An Evaluation of Knowledge Graph Embeddings for Autonomous Driving Data: Experience and Practice

Ruwan Wickramarachchi[1], Cory Henson[2], and Amit Sheth[1]
[1]Artificial Intelligence Institute, University of South Carolina, Columbia, SC, USA
ruwan@email.sc.edu, amit@sc.edu
[2]Bosch Research and Technology Center, Pittsburgh, PA, USA
cory.henson@us.bosch.com

Abstract

The autonomous driving (AD) industry is exploring the use of knowledge graphs (KGs) to manage the vast amount of heterogeneous data generated from vehicular sensors. The various types of equipped sensors include video, LIDAR and RADAR. Scene understanding is an important topic in AD which requires consideration of various aspects of a scene, such as detected objects, events, time and location. Recent work on knowledge graph embeddings (KGEs) - an approach that facilitates neuro-symbolic fusion - has shown to improve the predictive performance of machine learning models. With the expectation that neuro-symbolic fusion through KGEs will improve scene understanding, this research explores the generation and evaluation of KGEs for autonomous driving data. We also present an investigation of the relationship between the level of informational detail in a KG and the quality of its derivative embeddings. By systematically evaluating KGEs along four dimensions – i.e. quality metrics, KG informational detail, algorithms, and datasets – we show that (1) higher levels of informational detail in KGs lead to higher quality embeddings, (2) type and relation semantics are better captured by the semantic transitional distance-based TransE algorithm, and (3) some metrics, such as coherence measure, may not be suitable for intrinsically evaluating KGEs in this domain. Additionally, we also present an (early) investigation of the usefulness of KGEs for two use-cases in the AD domain.

1 Introduction

Forecasters predict that fully autonomous vehicles could be commercially available in the next few years; and within the next few decades (i.e. 2040) half of all vehicles sold and 40 percent of vehicle travel could be autonomous (Litman 2019). While racing to realize this vision, the automotive industry is investing heavily into machine learning and other relevant AI technologies. To meet the increasing data demands of ML algorithms, fleets of vehicles are now deployed in multiple cities around the world and collecting massive amounts of data. These vehicles are equipped with various types of heterogeneous sensors, including – but not limited to – video, LIDAR, and RADAR.

To manage these vast amounts of automotive sensor data, companies are beginning to experiment with the use of KGs. In other industries, and for many years, KGs have proven invaluable for helping to manage data stored within enterprise data lakes, which are growing rapidly in popularity. More specifically, KGs help to enable the principles of FAIR data – i.e. findability, accessibility, interoperability, and re-use – across an enterprise.

Current research into the topic of neuro-symbolic fusion[1] (Nickel et al. 2015; Sheth et al. 2019), however, is beginning to point to new and exciting uses of KGs. Essentially, KGs are a key source of high-quality domain knowledge and KGE technology is now enabling ML algorithms to more directly access this knowledge. Recent studies have already shown that the use of KGEs leads to improved performance and predictive capabilities (Chen et al. 2017; Wang et al. 2019; Myklebust et al. 2019). For this reason, we believe that neuro-symbolic fusion through KGEs may provide valuable knowledge needed to improve scene understanding for autonomous driving.

In this paper, we share our experience with generating and evaluating KGEs for the AD domain. To the best of our knowledge, this is the first attempt of its kind in this domain. Our investigation begins with two popular benchmark datasets from Aptiv and Lyft. From each of these datasets we generate multiple KGs with varying degrees of informational detail. The generated KGs focus on representing the various scenes, or situations, that an autonomous vehicle encounters on the road. The purpose of creating KGs with varying degrees of detail is to enable an examination of the relationship between KG detail and the quality of derivative embeddings. Each KG is then translated into a set of KGEs, each derived from one-of three popular embedding algorithms; including TransE (Bordes et al. 2013), RESCAL (Nickel, Tresp, and Kriegel 2011), and HoLE (Nickel, Rosasco, and Poggio 2016). Finally, the quality of each KGE is systematically evaluated based on the framework proposed in (Alshargi et al. 2019).

[1]https://www.digitaltrends.com/cool-tech/neuro-symbolic-ai-the-future/

Our analysis of evaluating the KGEs along four dimensions – i.e. quality metrics, KG informational detail, algorithms, and datasets – leads to some interesting findings. First, we show that KGE quality significantly improves as the informational detail of a KG increases. Second, focusing on the evaluation measures, we report that some of the metrics such as the coherence measure may not be suitable to evaluate KGEs in this domain. When considering the effectiveness of KGE algorithms, we identify that the semantic transitional distance-based TransE algorithm captures type and relational semantics better than algorithms from the class of semantic matching-based models. It is interesting to note that these findings are consistent across the evaluations on two datasets. Finally, we report preliminary observations on using KGEs for two use cases from the AD domain. Specifically, we demonstrate how the scene/sub-scene understanding was improved as KG informational detail was increased, and how KGEs can be used to compute scene similarity.

The two primary contributions of this paper include: (1) a demonstration of the process of creating and evaluating KGEs for AD data, and (2) an (early) examination of the relationship between KG detail and the quality of KGEs. In Section 2, we discuss the construction of KGs from the benchmark automotive driving datasets. The translation of KGs to KGEs is explained in Section 3, while Section 4 focuses on their evaluation. Details of the technology used, as well as related work, will be discussed in each individual section. An investigation on the usefulness of semantics in the AD domain is discussed in Section 5. Finally, in Section 6 we conclude with a summary of our overall results and directions for future research.

2 Scene Knowledge Graphs

To evaluate the KGEs for the AD domain, several KGs were created based on two popular benchmark datasets; NuScenes from Aptiv (Caesar et al. 2019) and Lyft-Level5 from Lyft (Kesten et al. 2019). To annotate the data from the datasets, a scene ontology was used.

2.1 Composition of the Datasets

Both the NuScenes and Lyft datasets follow a similar structure. NuScenes, for example, is divided into a set of 20 second driving segments/scenes, with ∼40 samples/sub-scenes per segment (i.e. one sample/sub-scene every 0.5 seconds). Each 20 second segment is associated with a temporal interval and spatial area, while each sample is associated with the data collected at a specific temporal instant (i.e. timestamp) and spatial coordinates.

The NuScenes dataset contains 850 driving segments with 34,149 samples. Each object and event detected in a sample is associated with one of 23 categories[2].

The Lyft dataset contains 180 driving segments with 22,680 samples. Lyft has only 9 object and event categories that are used for annotating samples.

