. Triple Classification

This task is to run a binary classification on a given triple (s, p, o) to check whether the triple is correct or not. The decision rule is simple: for a triple (s, p, o), if the dissimilarity score is lower than a threshold σ, then predict positive (correct). Otherwise, predict negative (wrong).

3.3. A Statistical View of Improvement

According to [16, 20, 21], the performance of link prediction acts as the major sign to indicate quality of embedding vectors so in this report we focus on link prediction. Researchers try to improve embedding models by decreasing the Mean rank and increasing Hits@10.

For example, TransM [20] is claimed to be an improved version of TransE and it has better performance in link prediction on both Mean rank and Hits@10. As shown in Figure 4, TransM and TransE are compared on two datasets WN18 and FB15K.

DATASET	WN18				FB15K			
METRIC	*MEAN RANK*		*MEAN HIT@10*		*MEAN RANK*		*MEAN HIT@10*	
	Raw	*Filter*	*Raw*	*Filter*	*Raw*	*Filter*	*Raw*	*Filter*
Unstructured	315	304	35.3%	38.2%	1,074	979	4.5%	6.3%
RESCAL	1,180	1,163	37.2%	52.8%	828	683	28.4%	44.1%
SE	1,011	985	68.5%	80.5%	273	162	28.8%	39.8%
SME(LINEAR)	545	533	65.1%	74.1%	274	154	30.7%	40.8%
SME(BILINEAR)	526	509	54.7%	61.3%	284	158	31.3%	41.3%
LFM	469	456	71.4%	81.6%	283	164	26.0%	33.1%
TransE	294.4	283.2	70.4%	80.2%	243.3	139.9	36.7%	44.3%
TransM	**292.5**	**280.8**	**75.7%**	**85.4%**	**196.8**	**93.8**	**44.6%**	**55.2%**

Figure 4. Link prediction results and the comparison between TransM and other baseline models, including TransE.

Unstructured [25], RESCAL [9], SE [15], SME (Linear) [25], SME (Bilinear) [25] and LFM [27] are baseline models which are not discussed in this report. The dataset WN18 is a subset of WordNet and FB15K is a subset of Freebase. Their detailed statistics are shown in Figure 5 where the number of entities, relations, training examples, validating examples and testing examples are compared.

DATASET	WN18	FB15K
#(ENTITIES)	40,943	14,951
#(RELATIONS)	18	1,345
#(TRAINING EX.)	141,442	483,142
#(VALIDATING EX.)	5,000	50,000
#(TESTING EX.)	5,000	59,071

Figure 5. Statistics of datasets WN18 and FB15K.

Similarly, TransH [21] is asserted to be an even better model to TransE and its link prediction results are shown in Figure 6.

Dataset	WN18				FB15k			
Metric	MEAN		HITS@10		MEAN		HITS@10	
	Raw	Filt.	Raw	Filt.	Raw	Filt.	Raw	Filt.
Unstructured (Bordes et al. 2012)	315	304	35.3	38.2	1,074	979	4.5	6.3
RESCAL (Nickel, Tresp, and Kriegel 2011)	1,180	1,163	37.2	52.8	828	683	28.4	44.1
SE (Bordes et al. 2011)	1,011	985	68.5	80.5	273	162	28.8	39.8
SME (Linear) (Bordes et al. 2012)	545	533	65.1	74.1	274	154	30.7	40.8
SME (Bilinear) (Bordes et al. 2012)	526	509	54.7	61.3	284	158	31.3	41.3
LFM (Jenatton et al. 2012)	469	456	71.4	81.6	283	164	26.0	33.1
TransE (Bordes et al. 2013b)	263	**251**	75.4	**89.2**	243	125	34.9	47.1
TransH (unif.)	318	**303**	75.4	**86.7**	211	**84**	42.5	**58.5**
TransH (bern.)	400.8	388	73.0	82.3	212	**87**	45.7	**64.4**

Figure 6. Link prediction results and the comparison between TransH and other baseline models, including TransE.

