Knowledge Representation Learning: Using Gaussian Embeddings to Learn Attributive and Hyponymous Relationships

Abstract

Knowledge representation learning aims to represent entities in the knowledge graph using multi-dimensional vectors, which capture the relationships between entities and can be used for various tasks such as knowledge graph completion and relation extraction. According to [1], relations (between entities) stored in knowledge graphs can be separated into three types: interrelation, attribute, and hyponymy. In this paper, we note that attributive and hyponymous relations share various similarities and thus introduce a model that can learn these two types of relations jointly. The model represents each entity, attribute, and hypernym as a multivariate Gaussian distribution and relationships between them as inclusion relationships between the distributions. Here, inclusion means that the mean of an entity's or hyponyms distribution is contained in its attributes or hypernyms distributions high probability region. The model iteratively updates diagonal covariance matrices and means of distributions during training, taking into consideration the global influence of each update. Our model is assessed on triple classification tasks, which require it to determine the truthfulness of unseen relations between entities given some existing relations. Our model achieves excellent performance in this task, indicating its potential in tasks such as automatic knowledge graph completion.

1 Introduction

The importance of general knowledge for artificial intelligence is manifested in recent developments in areas such as natural language processing. Knowledge about entities and concepts in social and natural worlds enhances AI's performance in tasks such as natural language understanding [2] and automatic recommendation[3]. In recent years, knowledge graphs such as Freebase and DBpedia are constructed to store general knowledge as triplets in the form of (*head entity*, *relation*, *tail entity*). For example, a piece of knowledge such as "Barack Obama is an American." will be stored in knowledge graphs (KG) as (*Barack Obama*, *nationality*, *American*). However, these knowledge graphs are still highly incomplete, which gives rise to research interest in automatic knowledge graph completion[4, 5, 6].

Knowledge Representation Learning One branch of research focuses on extending knowledge graphs by training models that can automatically identify the truthfulness of an unseen relational triple (i.e. a triple currently not stored in the graphs). To this end, some proposed to

represent entities and relations in knowledge graphs as vectors that capture objects' semantic information and can thus be used to directly classify triplets [7, 8, 9, 10]. It was later discovered that the trained vectors can be applied to tasks other than knowledge graph completion, such as question answering and language modeling to provide background knowledge. However, when training knowledge representation vectors, most models do not consider the differences between types of relations, leading to varied performance when classifying triplets containing relations of different types [1].

Relations According to [1], relations stored in knowledge graphs can be separated into three types: attribute, hyponymy, and interrelations. Information about and examples of different types can be found in Table 1.

Table 1: Information about Different Types of Relations

Type of Relations	Information Contained	Example Relations
Attribute	attributive information about entities	gender, nationality, taste
Hyponymy	hyponym-hypernym status between entities	species, genre, profession
Interrelation	relations between entities	spouse, friend, record_label

Hyponymous and Attributive Relations We notice that interrelational relations usually have a 1-to-1 mapping, while both hyponymous and attributive relations usually have a 1-to-n mapping. Furthermore, we notice that hyponymous and attributive relationships are highly similar to each other: each attribute can be seen as a hyponym of an entity. For example, in (*apple*, *grow_on*, *trees*), the attribute "*grow_on*, *trees*" can be seen as specifying a group of entities that grows on trees, which can be considered a hyponym of *apple*. On the other hand, hyponyms can be considered as a rich collection of attributes. For instance, in (*lion*, *class*, *mammal*), *mammal*, the hyponym of *lion*, can be seen as a collection of attributes such as $have\_fur\_or\_hair$, $females\_produce\_milk$. This similarity motivates us to build a model that trains representation vectors to capture these two types of information simultaneously.

Contributions In this paper, we propose a novel model to train knowledge representations to capture attributive and hyponymous relationships. In this model attributes and hyponyms are represented as multivariate Gaussians, while entities are represented as low dimensional vectors. In training, the model iteratively updates the vectors and covariance matrices so as to ensure that the probability of an entity vector in its attribute's or hyponym's Gaussian distribution is maximised. Furthermore, each update takes into consideration its estimated global influence, which can effectively prevent the forgetting of previously learnt knowledge. The model is evaluated on the task of triple classification and achieves excellent performance.

By proposing this model, we demonstrate the effectiveness of a novel approach in knowledge representation learning: directly representing entities as rich aggregation of attributes. This approach contrasts with the traditional approach used by previous models, which focuses on extracting attributive information from entity vectors[7, 8, 10, 9]. Our approach has a

unique advantage over the traditional approach — its transparency: the trained presentations (embeddings) form a meaningful, interpretable space that can be visualised and understood. Further, it can conduct inductive and deductive reasoning in a manner similar to reasoning with Venn diagrams (elaborated in section **5.3**), which may explain its good performance in triple classification tasks.