[2]https://www.nuscenes.org/data-annotation

2.2 Scene Ontology

A scene is described as an observable volume of time and space (Henson et al. 2019). In the AD domain, a scene depicts a situation encountered by a vehicle. A few examples may include a vehicle stopped at a traffic light, cruising on the highway, or crashing into another vehicle. The concept of scene acts as the polestar with which all information about the vehicle, and its situation, are integrated. More specifically, a scene may include information about time and location, the occurring events, and the participating objects.

A scene may also include sub-scenes. For example, consider a vehicle driving for 20 seconds on a highway. This drive may be represented as a single scene. However, during this drive the vehicle may encounter several different situations, each of which may also be represented as a scene.

Formally, Figure 1 shows the properties associated with a scene (depicted in Protege[3]).

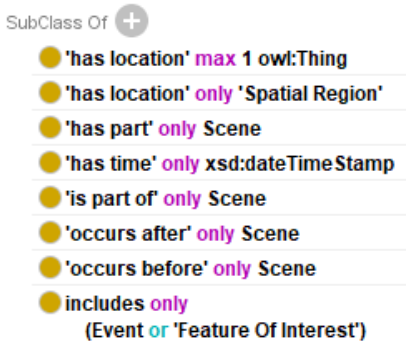


Figure 1: Formal definition of a scene, as defined by the Scene Ontology

A subset of the events and features-of-interest (i.e. objects) represented within the Scene Ontology is shown in Figure 2. For this work, the Scene Ontology has been extended to subsume all concepts found in both the NuScenes and Lyft datasets.

2.3 Informational Detail of KGs

For each dataset, three distinct KGs are generated with differing levels of informational detail. It should be noted that the level of informational detail for each KG refers to the inclusion of additional information about scenes. The three levels include: (1) a base KG, (2) a KG with inferred *type* relations for objects/events, and (3) a KG with additional *includes* relations between scenes and object/events. It should be noted that this additional information does not correspond to an increase in the logical expressivity of the KGs. It is also worth mentioning that each KG includes the Scene Ontology along with the facts derived from each dataset.

Base KG Within the Base KG, each 20 second segment is instantiated as a scene, and each sample is instantiated as a sub-scene. The scene representing a 20 second segment is associated with a temporal interval (of 20 seconds) and

[3]https://protege.stanford.edu/

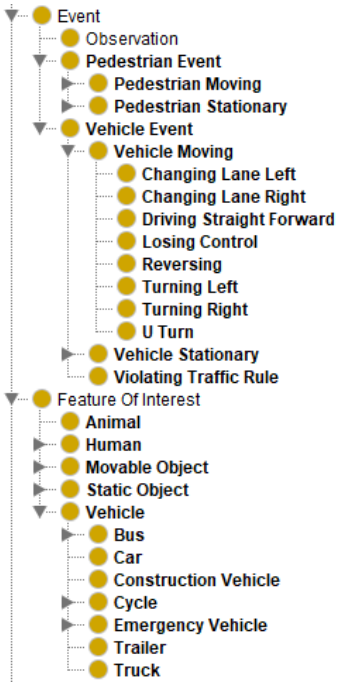


Figure 2: Subset of events and features-of-interest contained in the Scene Ontology.

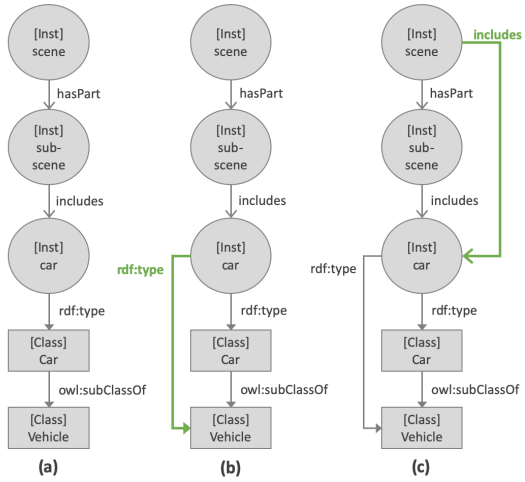


Figure 3: Example KGs: (a) Base KG, (b) KG w/ inferred types, and (c) KG w/ include path

a spatial location (e.g. a city). This scene is also associated with a set of sub-scenes, representing the samples. Each sub-scene is associated with a temporal instant, spatial point, and the objects and events that participate in the scene. See Figure 3(a) for an example.

KG with Inferred Types In the Base KG, objects and events are explicitly typed to the most specific class possible. For example, an object instance representing a car is typed to the *Car* class. Because the *Car* class is a sub-class of *Vehicle*, then the instance is also a type of *Vehicle*. However, this knowledge is only implicit in the KG; implied by the semantics of the `owl:subClassOf` relation. To make this knowledge explicit, a reasoner is used to infer all implied types for each object and event instance. See Figure 3(b) for an example.

Example RDF (new triples proceeded by →)
:inst-scene rdf:type scene:Scene .
:inst-scene scene:hasPart :inst-sub-scene .
:inst-sub-scene includes :inst-car .
:inst-car rdf:type scene:Car .
→ *:inst-car rdf:type scene:Vehicle* .
→ *:inst-car rdf:type scene:FeatureOfInterest* .

KG with Include Paths Within the Base KG, objects and events are associated with sub-scenes derived from samples of a 20 second drive. The scene representing the entire drive is associated with these participating objects and events through a two-hop path.

Example RDF
:inst-scene scene:hasPart :inst-sub-scene .
:inst-sub-scene scene:includes :inst-car .

In order to make a more direct association between a scene representing a drive and the detected objects and events, new *includes* relations are added to the KG. See Figure 3(c) for an example.

Example RDF (new triples proceeded by →)
:inst-scene scene:hasPart :inst-sub-scene .
:inst-sub-scene scene:includes :inst-car .
→ *:inst-scene scene:includes :inst-car* .

Table 1 summarizes the statistics of the three KG versions we have generated.

		Base	W/ inferred types	W/ include paths
NuScenes	# triples	5.95M	8.78M	10.80M
	# entities	2.11M		
	# relations	11		
Lyft	# triples	3.94M	5.85M	7.12M
	# entities	1.33M		
	# relations	11		

Table 1: Statistics of the three KG versions generated from the NuScenes and Lyft datasets

3 KG Embeddings for AD Scenes

The goal of learning embeddings from a KG is to represent the entities and relations in low-dimensional vector space while also maintaining the semantics contained in the KG. This transformation allows KGs to be more easily manipulated and used for downstream learning tasks (e.g. link pre-

diction (Xiao, Huang, and Zhu 2016) and KG completion (Lin et al. 2015)). Vector representation of KGEs also allows background knowledge contained in KGs to be easily integrated with other input features of a machine learning model.