Apparently, this evaluation method of link prediction is from a very general statistical view, all predicates (relations) are discussed together without any difference.

However, A. Bordes et al. [16] provides a more interesting and insightful review. It groups relations according to cardinalities of their subject and object entities into four categories: 1-TO-1, 1-TO-MANY, MANY-TO-1 and MANY-TO-MANY. Basically, for each predicate r, the average number of object entities per subject entity (ops_r) and the average number of subject entities per object entity (spo_r) are calculated. If $ops_r < 1.5$ and $spo_r < 1.5$, r is regarded as 1-TO-1. If $ops_r \geq 1.5$ and $spo_r < 1.5$, r is regarded a 1-TO-MANY. If $ops_r < 1.5$ and $spo_r \geq 1.5$, r is regarded as MANY-TO-1. If $ops_r \geq 1.5$ and $spo_r \geq 1.5$, r is regarded as MANY-TO-MANY. Figure 7 shows link prediction results (Hits@10) by relation category of TransE, TransM and TransH on the same dataset FB15K pulled out from their original papers.

Compared to other baseline models, TransE has dramatic improvement on 1-TO-1 relations but does not improve that much on 1-TO-MANY, MANY-TO-1 and MANY-TO-MANY relations. As for TransM and TransH, both of them report significant improvement than TransE, especially on 1-TO-MANY, MANY-TO-1 and MANY-TO-MANY relations. In fact, A. Bordes et al. [16] pointed out the weakness of TransE when dealing with MANY-associated categories of relations. TransM and TransH were indeed proposed to tackle this disadvantage.

TASK	Predicting head				Predicting tail			
REL. Mapping	1-TO-1	1-TO-M.	M.-TO-1	M.-TO-M.	1-TO-1	1-TO-M.	M.-TO-1	M.-TO-M.
Unstructured	34.5%	2.5%	6.1%	6.6%	34.3%	4.2%	1.9%	6.6%
SE	35.6%	62.6%	17.2%	37.5%	34.9%	14.6%	68.3%	41.3%
SME (LINEAR)	35.1%	53.7%	19.0%	40.3%	32.7%	14.9%	61.6%	43.3%
SME (BILINEAR)	30.9%	69.6%	19.9%	38.6%	28.2%	13.1%	76.0%	41.8%
TransE	59.7%	77.0%	14.7%	41.1%	58.5%	18.3%	80.2%	44.7%
TransM	**76.8%**	**86.3%**	**23.1%**	**52.3%**	**76.3%**	**29.0%**	**85.9%**	**56.7%**

(a) TransM vs. TransE (and other models)

Task	Predicting left (HITS@10)				Predicting right (HITS@10)			
Relation Category	1-to-1	1-to-n	n-to-1	n-to-n	1-to-1	1-to-n	n-to-1	n-to-n
Unstructured (Bordes et al. 2012)	34.5	2.5	6.1	6.6	34.3	4.2	1.9	6.6
SE (Bordes et al. 2011)	35.6	62.6	17.2	37.5	34.9	14.6	68.3	41.3
SME (Linear) (Bordes et al. 2012)	35.1	53.7	19.0	40.3	32.7	14.9	61.6	43.3
SME (Bilinear) (Bordes et al. 2012)	30.9	69.6	19.9	38.6	28.2	13.1	76.0	41.8
TransE (Bordes et al. 2013b)	43.7	65.7	18.2	47.2	43.7	19.7	66.7	50.0
TransH (unif.)	**66.7**	**81.7**	**30.2**	**57.4**	**63.7**	**30.1**	**83.2**	**60.8**
TransH (bern.)	**66.8**	**87.6**	**28.7**	**64.5**	**65.5**	**39.8**	**83.3**	**67.2**

(b) TransH vs. TransE (and other models)

Figure 7. Link prediction results by relation category of TransE, TransM, TransH and other baseline models on FB15K.