2 Related Work

Diverse models are proposed to train knowledge representation vectors. The models can be roughly divided into two types: neural models and translation models. In the following paragraphs, a knowledge triple is defined as (h, r, t) where r is the relation, and h and t are head and tail entities respectively.

Neural Models Neural models [4, 11, 12] treat triplet classification tasks as typical binary classification tasks. While different models have varied score functions, activation functions, and network architecture, they essentially leverage the idea that representation vectors are network parameters that need to be updated using back propagation to improve classification accuracy. The score functions of different neural models are shown in Table 2.

Table 2: Score Functions of Some Neural Models

Model	Score Function
Single Layer Model	$f_r(h,t) = \mathbf{u}_r^\intercal \tanh(\mathbf{M}_{r,1}\mathbf{h} + \mathbf{M}_{r,2}\mathbf{t})$
Multi Layer Perceptron	$f_r(h,t) = \mathbf{u}_r^\intercal \tanh(\mathbf{M}_1\mathbf{h} + \mathbf{M}_2\mathbf{r} + \mathbf{M}_3\mathbf{t})$
Neural Tensor Network	$f_r(h,t) = \mathbf{u}_r^\intercal \tanh(\mathbf{h}^\intercal\mathbf{M}_r\mathbf{t} + \mathbf{M}_{r,1}\mathbf{h} + \mathbf{M}_{r,2}\mathbf{t} + \mathbf{b}_r)$

In the multi layer perceptron model, all head or tail vectors are transformed by the same weight matrix $(\mathbf{M}_1, \mathbf{M}_2)$ regardless of how they are related. The single layer model takes the relation into consideration by letting each relation has its own head and tail matrices $(\mathbf{M}_{r,1}, \mathbf{M}_{r,2})$. Neural Tensor Network improves upon the previous models by incorporating the interaction between head and tail entities $(\mathbf{h}^\intercal\mathbf{M}_r\mathbf{t})$.

Translation Models The central idea underlying translation models [7, 8, 9, 13] is that relations can be encoded into the translation differences between entity vectors. This idea is inspired by the observation that translation differences between word embeddings often contain relational information about words. For instance, the difference between embeddings of *Barcelona* and *Spain* is similar to the difference between embeddings of *Paris* and *France*. This difference encodes the relation $capital\_of$. Tapping on this idea, researchers proposed various models. The scoring functions of the models are shown in Table 3 below.

Table 3: Score Functions of Some Translation Models

Model	Score Function
TransE	$f_r(h,t) = \|\mathbf{h} + \mathbf{r} - \mathbf{t}\|_{L_1/L_2}$
TransH	$\mathbf{h}_r = \mathbf{h} - \mathbf{w}_r^\intercal \mathbf{h} \mathbf{w}_r, \quad \mathbf{t}_r = \mathbf{t} - \mathbf{w}_r^\intercal \mathbf{t} \mathbf{w}_r, \quad f_r(h,t) = \|\mathbf{h}_r + \mathbf{r} - \mathbf{t}_r\|_{L_1/L_2}$
TransR	$\mathbf{h}_r = \mathbf{h}\mathbf{M}_r, \quad \mathbf{t}_r = \mathbf{t}\mathbf{M}_r, \quad f_r(h,t) = \|\mathbf{h}_r + \mathbf{r} - \mathbf{t}_r\|_{L_1/L_2}$

TransE model proposed in [7] aims to ensure that the translation difference between two embeddings is the vector embedding of their relation; that is, $\mathbf{h} - \mathbf{t} = \mathbf{r}$. However, it is found that TransE are unable to model complex relations as it lacks flexibility. TransH and TransR [10, 8] improve on TransE by letting an entity have different distributed representations in triples with different relations. TransH does so by projecting entities onto a hyperplane specific to a particular relation, while TransR accomplishes this by projecting entities from entity space to r-relation space.

Both neural models and translation models have the same training methods and score functions for all types of relations, leading to varied performance in tasks specific to a particular type of relation[14]. This indicates that there is need to build models that specifically tackle each type of relations.

Relation-Specific Model In [14], the authors proposed a new model (KR-EAR) that has different training methods and score functions for attributive relations and interrelational relations. The model adopts TransE and TransR for the learning of interrelations and a neural model for the learning of attributes. The author defines an attribute triple as (e, a, v), where e is the entity, a is the attribute, and v is the value of the attribute. A sample triple would be $(lemon, taste, sour)$. The authors train a multi-class classification model to predict attribute value v given e, a. The key insight here is that we can regard attribute values as classes, entities as instances. Interrelations relate instances to instances, while attributes relate instances to classes. In fact, the actual classes are not attribute values. *American* itself is not a class, but (*nationality*, *American*), (*produced_in*, *American*) are. Therefore, It is an **attribute-value** pair that defines a class.

Classes are more general concepts than instances. It is intuitive that uncertainties should be allowed for classes.