To select candidate algorithms for our experiments, we referred to the classification of KGE algorithms established by (Wang et al. 2017) and (Sharma, Talukdar, and others 2018). KGE algorithms are categorized into two main classes: (1) Transitional distance-based algorithms and (2) Semantic Matching Models. For transitional distance-based algorithms (i.e. additive methods) the scoring function is composed of distance measures, and vector addition/subtraction is used to capture the vector interaction. For Semantic Matching Models (i.e. multiplicative methods) the scoring function is based on a similarity measure, and entity-relation-entity interaction is captured via a multiplicative score function. We initially selected one algorithm from each class. Specifically, we selected TransE from the former category and RESCAL from the latter category. RESCAL, however, has limitations in handling big KGs due to its high space and time complexity. As a result, we also included HolE into our experiments, which is a more space and memory-efficient successor of RESCAL.

3.1 Preliminaries

Next, the symbols and notations used throughout the paper are introduced, along with important details of the KGE algorithms.

Notation: Given a set of entities $\mathcal{E}$ and set of relations $\mathcal{R}$, we define KG to be a set of triples (h, r, t), $\mathcal{T} = \mathcal{E} \times R \times (\mathcal{E} \cup \mathcal{L})$ where $\mathcal{L}$ is the set of literals. We consider $\mathcal{E} = \mathcal{C} \cup \mathcal{N}$ where $\mathcal{C}$ is set of concepts from the ontology and $\mathcal{N}$ is set of individuals. Lowercase bold characters represent vectors of an entity or relation and uppercase bold characters represent a set of vectors. For example, $\mathbf{e} \in \mathbf{E}$ is an embedding vector of $e \in \mathcal{E}$. Most of the embedding algorithms – including some used in our evaluation – generate vectors of dimension d to represent entities $\mathbf{e} \in \mathbb{R}^d$ for $e \in \mathcal{E}$ and $\mathbf{r} = \mathbb{R}^d$ to represent relations $r \in \mathcal{R}$. However, some algorithms learn a projection matrix $M_r \in \mathbb{R}^{d \times d}$ to represent relations. The scoring function $\sigma : \mathcal{E} \times \mathcal{R} \times \mathcal{E} \rightarrow \mathbb{R}$ used in each algorithm is different. Transitional distance-based models use distance measures whereas semantic matching based methods use similarity measures. The learning of embeddings involve optimizing parameter θ in the loss function $\mathcal{L}(\mathcal{T}, \mathcal{T}', \theta)$ where $\mathcal{T}$ is the set of positive triples and $\mathcal{T}'$ is the set of corrupted triples. When considering time and space complexities of each algorithm, we consider $n = |E|$, $m = |R|$ and n_t to be the number of training triples.

The details of each algorithm used are briefly discussed below;

TransE TransE is considered the most representative of the translational distance-based class of algorithms. Given a triple (h, r, t), TransE represents r as a translation vector from h to t. Hence $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$ when the triple (h, r, t) holds true. The scoring function of TransE f_r is defined as the negative distance between $\mathbf{h} + \mathbf{r}$ and $\mathbf{t}$, and when the (h, r, t) holds, f_r is expected to be large.

$$f_r(h, t) = -|\mathbf{h} + \mathbf{r} - \mathbf{t}|_{1/2} \tag{1}$$

TransE is one of the most efficient KGE algorithms having $\mathcal{O}(nd + md)$ space complexity and $\mathcal{O}(n_t d)$ time complexity. Despite it's benefits, TransE falls short in capturing 1-N, N-1 and N-N relations in KGs.

RESCAL RESCAL belongs to the semantic matching/multiplicative class of KGE algorithms. RESCAL is an expressive model which takes into account the inherent structure in multi-relational KGs and captures complex patterns over multiple hops in the KG. It represents each relation r as matrix M_r that captures all the interaction between vectors $(\mathbf{h}, \mathbf{t})$ of the entities h and r. The scoring function $f_r(h, t)$ is a bi-linear function which computes pairwise interaction of entities with respect to each relation r.

$$f_r = \mathbf{h}^T \mathbf{M}^r \mathbf{t} = \sum_{i=0}^{d-1} \sum_{j=0}^{d-1} [\mathbf{M}_r]_{ij} . [\mathbf{h}]_i . [\mathbf{t}]_j \tag{2}$$

The main limitation of using RESCAL with big KGs is due to its high space and time complexity. RESCAL has $\mathcal{O}(nd + md^2)$ space complexity and $\mathcal{O}(n_t d^2)$ time complexity.

HolE HolE is an efficient successor of RESCAL and addresses the high space complexity of RESCAL while retaining the expressive power. HolE represents both entities and relations as vectors in $\mathbb{R}^d$. Given a triple (h, r, t), HolE first creates a compositional vector using circular correlation operation which aims at compressing the pairwise interaction;

$$[\mathbf{h} \star \mathbf{t}] = \sum_{k=0}^{d-1} [\mathbf{h}]_k . [\mathbf{t}]_{(k+i) \mod d} \tag{3}$$

The total score for a given fact is then calculated by using the function $f_r(h, t)$ that considers both the compositional vector and the relation vector.

$$f_r(h, t) = \mathbf{r}^T (\mathbf{h} \star \mathbf{t}) = \sum_{i=0}^{d-1} [\mathbf{r}]_i \sum_{k=0}^{d-1} [\mathbf{h}]_k . [\mathbf{t}]_{(k+i) \mod d} \tag{4}$$

Due to the use of circular correlation, HolE achieves $\mathcal{O}(nd + md)$ space complexity and $\mathcal{O}(n_t d \log d)$ time complexity.

3.2 Visualizing KG Embeddings

We selected 10 driving segments (including their samples) from each dataset - NuScenes and Lyft - and created "mini" KGs to visualize the embeddings in 2-dimensional (2D) space. After experimenting with both PCA (Wold, Esbensen, and Geladi 1987) and t-Distributed Stochastic Neighbor Embedding (t-SNE)(Maaten and Hinton 2008), t-SNE was selected for dimensionality reduction as its 2D projections yielded more meaningful clusters than PCA for the generated embeddings. The embedding dimension d was set to 100 when generating embeddings for all our experiments.