M. Fan et al. [20] introduces TransM. The basic idea is to assign different weights to different relations according to their categories so that during the training process: 1-TO-1 relationships will have greater weights, 1-TO-MANY/MANY-TO-1 relationships have less weights and MANY-TO-MANY relationships have least weights. Then relations associated with MANY will be relaxed during the learning process. As a result, head or tail entities that appear at the MANY side will not be forced to be the same. The model adapts the score function as follows.

$$f_r(h,t) = w_r \|h + r - t\|_{L_1/L_2},$$

where $w_r = \frac{1}{log\,(h_r pt_r + t_r ph_r)}$ and$h_r pt_r$ means "heads per tail" and $t_r ph_r$ means "tails per head" (head refers to subject entity and tail refers to object entity).

Z. Wang et al. [21] introduces TransH which models a relation as a hyperplane with a translation operation on it.

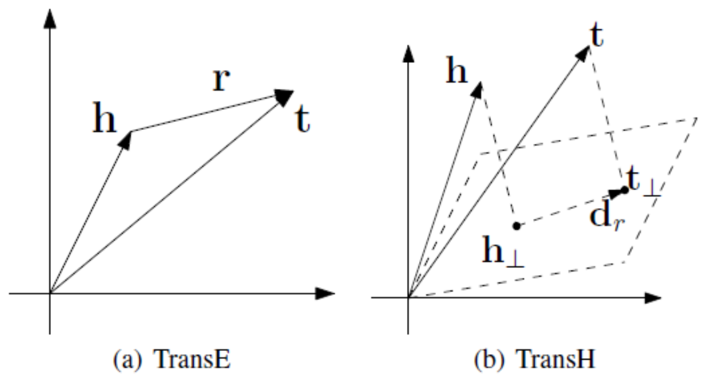

Figure 8. Simple illustration of TransE and TransH.

As shown in Figure 8, the basic idea is that an entity can have different representations when involved in different relations. It enables different roles of an entity in different relations/triples by projecting them to relation-specific hyperplanes. One advantage is that this adaption does not introduce too many extra parameters but still well handles the trade-off between expressivity and efficiency.

Therefore, it is the performance deficiency of TransE on MANY-associated categories of relations that drives the invention of TransM and TransH. With merely general link prediction results like Figure 4 and Figure 6, it is very possible to ignore the clues. The way to categorize relations into different types indeed helps discover meaningful distinctions between relations. In a view of Semantic Web, different category of relations actually conveys different "semantics". By considering different semantics of different relation categories, we are able to design better models. This idea encourages us to look deeper into the diversity of relations, in a more advanced view of Semantic Web.

3.4. A View of Semantic Web

Since TransM and TransH have better results on 1-TO-MANY, MANY-TO-1 and MANY-TO-MANY relations, it means that they have learnt better semantics for these categories of relations of the knowledge graph. Therefore, we come up with a hypothesis: *A reasonable semantic* *categorization of relations helps us better evaluate embedding models*. We can verify if a learning model really learns semantics of predicates of a knowledge graph or not. We can also have a better understanding about which type of relations need to be enhanced to really improve the future model.

In the Semantic Web, there have been extensive researches on characteristics of properties and predicates. In OWL/OWL2, a predicate could be: functional, inverse functional, transitive, symmetric, asymmetric, reflexive and irreflexive [3, 22]. We therefore propose our categorization of relations by following these 7 characteristics.

4. Experiments

We categorized relations into functional, inverse functional, transitive, symmetric, asymmetric, reflexive and irreflexive ones. We attempted to get link prediction results of all learning models by our relation category. But we were just able to obtain source codes of TransE and TransM so we only compare these two models in this report. As the beginning step, we carried out experiments on FB15K dataset.

4.1. Real Word Dataset: FB15K

However, as FB15K was from Freebase where ontology information is extremely lack, we didn't have a clear category list for each relation in the dataset. Therefore, we decided to assign characteristics for relations by ourselves. Rather than handling all 1,345 relations of FB15K, we selected the top 100 relations according to their frequencies of appearance in the data.