Gaussian Embeddings In [15], the authors take into consideration the uncertainties and generality of entities and relations in knowledge graph, and propose a model (KG2E) in which entities and relations are represented as Gaussian embeddings. Recall a triplet as (h, r, t). The score function in this model is the similarity (measured by KL-divergence) between the relation distribution $\mathcal{P}_r \sim \mathcal{N}(\mu_r, \Sigma_r)$ and the difference in distributions of head and tail entities $\mathcal{P}_d \sim \mathcal{N}(\mu_h - \mu_t, \Sigma_h + \Sigma_t)$. However, the model in essence is still a translation model, the main difference is that it now forces distributions rather than vectors to be similar. Hence, it is still not flexible enough to model the generality of attributes.

3 Problem Formulation

In this paper, we define each knowledge triple stored in the knowledge graph as (h, r, t), where h and t are head and tail entities respectively, and r is the relation between the two. Further, for the triples that contain attributive or hyponymous relations, we define **entity-attribute pair** to be (e, a), where $e = h$, $a = (r, t)$. We group r and t together because, as mentioned in Section **2**, an attribute-value pair (r, t) defines a class. Hence, a, the attribute or hyponym, is the class, and e is its instance. For convenience sake, we will refer to e as entities, and a as attributes.

Our task is to learn knowledge representations, in the form of Gaussian embeddings, for entity and attributes based on existing knowledge graphs. The learnt representations should capture the attributive and hyponymous relationships between entities. Additionally, they should be able to perform triple classification tasks well so they can readily be used for automatic knowledge graph completion tasks.

4 Methodology

In this section, we will describe the structure of our model.

In the following sections, we define the number of attributes to be M, the number of entities to be N, the dimension of the embeddings to be k. An entity's embedding is defined as a k-dimensional vector e. An attribute's embedding is defined as a Gaussian embedding $\mathcal{P} \sim \mathcal{N}(\mu, \Sigma)$, where μ is the mean vector and Σ is the covariance matrix. We set Σ to be a diagonal matrix to limit the number of parameters and reduce time complexity. Let $\sigma_q = (\Sigma_{q,q})^{\frac{1}{2}}, q \in \{1, 2, ..., k\}$.

Briefly speaking, the model iteratively updates means and covariance matrices of the Gaussian embeddings of entities and attributes to maximise the relative probability of an entity in the Gaussian distribution of its attribute and minimise the relative probability of an entity in the Gaussian distribution of an unrelated attribute. Here, relative probability p_r of a point x in a distribution $\mathcal{P} \sim \mathcal{N}(\mu, \Sigma)$ is defined as:

$$p_r(x; \mu, \Sigma) = \frac{p(x; \mu, \Sigma)}{\frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}}} \tag{1}$$

where the denominator is the peak value, or maximum probability, in the distribution.

4.1 Initialization

The model firstly initialises the Gaussian embeddings of all the attributes. We want the attribute embeddings to be initialised in such a way so that two attributes sharing many similar entities will be close together in the vector space, for in this way the shared entities' vectors can have high probabilities in both distributions.

To achieve this aim, we initialise the mean vectors, μ, of attribute embeddings as N-dimensional binary vectors, $\mu_{j,i} = 1$ if entity i has attribute j; otherwise, $\mu_{j,i} = 0$. The binary vectors are then reduced to k-dimensional vectors through principal component analysis (PCA). We explore the properties of the space after dimensionality reduction and confirm our hypothesis that attributes sharing common entities are closer together in the vector space. However, we find that our hypothesis mainly applies to the attributes whose original binary vectors are not very sparse.

4.2 Updating

In each iteration, a positive (i.e. contain a true relation) or negative (i.e. contain a false relation) entity-attribute pair is passed into the model. Let the pair be (e, a), where $a = \mathcal{P} \sim \mathcal{N}(\mu, \Sigma)$. Define

$$L = \sum_{q=1}^{k} (max(|e_q - \mu_q|) - c\sigma_q, 0)) \tag{2}$$

where c is a hyper-parameter to be determined. Vector L estimates the distance between e_q and an ellipsoid surrounding μ. The ellipsoid is the **isocontours**, or the **level curves** of $\mathcal{P}$: all points on the ellipsoid have the same relative probability, f, in the distribution $\mathcal{P}$. f is can be determined by c. Any point inside the ellipsoid have a relative probability larger than f. Each entry in the vector $\sum_{q=1}^{k} c\sigma_q$ is the half the length of an axis of the ellipsoid.[16]

Define the learning rate to be s. The embeddings are updated as follow.

if the pair (e, a) is positive:

$$e = e + 0.5sL(\mu - e) \tag{3}$$

$$\mu = \mu + 0.5sL(e - \mu) \tag{4}$$

$$\sigma_q = \sigma_q + sL \tag{5}$$

if the pair (e, a) is negative:

$$e = e - 0.5sL(\mu - e) \tag{6}$$

$$\mu = \mu - 0.5sL(e - \mu) \tag{7}$$

$$\sigma_q = \sigma_q - sL \tag{8}$$

Interpretation After an update based on a positive pair, the entity vector e and the mean vector μ will move closer towards each other to maximise the relative probability (p_r) of the entity vector in the attribute's distribution. The axis of the isocontour ellipsoid will also extend towards the direction of the entity vector. An update based on a negative pair has the exactly opposite effect. Note that if the $p_r > f$, L will be zero and no updates will happen, which is reasonable as this means p_r is already sufficiently high.