Our extended Scene Ontology identifies *events* and *features-of-interests (FoI)* as top-level classes in the ontology, and each instance of *event* or *FoI* are linked to Scenes via the *includes* relation. *FoIs* are related to *events* through the *isParticipantOf* relation. Therefore, we first look at how FoIs and events are manifested in the embedding space for each dataset.

NuScenes KG Embeddings Figure 4 shows how the events and FoIs form clusters in the embedding space based on their type. For the sake of brevity, only *cars* and the events in which they participate are highlighted. From this visualization, you can see that instances of events such as *stopped car*, *moving car* and *parked car* are clustered around the instances of *car*. The embeddings represented in this figure are generated from TransE on the "Base KG".

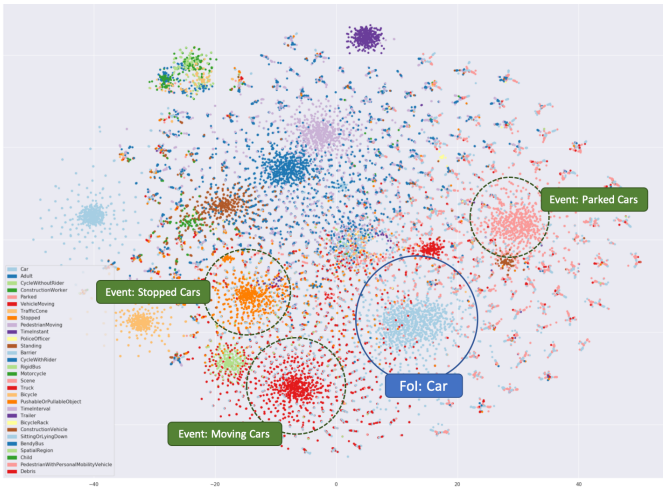


Figure 4: Clustering of FoIs with events in NuScenes dataset

Lyft KG Embeddings Figure 5 shows a similar visualization of the embeddings from the Lyft "Base KG". This image shows how the events in Lyft are clustered together with FoIs. It may be noticed that Lyft contains fewer clusters than NuScenes. This is the case since the Lyft dataset only contains annotations for a few FoIs. Similar to what we've seen with NuScenes, Lyft embeddings also show how the instances of events such as *stopped car*, *parked car* and *driving straightforward* are clustered around instances of *car*.

4 Evaluation

The primary objective of our evaluation is to determine how well the salient features and rich semantics of KGs, such as type and relation semantics, transfer to learned embeddings. Our evaluation deviates from most prior work evaluating KGE algorithms, which focus on evaluating the performance of some extrinsic downstream task (such as entity classification). Here, we focus more on an intrinsic evaluation of embeddings.

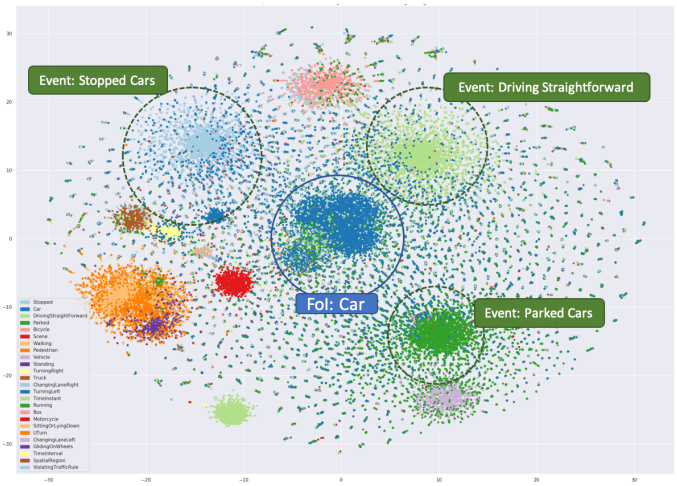


Figure 5: Clustering of FoIs with events in Lyft dataset

4.1 Evaluation of KG Embeddings

There exists a large body of literature that evaluates the effect of using KGEs on a downstream task. To our knowledge, however, there's only one recent work which introduces metrics to evaluate and quantify the intrinsic quality of embeddings. Of the metrics introduced in (Alshargi et al. 2019), we adapt three metrics for our evaluation: *categorization measure, coherence measure*, and *semantic transition distance*. Figure 6 depicts the four dimensions involved in our evaluation – i.e. quality metrics, datasets, KGE algorithms and KGs with varying degrees of informational detail.

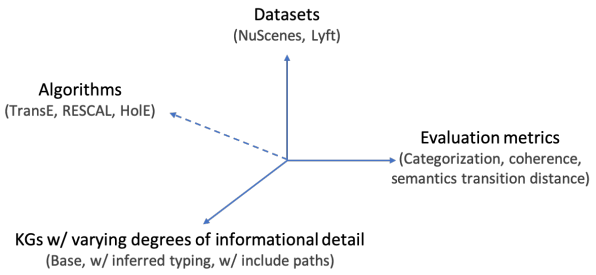


Figure 6: Four dimensions involved in our evaluation

Next we provide a brief overview of each of the evaluation metrics;

The **categorization measure** captures how well the entities that are "typed" by the same background concept cluster together. For example, all the entities that are typed by the concept *Car* ($\forall e_i \in c_{k=Car}$) share common characteristics and should be clustered together. Hence this metric is computed by taking the cosine similarity $s(V_1, V_2) = \frac{V_1 . V_2}{|V_1||V_2|}$ of the averaged embedding vector (equation 5) of all such entities and the embedding vector of the background concept k, $\mathbf{c_k}$.

$$\forall e_i \in c_k, \bar{\mathbf{e}}_\mathbf{k} = \frac{1}{n}\sum_{i=1}^{i=n} \mathbf{e_i} \tag{5}$$

$$Categorization(\mathbf{c_k}) = s(\bar{\mathbf{e}}_\mathbf{k}, \mathbf{c_k}) \tag{6}$$

The **coherence measure** captures whether the adjacent entities in the embedding space share a common background concept. In introducing this measure, authors hypothesise that in the ideal case, all the entities that are typed by the same background concept should form a cluster and the background concept should be the centroid of this cluster. This is quantified (equation 7) by taking n closest entities of the background concept c_i and taking the proportion of which that are actually typed by c_i. For our experiments, we choose n to be 1000.

$$Coherence(\mathbf{c_i}) = \frac{\#\mathbf{e_i}|\mathbf{e_i} \in c_i}{n} \tag{7}$$

The **semantic transition distance** is a widely used metric in word embedding literature and re-introduced to KGEs to capture the relational semantics of KGs. For example, assume h_i is asserted as the domain of the property r_i and t_i is asserted as its range. Then, if (h_i, r_i, t_i) is correctly represented in the embedding space, the transition distance between $\mathbf{h_i} + \mathbf{r_i}$ should be close to $\mathbf{t_i}$. This is formally represented in equation 8 where T_r is semantic transition distance of relation r and s denotes cosine similarity.

$$T_r(h_i + r_i, t_i) = s(\mathbf{h_i} + \mathbf{r_i}, \mathbf{t_i}) \tag{8}$$

Next we report our evaluation results of these three metrics with respect to each dataset/algorithm.