But we soon noticed many relations were in the following form where two components were connected by a dot.

"/soccer/football_team/current_roster./soccer/football_roster_position/position"

By checking the definition and structure of original Freebase, we figured out that this kind of combination relations in the form “p1.p2” were in fact derived (synthetic) representations that did not appear in the official Freebase RDF dump. There are some problems to adopt these representations. First, there could be information loss. Let's take the Kansas City Chiefs team (/m/0487_) as an example; the official RDF dump has related triples:

/m/0487_ (Kansas City Chiefs)	/roster	/m/0_r05xj (anonymous node)
/m/04tm16 (Bill Maas)	/teams	/m/0_r05xj (anonymous node)
/m/0q5dv (Defensive tackle)	/players	/m/0_r05xj (anonymous node)

The anonymous node (/m/0_r05xj) represents a row of a table, which then has attributes like “player”, “team”, “from (year)”, “to (year)” and “position”. Thus, linking a team entity, say Manchester United, directly with a position entity, say Forward via a combination relation can lose information because we are not able know the associated player. Also, statements like “*Manchester United* has a position *Forward*” are not very interesting or useful facts.

So we got another 100 relations by removing them from the list, which we called “nodot” relations. These “nodot” relations had no information loss and were more meaningful than combination relations. We went through these 100 relations and assigned characteristics manually; the final assignment is available for download from the repository of this report[1]. There are only 39 from 100 relations can be easily recognized as having special characteristics: 21 functional, 14 inverse functional and 4 transitive relations.

We summarized link perdition results (Hits@10) of TransE and TransM on FB15K for functional, inverse functional and transitive relations as shown in Table 2 and Figure 9. TransM still performed better than TransE on these three categories but not as significant as in the general statistics. It indicated that TransM did not learn much better semantics of these particular three types of predicates.

[1] https://github.com/iluoyi/knowledge-graph-embedding

Table 2. Link prediction results (Hits@10) of TransE and TransM by relation category on FB15K

Category	Functional		Inverse functional		Transitive	
	raw	filt.	raw	filt.	raw	filt.
TransE	44.15%	45.96%	47.25%	49.50%	47.44%	52.55%
TransM	**47.45%**	**49.78%**	**47.91%**	**51.27%**	**48.27%**	**53.05%**

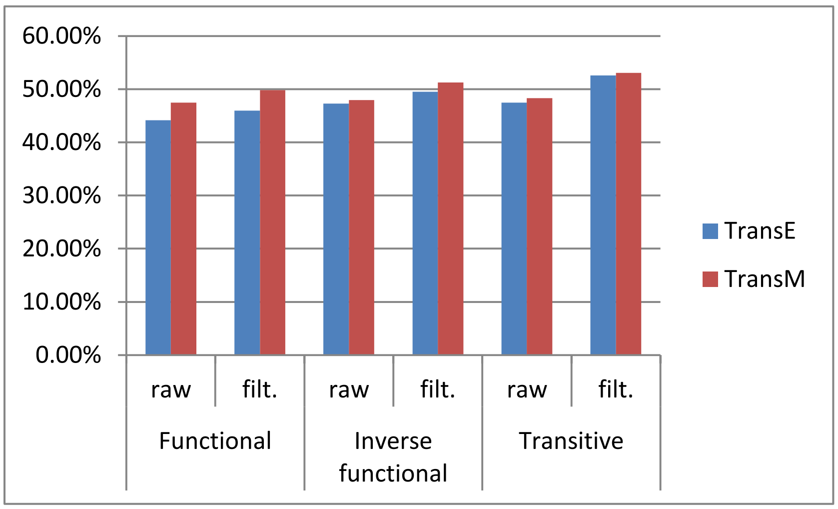

Figure 9. Link prediction results (Hits@10) of TransE and TransM on FB15K. Results of two models are comparable.

We considered finding a dataset with richer semantics (i.e. richer T-Box statements) which would have more types of relations. But after exploring many existing structure knowledge bases, we found most of them consisted of very shallow ontologies and huge amounts of A-Box level statements. It was difficult to find a qualified dataset. Thus, we came up with an idea to create a synthetic dataset.

4.2. Synthetic Dataset: Pizza8

It is reasonable to generate synthetic dataset for structured knowledge bases and corresponding experiments. In natural language processing and text mining, input data is usually in an unstructured style, making it difficult to generate synthetic data without any bias. But for knowledge graphs and structured knowledge bases, life is easier because the number of parameters is under control. There has been some outstanding work with similar ideas to generate synthetic data for the testing of structured systems, such as the Lehigh University Benchmark (LUBM) [19] which was proposed for the evaluation of Semantic Web repositories in a standard and systematic way.

An ideal step to make synthetic data for our experiment is to find an ontology that covered all interesting types of relations, and then generate a controlled number of triples by creating instances and linking them together.

The problem to generate appropriate synthetic data for a given ontology itself could be an interesting research work. In our experiment, we applied some heuristics and simplified the process. We browsed several online libraries and selected pizza.owl[1] as our experimental ontology. It had a small number of classes and only 8 object properties as shown in Table 3.

Table 3. 8 properties of pizza.owl and their characteristics (Y means yes, N means no).

Properties	Domain	Range	Functional	Inverse functional	Transitive
isBaseOf	PizzaBase	Pizza	Y	Y	N
hasBase	Pizza	PizzaBase	Y	Y	N
isToppingOf	PizzaTopping	Pizza	Y	N	N
hasTopping	Pizza	PizzaTopping	N	Y	N
hasIngredient	Food	Food	N	N	Y
isIngredientOf	Food	Food	N	N	Y
hasCountryOrigin	Pizza	Country	N	N	N
hasSpiciness	PizzaTopping	Spiciness	N	N	N

We created instances for leaf node classes of Pizza ontology and linked instances to generate real triples through the following heuristic rules:

1. Each pizza has 0 or 1 origin country;
2. Each pizza has 1 and only 1 pizza base;
3. Each pizza has 0 to 5 pizza toppings;
4. Each pizza topping has 0 or 1 spicy degree.

[1] http://130.88.198.11/co-ode-files/ontologies/pizza.owl

Finally, we generated 910,800 entities and 1,482,727 triples and named the synthetic dataset as Pizza8 (i.e. 8 relations). We split these triples into Train (1,384,198 triples), Valid (19,705 triples) and Test (78,824 triples) datasets in the same way as preparing FB15K in [15]. Detailed statistics of Pizza8 versus FB15K are shown in Figure 10, obviously, Pizza8 has a significantly larger scale.

Dataset	FB15K	Pizza8
#(Entities)	14,951	910,800
#(Relations)	1,345	8
#(Training Ex.)	483,142	1,384,198
#(Validating Ex.)	50,000	19,705
#(Testing Ex.)	59,071	78,824

Figure 10. Statistics of datasets Pizza8 and FB15K.

Since the test program replaced subject or object entity with every other entity from the vocabulary to make negative triples for each testing triple, it could take a long time for the evaluation process. On frodo[1], it took 2 days to finish the test of TransE. We thus considered parallel computing to speed up the process. The test program traversed all testing triples and calculated their ranks independently, so this process could be paralleled in a fork/join style.

We deployed test programs on 8 Sunlab machines[2] and executed testing for TransE and TransM respectively. The average running time was reduced from more than 50 hours to 7 hours. At last, we got link prediction results of two models as shown in Table 4 and Figure 11.