Evaluation on the Lyft Dataset Figure 7 summarizes the results of the categorization measure computed on the embeddings generated from three algorithms on three KG versions. It is clear from the figures that TransE performs better compared to RESCAL and HolE and the categorization quality is mostly better in the KG with more informational detail (i.e. KG w/ include paths) compared to other two less expressive variants. In our experimental setting, this measure is computed considering the top level concepts (FoIs and events) in the Scene Ontology.

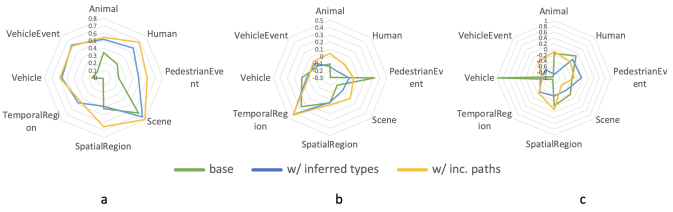


Figure 7: Categorization measure computed for different KGs on the embeddings generated from (a) TransE, (b) RESCAL and (c) HolE

The results of the coherence measure, as depicted in figure 8, shows a similar trend as the categorization measure; TransE is performing better and both RESCAL and HolE fail to generate entity clusters with high purity and closer to the background concept of those entities. It is interesting to note that, the KG with highest informational detail shows significant improvement in coherence measure on the embeddings generated from TransE.

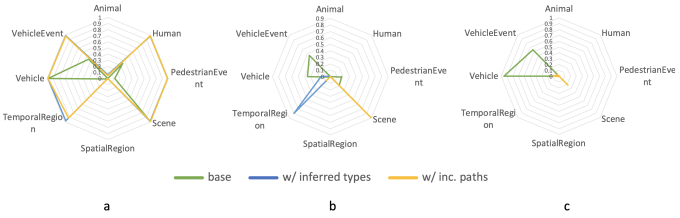


Figure 8: Coherence measure computed for different KGs on the embeddings generated from (a) TransE, (b) RESCAL and (c) HolE

Next we look at how the relational semantics in KGs are transferred to KGEs by computing semantic transition distance for 11 relations defined in the Scene ontology. As per figure 9, KGs with include paths are able to capture relational semantics better than the other two variants across all three algorithms. An interesting observation to note here is that the *isPartOf* relation performs significantly better in KGs with include paths across all three algorithms even though we have only added implicit *include* relations. A possible explanation could be that the implicit *include* paths make the relationship between scenes and sub-scenes stronger in KGs with the highest informational detail.

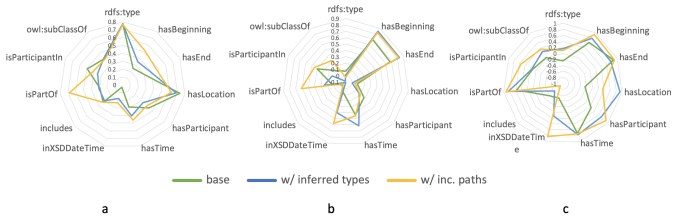


Figure 9: Semantic transition distance computed for different KGs on the embeddings generated from (a) TransE, (b) RESCAL and (c) HolE

Evaluation on the NuScenes Dataset The evaluation process on the NuScenes dataset is similar to Lyft. However, we report that RESCAL was not scalable to the NuScenes KGs (having 10.8+ million triples and 2.1+ million entities). Therefore, we evaluate NuScenes KGEs only on TransE and HolE.

The results of the categorization measure for the NuScenes dataset follows a similar trend as Lyft (see Figure 10). TransE embeddings on KG w/ include paths yields the best categorization performance, with the exception of a few concepts where HolE outperforms on the base KG. Except for these few outliers, the results show that higher level of informational detail in KG achieves better categorization irrespective of the KGE algorithms used for training.

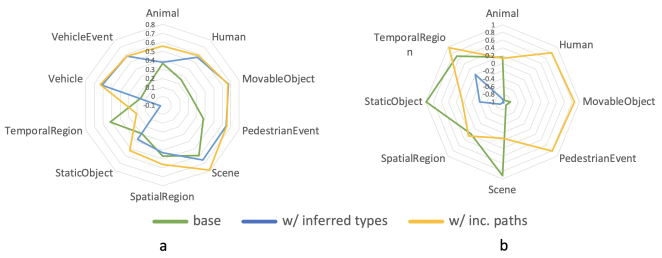


Figure 10: Categorization measure computed for different KGs on the embeddings generated from (a) TransE and (b) HolE

The benefits of information detail in KG are portrayed well in the results of coherence measure (see Figure 11). Even though the coherence measure, for many concepts, are either non-existent or closer to zero, the KG w/ include paths significantly outperform the other two KG variants for those values that exist.

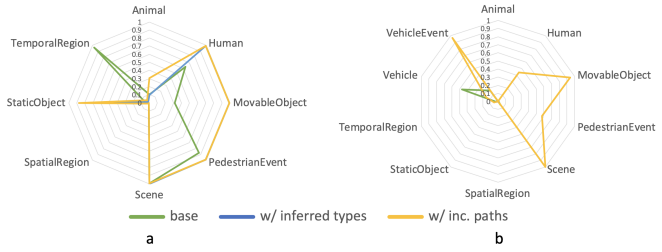


Figure 11: Coherence measure computed for different KGs on the embeddings generated from (a) TransE and (b) HolE

The results of the semantic transition distance for TransE show consistent patterns similar to Lyft (see Figure 12). HolE, however, shows that base KGEs perform on par with embeddings trained on the KG w/ include paths.