Table 4. Link prediction results (Hits@10) of TransE and TransM by relation category on Pizza8

Category	**Functional**		**Inverse functional**		**Transitive**	
	raw	filt.	raw	filt.	raw	filt.
TransE	**92.86%**	**93.40%**	**93.88%**	**94.36%**	**88.77%**	**90.39%**
TransM	73.86%	74.55%	71.94%	72.56%	64.34%	65.75%

[1] frodo.cse.lehigh.edu
[2] ariel, caliban, dactyl, enceladus, iapetus, mars, jupiter, saturn

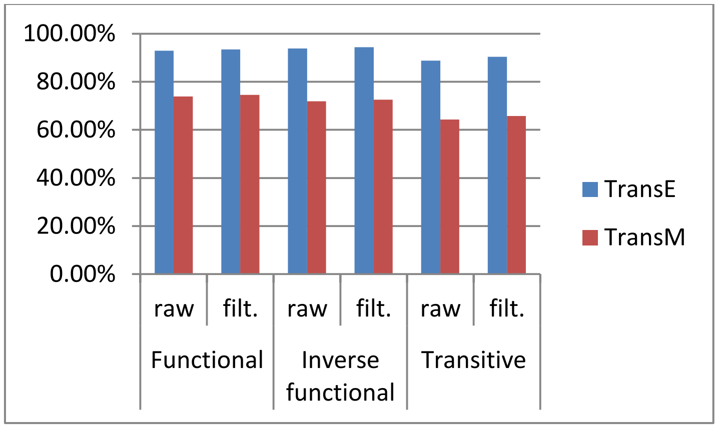

Figure 11. Link prediction results (Hits@10) of TransE and TransM on Pizza8. TransE performs better than TransM.

On Pizza8 dataset, one interesting phenomenon was TransE had better link prediction results than TransM, even though TransM was claimed to be an improved model.

4.3. Discussions of Experiment Results

By comparing Table 4 and Table 2 we get the diagram in Figure 12, we can see that both models have better performance on dataset Pizza8. For example, the Hits@10 (filt.) value of transitive relations has been increased from 52.55% to 90.39% for TransE and from 53.05% to 65.75% for TransM. It is because of the significant increase of training data size from 0.48million to 1.38 million triples. With more training data, parameters of the learning model could be updated with more samples so as to reach a more optimal configuration.

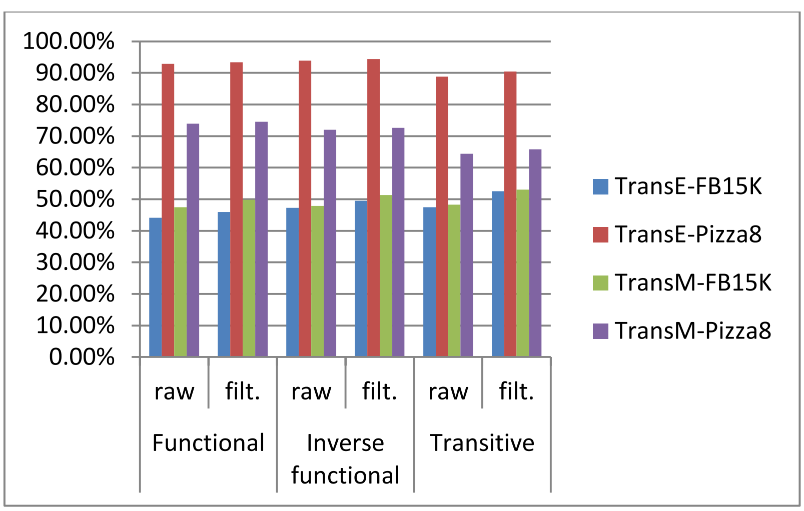

Figure 12. Both TransE and TransM perform better on Pizza8 than on FB15K.

However, the augment of Hits@10 values of functional, inverse functional and transitive relations of TransM is not as dramatic as that of TransE. Consequently, TransE has better results than TransM on Pizza8. TransM is claimed to perform better because it handles 1-TO-MANY, MANY-TO-1 and MANY-TO-MANY relations more gracefully than TransE. Hence for Pizza8, we also summarize link prediction results as Table 5 by following the same categorization method as Figure 7. Unfortunately, we don't have MANY-TO-MANY relations in Pizza8.

Table 5. Link prediction results (Hits@10) of TransE and TransM by 4 relation categories on Pizza8

Category	**1-TO-1**		**1-TO-MANY**		**MANY-TO-1**	
	raw	filt.	raw	filt.	raw	filt.
TransE	**93.94%**	**93.94%**	**89.34%**	**90.81%**	**60.90%**	**61.94%**
TransM	71.84%	71.84%	67.05%	68.54%	46.50%	47.53%

From these results, we think the Pizza8 dataset may be biased since it has no MANY-TO-MANY relations so that TransM cannot reveal its advantages which deal with MANY-associated relations better than TransE. But it still gives us clear information that TransM does NOT outperform TransE on the learning of real semantics. It only wins TransE in a general statistical view on a dataset with much more MANY-associated relations. In fact, FB15K has 26.2% 1-TO-1 relations and 73.8% MANY-associated relations (22.7% of 1-TO-MANY, 28.3% of MANY-TO-1, and 22.8% of MANY-TO-MANY) [16]. As a future work, we need to compare these models on another synthetic dataset which has more categories of relations.

5. Conclusions and Future Work

In this report, we introduced two branches of researches about how to exploit structured knowledge graphs and moved our focus onto statistical learning methods rather than rule based methods. We reviewed two major embedding models which were Tensor Factorization and Translating Embedding and introduced their designs and properties. We discussed their evaluation strategies that revealed the pros and cons of these models. We observed that the approach of categorizing relations of the knowledge graph from one general group into 1-TO-1, 1-TO-MANY, MANY-TO-1 and MANY-TO-MANY categories poses limitations for a learning model. This notion triggered us to think over the importance of *semantics* since different categories of relations actually convey different semantics. We therefore decided to evaluate embedding models in a view of the Semantic Web.

Our hypothesis was that "a reasonable semantic categorization of relations helps us better evaluate embedding models". In our evaluation design, we used 7 characteristics: functional, inverse functional, transitive, symmetric, asymmetric, reflexive and irreflexive for relation categorization. We carried out experiments on TransE and TransM with our evaluation method on both a real life dataset FB15K and a synthetic dataset Pizza8. We got many interesting results as discussed in section 4.3. However, to verify our hypothesis, we have some future work to do:

1. We need to execute experiments for more models such as RESCAL, TransH and other embedding models. We will look at their link prediction results in a general view and summarize those results by different relation categories. Then we are able to know which model has a better coverage on different semantics and which model only emphasizes on one or more categories of relations.
2. We will have a deeper research on how to generate synthetic knowledge graph to best simulate real life datasets. A reasonable starting point is to have a detailed list of parameters we want to control along with their definitions. Then we need to review sufficient practical structured knowledge bases and obtain statistics about how they compose in order to guide the construction of our synthetic data.
3. By analyzing prediction results on different categories of relations for different models, we are able to know their learning drawbacks and how they behave on different semantics. Therefore, we can come up with corresponding enhancement for those models.

As a future work, we can adapt a model and improve its performance under the guidance of our evaluation strategy.

References

[1] T. Berners-Lee, J. Hendler and O. Lassila. The Semantic Web. *Scientific American*, May 2001, p. 29-37.

[2] Semantic Web. Wikipedia, http://en.wikipedia.org/wiki/Semantic_Web.

[3] OWL 2 Web Ontology Language Document Overview (Second Edition). December 11 2012. http://www.w3.org/TR/owl2-overview/.

[4] T. Mikolov, Q.V. Le and I. Sutskever. Exploiting Similarities among Languages for Machine Translation. arXiv 2013.

[5] T. Mikolov, I. Sutskever, K. Chen, G. Corrado and J. Dean. Distributed Representations of Words and Phrases and their Compositionality. ICML 2014.

[6] T. Mikolov, K. Chen, G. Corrado and J. Dean. Efficient estimation of word representations in vector space. ICLR Workshop 2013.