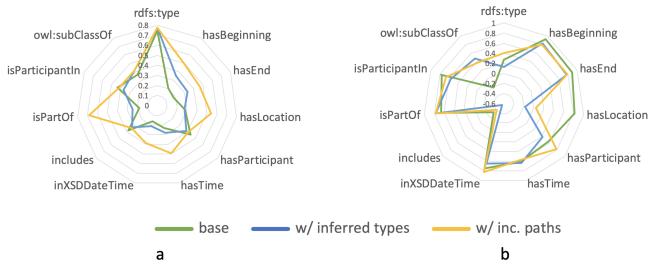


Figure 12: Semantic transition distance computed for different KGs on the embeddings generated from (a) TransE and (b) HolE

4.2 Discussion

The evaluation of KGEs for AD domain lead to some interesting observations. We discuss our observations in three perspectives: (1) KGE algorithmic perspective, (2) evaluation measures, and (3) various levels of KG informational detail.

First, looking at the overall performance of KGE algorithms, TransE performs better than RESCAL and HolE in capturing both type and relational semantics. TransE is also scalable to large KGs and shows consistent performance across datasets. In addition to RESCAL's space and time complexity, it's performance on all three metrics is worse than TransE and HolE. Even though HolE's performance is sub-optimal compared to TransE, it was consistent across the two datasets and the derived KGs. We hypothesize that the better performance of TransE on all three metrics is due to the way it is learning embeddings; i.e. using the translational distance based scoring function inspired by word embedding algorithms. The KGE quality metrics introduced by (Alshargi et al. 2019) are inspired by the word embedding literature. They evaluated these metrics on KGEs generated from word embedding based RDF2Vec (Ristoski and Paulheim 2016) algorithm. Hence, it may be worth examining whether these metrics are suitable only for evaluating embeddings generated from translational distance / word embedding inspired KGE algorithms.

Second, when considering the evaluation measures used, we observe that the coherence measure is not very meaningful in this domain to evaluate the quality of KGEs. The entities are mostly clustered based on either scenes/sub-scenes or FoIs/events. Hence, the n most similar entities to a class (e.g. *Human*) are mostly not homogeneous, resulting in zero or close to zero coherence value.

Third, it has been consistently shown across multiple datasets and algorithms that KGs with the highest levels of informational detail are able to capture both type and relational semantics better than the other two less expressive variants. This discovery leads to an interesting future direction for research. To the best of our knowledge, all existing KGE algorithms in the literature are evaluated on base KGs (i.e. KGs without any inference). Therefore, it stands to reason that a KG embedding derived from a KG with more informational detail should capture more salient features and rich semantics of the KG. Such informational details can be captured either through pre-processing or by automatically extracted by the KGE algorithm.

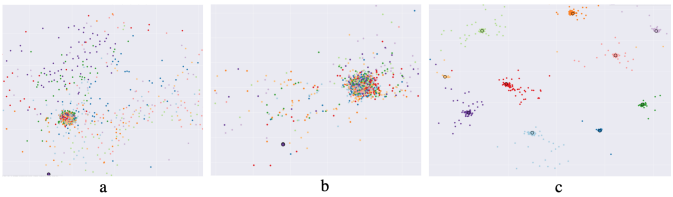
5 Investigating Semantics of AD Domain

In the previous section we looked at different quality aspects of embeddings which are common to KGs in any domain. Here, we report additional experiments which look at certain aspects specific to AD domain and scene understanding.

5.1 Scene/sub-scene Relationship

An understanding of complex AD scenes is an important task in AD domain. The ability to distinguish one complex scene from another requires looking at (1) how the scene/sub-scene relationship (formally defined by *isPartOf* relation) is captured by the embeddings in different KG versions, and (2) how well the FoIs and events cluster based on the scenes/subscenes they belong to.

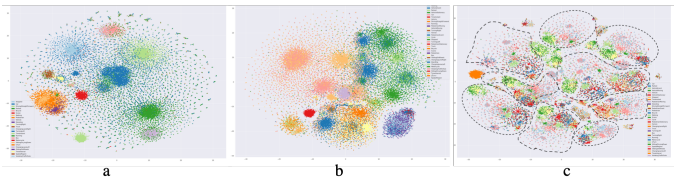
Figure 13 shows the manifestation of the scene-subscene relationship that is moving from a "no-relation" (figure 13(a)) in the base KG to a more "meaningful" one (figure 13(c)) in the KG with the highest informational detail. Then we look at how the various levels of informational detail in KGs affect the clustering of FoIs and events of a scene based on scene/sub-scene relationship.



a b c

Figure 13: Clustering of scenes with sub-scenes in different KG versions of Lyft: (a) base KG, (b) KG w/ inferred types, and (c) KG w/ include paths

Figure 14 shows how the FoI/event dominant clusters in the base KG transfer to a clustering based on 10 scenes in the KG w/ include paths. Interestingly, we can still see small clusters formed based on FoIs/events inside the larger scene clusters. This suggests that the KGs with access to more informational detail are able to distinguish a scene by both participating FoIs/events as well as scene/sub-scene relationships.

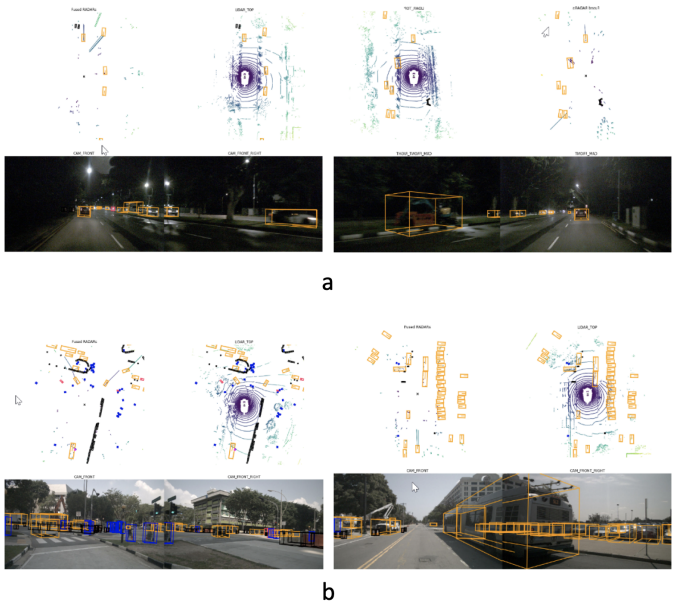


a b c

Figure 14: Clustering of FoIs and events together with scenes/sub-scenes in different KG versions: (a) base KG, (b) KG w/ inferred types, and (c) KG w/ include paths