[7] Y. Koren, R. Bell and C. Volinsky. Matrix Factorization Techniques for Recommender Systems. IEEE Computer Society, August 2009.

[8] M. Nickel and V. Tresp. Tensor Factorization for Multi-Relational Learning. ECMLPKDD 2013.

[9] M. Nickel, V. Tresp and H. Kriegel. A Three-Way Model for Collective Learning on Multi-Relational Data. ICML 2011.

[10] M. Nickel, V. Tresp and H. Kriegel. Factorizing YAGO. WWW 2012.

[11] M. Nickel and V. Tresp. Machine Learning on Linked Data: Tensors and their Applications in Graph-Structured Domains. ISWC 2012 Tutorial.

[12] V. Tresp, M. Bundschus, A. Rettinger and Y. Huang. Towards machine learning on the semantic web. Uncertainty reasoning for the Semantic Web I. 2008.

[13] R. Socher, D. Chen, C. Manning and A. Ng. Reasoning With Neural Tensor Networks for Knowledge Base Completion. NIPS 2013.

[14] T. Franz, A. Schultz, S. Sizov and S. Staab. TripleRank: Ranking Semantic Web Data By Tensor Decomposition. ISWC 2009.

[15] A. Bordes, J. Weston, R. Collobert and Y. Bengio. Learning Structured Embeddings of Knowledge Bases. AAAI 2011.

[16] A. Bordes, N. Usunier and A. Garcia-Duran. Translating Embeddings for Modeling Multi-relational Data, NIPS 2013.

[17] A. Bordes, J. Weston and N. Usunier. Open Question Answering with Weakly Supervised Embedding Models, ECML 2014.

[18] A. Bordes, S. Chopra and J. Weston. Question Answering with Subgraph Embeddings. EMNLP 2014.

[19] Y. Guo, Z. Pan and J. Heflin. LUBM: A Benchmark for OWL Knowledge Base Systems. Web Semantics. July 2005. pp.158-182.

[20] M. Fan, Q. Zhou, E. Chang and T. Zheng. Transition-based Knowledge Graph Embedding with Relational Mapping Properties. PACLIC 2014.

[21] Z. Wang, J. Zhang, J. Feng, Z. Chen. Knowledge Graph Embedding by Translating on Hyperplanes. AAAI 2014.

[22] G. Antoniou, P. Groth, F. Harmelen and R. Hoekstra. A Semantic Web Primer. Third Edition 2012.

[23] J. Hendler. Semantic Web: The Inside Story. July 2014. http://www.slideshare.net/jahendler/semantic-web-the-inside-story

[24] T. Rocktaschel, M. Bosnjak, S. Singh and S. Riedel. Low-Dimensional Embeddings of Logic. ACL Workshop on Semantic Parsing 2014.

[25] A. Bordes, X. Glorot, J. Weston and Y. Bengio. A semantic matching energy function for learning with multi-relational data. *Machine Learning* 2013.

[26] A. A. Yeung, Matrix Factorization: A Simple Tutorial and Implementation in Python. September 2010.

[27] R. Jenatton, N. Le Roux, A. Bordes, G. Obozinski, et al. A latent factor model for highly multi-relational data. NIPS 2012.

[28] Fan, Miao, Qiang Zhou, Thomas Fang Zheng, and Ralph Grishman. "Probabilistic Belief Embedding for Knowledge Base Completion." arXiv preprint arXiv:1505.02433 (2015).

[29] Fan, Miao, Qiang Zhou, and Thomas Fang Zheng. "Learning Embedding Representations for Knowledge Inference on Imperfect and Incomplete Repositories." arXiv preprint arXiv:1503.08155 (2015).

[30] Fan, Miao, Qiang Zhou, Thomas Fang Zheng, and Ralph Grishman. "Large Margin Nearest Neighbor Embedding for Knowledge Representation." arXiv preprint arXiv:1504.01684 (2015).