5.2 KGEs for Computing Scene Similarity

We report preliminary results of using KGEs for computing scene similarity. Our objective here is to determine whether two scenes are similar by considering only KGEs. Given a set of scene pairs, we calculate the cosine similarity between the KGE vectors of two scenes in a pair and then select pairs with the highest cosine similarity. Figure 15(a) shows the two most similar sub-scenes when pairs include sub-scenes from the same parent scene. With further investigation, we found that these two sub-scenes are in fact subsequent frames (or samples) from the same 20 second driving segment. Figure 15(b) shows the two most similar sub-scenes when pairs contain only sub-scenes from different scenes. It is interesting to note that the KGE based similarity computation was able to identify two sub-scenes which are not visually similar, but share common characteristics. For example, the black string of objects in Figure 15(b) (Left) are barriers (a *Static Object*) and the orange string of objects in Figure 15(b) (Right) are set of stopped cars.



a

b

Figure 15: Most similar sub-scenes computed using KGEs trained on NuScenes Base KG; (a) sub-scenes from the same scene and (b) sub-scenes from two different scenes

6 Conclusion

In this paper, we present an evaluation of KGEs for the autonomous driving domain that considers multiple datasets, metrics, algorithms and levels of informational detail. This evaluation supports the hypothesis that a KG with more detailed information yields higher quality KG embeddings with respect to both type and relational semantics. Furthermore, this evaluation highlights an important question about the suitability of metrics used in the existing literature, to evaluate a wide range of KGE algorithms. Finally, opening rich areas for future research, we present an early investigation into the use of KGEs for two important use-cases from the AD domain: scene/sub-scene understanding and computing scene similarity.

References

Alshargi, F.; Shekarpoour, S.; Soru, T.; and Sheth, A. 2019. Metrics for evaluating quality of embeddings for ontological concepts. In *AAAI 2019 Spring Symposium on Combining Machine Learning with Knowledge Engineering (AAAI-MAKE 2019)*.

Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; and Yakhnenko, O. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, 2787–2795.

Caesar, H.; Bankiti, V.; Lang, A. H.; Vora, S.; Liong, V. E.; Xu, Q.; Krishnan, A.; Pan, Y.; Baldan, G.; and Beijbom, O.

2019. nuscenes: A multimodal dataset for autonomous driving. *ArXiv* abs/1903.11027.

Chen, M.; Tian, Y.; Yang, M.; and Zaniolo, C. 2017. Multilingual knowledge graph embeddings for cross-lingual knowledge alignment. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 1511–1517. AAAI Press.

Henson, C.; Schmid, S.; Tran, T.; and Karatzoglou, A. 2019. Using a knowledge graph of scenes to enable search of autonomous driving data. In *Proceedings of the 2019 International Semantic Web Conference (ISWC 2019)*.

Kesten, R.; Usman, M.; Houston, J.; Pandya, T.; Nadhamuni, K.; Ferreira, A.; Yuan, M.; Low, B.; Jain, A.; Ondruska, P.; Omari, S.; Shah, S.; Kulkarni, A.; Kazakova, A.; Tao, C.; Platinsky, L.; Jiang, W.; and Shet, V. 2019. Lyft level 5 av dataset 2019. https://level5.lyft.com/dataset/.

Lin, Y.; Liu, Z.; Sun, M.; Liu, Y.; and Zhu, X. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Twenty-ninth AAAI conference on Artificial Intelligence*.

Litman, T. 2019. Autonomous vehicle implementation predictions: Implications for transport planning. https://www.vtpi.org/avip.pdf.

Maaten, L. v. d., and Hinton, G. 2008. Visualizing data using t-sne. *Journal of machine learning research* 9(Nov):2579–2605.

Myklebust, E. B.; Jimenez-Ruiz, E.; Chen, J.; Wolf, R.; and Tollefsen, K. E. 2019. Knowledge graph embedding for ecotoxicological effect prediction. In *International Semantic Web Conference*, 490–506. Springer.

Nickel, M.; Murphy, K.; Tresp, V.; and Gabrilovich, E. 2015. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE* 104(1):11–33.

Nickel, M.; Rosasco, L.; and Poggio, T. 2016. Holographic embeddings of knowledge graphs. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Nickel, M.; Tresp, V.; and Kriegel, H.-P. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, 809–816. Omnipress.

Ristoski, P., and Paulheim, H. 2016. Rdf2vec: Rdf graph embeddings for data mining. In *International Semantic Web Conference*, 498–514. Springer.

Sharma, A.; Talukdar, P.; et al. 2018. Towards understanding the geometry of knowledge graph embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 122–131.

Sheth, A.; Gaur, M.; Kursuncu, U.; and Wickramarachchi, R. 2019. Shades of knowledge-infused learning for enhancing deep learning. *IEEE Internet Computing* 23(6):54–63.

Wang, Q.; Mao, Z.; Wang, B.; and Guo, L. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering* 29(12):2724–2743.

Wang, H.; Zhang, F.; Zhao, M.; Li, W.; Xie, X.; and Guo, M. 2019. Multi-task feature learning for knowledge graph enhanced recommendation. In *The World Wide Web Conference*, 2000–2010. ACM.

Wold, S.; Esbensen, K.; and Geladi, P. 1987. Principal component analysis. *Chemometrics and intelligent laboratory systems* 2(1-3):37–52.

Xiao, H.; Huang, M.; and Zhu, X. 2016. From one point to a manifold: knowledge graph embedding for precise link prediction. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, 1315–1321. AAAI Press.

Appendices

We include detailed evaluation results with respect to each dataset, evaluation metric, KG and algorithm in appendices. Tables (2, 3, 4) in appendix A contains results of the Lyft dataset whereas appendix B contains tables (5, 6, 7) summarizing the results of the NuScenes dataset.

Appendix A: Evaluation Results of the Lyft dataset

	TransE			RESCAL			HolE		
Class	base	w/ types	w/ paths	base	w/ types	w/ paths	base	w/ types	w/ paths
Animal	0.3404	0.5192	0.5420	-0.1072	-0.1420	0.0405	-0.1465	-0.8855	-0.0796
Human	0.2570	0.5644	0.6767	-0.2916	-0.1684	-0.0281	0.0867	-0.0756	-0.2007
PedestrianEvent	0.2057	0.4691	0.5891	0.3228	-0.0310	0.0271	-0.3618	-0.0415	-0.2968
Scene	0.6654	0.7390	0.7904	-0.1591	-0.0511	0.1011	-0.1861	-0.3845	-0.6410
SpatialRegion	0.4127	0.3791	0.6552	0.0443	0.0373	0.0735	-0.0349	-0.3709	0.1069
TemporalRegion	0.0075	0.4787	0.4500	0.2709	0.4206	0.4332	-0.9270	-0.2917	-0.2464
Vehicle	0.1604	0.5616	0.5898	0.0997	0.0286	0.0045	0.9681	-0.5235	-0.5934
VehicleEvent	0.1173	0.6167	0.6063	-0.0617	-0.0114	0.0259	-0.9617	-0.5908	-0.2173

Table 2: Results of categorization measure computed on embeddings generated from TransE, RESCAL and HolE

	TransE			RESCAL			HolE		
Class	base	w/ types	w/ paths	base	w/ types	w/ paths	base	w/ types	w/ paths
Animal	0.002	0.062	0.047	0	0	0	0	0	0
Human	0.346	0.986	0.991	0.001	0	0	0	0.002	0
PedestrianEvent	0.117	0.983	0.986	0.176	0	0	0.001	0	0
Scene	0.996	0.996	0.998	0.193	0.674	0.89	0.004	0.003	0.215
SpatialRegion	0.001	0.001	0.001	0	0	0	0	0	0
TemporalRegion	0.002	0.982	0.917	0.001	0.794	0.077	0	0	0.001
Vehicle	0.987	0.99	0.99	0.352	0.146	0	0.947	0	0.108
VehicleEvent	0.445	0.987	0.986	0.46	0.009	0	0.642	0	0.036

Table 3: Results of coherence measure computed on embeddings generated from TransE, RESCAL and HolE

	TransE			RESCAL			HolE		
Relation	base	w/ types	w/ paths	base	w/ types	w/ paths	base	w/ types	w/ paths
rdfs:type	0.7650	0.7744	0.7626	0.0722	0.0119	-0.0243	-0.2436	0.1521	0.0890
hasBeginning	0.2483	0.3510	0.5145	0.6912	0.8491	0.8285	0.6179	0.7860	0.9338
hasEnd	0.3134	0.3939	0.4732	0.6720	0.8314	0.8203	0.8675	0.7588	0.9001
hasLocation	0.7310	0.6824	0.6261	0.0877	0.0048	0.0481	-0.2506	0.9296	0.3473
hasParticipant	0.4392	0.3416	0.4010	0.2772	0.1781	0.2010	0.2363	0.6978	0.8979
hasTime	0.2883	0.4057	0.4601	0.4375	0.6180	0.4592	0.7264	0.7720	0.7412
inXSDDateTime	0.0260	0.1786	0.2335	0.0589	0.4005	0.5822	-0.4892	0.1231	0.8463
includes	0.3271	0.3051	0.3327	-0.0341	-0.0945	-0.0537	-0.3785	-0.6282	-0.8829
isPartOf	0.2893	0.3039	0.6799	0.1291	0.2470	0.6166	0.5646	0.8461	0.9255
isParticipantIn	0.4964	0.3480	0.4121	0.3967	0.1403	0.4515	0.1110	0.1449	0.5452
owl:subClassOf	0.4036	0.4253	0.3969	0.1133	-0.0816	0.3055	0.0297	0.2703	0.3664

Table 4: Semantic transition distance computed on embeddings generated from TransE, RESCAL and HolE

Appendix B: Evaluation Results of the NuScenes dataset

	TransE			HolE		
Class	base	w/ types	w/ paths	base	w/ types	w/ paths
Animal	0.3649	0.3769	0.5565	0.1582	-0.9250	0.1250
Human	0.2530	0.5573	0.5798	-0.9436	-0.9865	0.7919
MovableObject	0.2276	0.6694	0.6653	-0.7962	-0.9832	0.8592
PedestrianEvent	0.3798	0.6447	0.6514	-0.8769	-0.9729	0.8101
Scene	0.5931	0.6585	0.7950	0.9181	-0.9856	-0.0491
SpatialRegion	0.4701	0.4313	0.5618	0.1635	-0.9134	0.2596
StaticObject	0.2914	0.3748	0.5274	0.9906	-0.4043	0.0409
TemporalRegion	0.5120	-0.0722	0.2073	0.6798	-0.0011	0.9696
Vehicle	0.1556	0.6140	0.6352	0.7423	-0.7442	0.8521
VehicleEvent	0.1761	0.5740	0.5768	-0.9159	-0.5617	-0.8291

Table 5: Results of categorization measure computed on embeddings generated from TransE and HolE

	TransE			HolE		
Class	base	w/ types	w/ paths	base	w/ types	w/ paths
Animal	0.099	0.096	0.308	0	0	0
Human	0.63	0.992	0.995	0.014	0.001	0.441
MovableObject	0.315	0.994	0.995	0	0	0.944
PedestrianEvent	0.875	0.995	0.994	0.005	0.003	0.574
Scene	0.997	0.999	0.997	0.049	0	0.999
SpatialRegion	0.004	0.004	0.004	0	0	0
StaticObject	0.044	0.535	0.869	0	0	0
TemporalRegion	0.964	0.019	0.041	0.022	0.001	0.048
Vehicle	0.477	0.984	0.983	0.471	0.007	0.156
VehicleEvent	0.917	0.993	0.992	0.165	0.021	0.965

Table 6: Results of coherence measure computed on embeddings generated from TransE and HolE

	TransE			HolE		
Relation	base	w/ types	w/ paths	base	w/ types	w/ paths
rdfs:type	0.7426	0.7718	0.7733	0.2700	0.1280	0.3995
hasBeginning	0.2040	0.3510	0.4977	0.9243	0.8186	0.7896
hasEnd	0.1811	0.3344	0.4635	0.8960	0.7712	0.7919
hasLocation	0.2823	0.2767	0.5411	0.8240	-0.1710	0.0468
hasParticipant	0.4417	0.3869	0.4057	0.5669	0.4233	0.7795
hasTime	0.2325	0.2837	0.4973	0.6117	0.6332	0.5675
inXSDDateTime	0.1650	0.2134	0.3917	0.7465	0.6430	0.8233
includes	0.3767	0.3311	0.3404	-0.3296	-0.5496	-0.4141
isPartOf	0.1826	0.2498	0.6831	0.6641	0.7712	0.7725
isParticipantIn	0.4307	0.3674	0.4203	0.7694	0.5585	0.6632
owl:subClassOf	0.3646	0.4141	0.4131	-0.2153	0.4608	0.3716

Table 7: Semantic transition distance computed on embeddings generated from TransE and HolE