4.3 Estimation of Global Influence

Forgetting Effect After running some tests, we notice that if we simply let the model updates the embeddings according to section **4.2**, the embeddings may easily "forget" relations that are learnt previously. In particular, entity embeddings seem to be moving for a large distance after each update, rendering previous updates useless. To prevent this forgetting effect, before each update, the model will estimate the global influence of the update and use this information to constrain the update. More specifically, the model estimates how moving to a particular position will affect the entity vector's relative probabilities p_r in other attribute distributions by calculating $\frac{\partial p_r}{\partial d}$, where $d = e - \mu$ and e is the updated vector, for n distributions sampled from the distributions of attributes that have already been learned.

For the distribution of attribute j, $\mathcal{P}_j \sim \mathcal{N}(\mu_j, \Sigma_j)$ where Σ_j is diagonal,

$$p_r(e; \mu_j, \Sigma_j) = \exp(-\frac{1}{2}(e - \mu_j)^\intercal \Sigma (e - \mu_j) = \exp(-\frac{1}{2} d^\intercal \Sigma d) \tag{9}$$

$$\frac{\partial p_{rj}}{\partial d} = \sum_{q=1}^{k} \frac{\partial p_{rj}}{\partial d_q} = \sum_{q=1}^{k} (-\frac{d_q}{(\Sigma_{q,q})^2} p_{rj}) \tag{10}$$

The estimated global influence of the update, then, can be encapsulated in a vector g, where

$$g = \sum_{j=1}^{n} (\frac{\partial p_{rj}}{\partial d}) \tag{11}$$

We then normalise g by dividing it by $max(|g_1|, |g_2|, \ldots, |g_k|)$, where g_i is the value of dimension i in g. Lastly, we set $g_i = |min(g_i, 0)|$ for $i \in \{1, 2, \ldots, k\}$.

The value of each dimension of g, ranging from 1 to 0 denotes the degree to which this update will negatively influence the relative probabilities of the entity in other distributions. The larger the value is, the more drastic the relative probabilities will decrease due to update in that dimension. We want to constrain the update of the dimensions that may have negative global influence, so the adjusted update rules for entity embeddings are:

if the pair (e, a) is positive:

$$e = e + 0.5sL(1 - g)(\mu - e) \tag{12}$$

if the pair (e, a) is negative:

$$e = e - 0.5sL(1 - g)(\mu - e) \tag{13}$$

During training, in each iteration we pass into the model a positive or negative entity-attribute pair and then update the embeddings based on rules outlined in equations (4), (5), (7), (8), (12) and (13).

5 Results and Discussions

In this section, we will firstly describe the task that our model is tested on – triple classification. We will then introduce the datasets and training details. Lastly, we will analyse the experimental results and compare our model's performance to other models'.

5.1 Triple Classification

Triple classification asks models to predict the truthfulness of unseen relation triples using existing facts in knowledge graphs.

This task can be seen as a sub-task of automatic knowledge graph completion [4]. By extracting from raw text or automatically generating potentially true triples and evaluating their truthfulness using models trained on triple classification task, knowledge graphs can be automatically extended.

5.1.1 Datasets

We train our models on dataset FB24k, a dataset that is based on Freebase and separates attributive and interrelational relations [14]. The statistics of the dataset can be found in Table 4.

Table 4: Statistics of the Dataset

Dataset	**FB24k**
#Entities	23634
#Attributes	314
#Relation Triples in Training Dataset	196,850
#Relation Triples in Testing Dataset	10,301

As the dataset only contains positive triples, we constructed an equal number of negative triples. We observe it is more likely for attributes that appear in many triples (i.e. frequently mentioned attributes) to classify a triple as true. This is possibly because the distributions of frequently mentioned attributes are broader and more encompassing so that more entity vectors can have a high probability in it. Therefore, to enhance training and ensure the rigor of testing, we set the number of negative triples for a particular attribute as the same as the number of positive triples for that attribute, so that frequently appeared attributes would not simply give positive output for all entities. We then transform the triples, which are in the form of (h, r, t), into entity-attribute pairs of the form (e, a), where $e = h$, $a = (r, t)$, and use them for training and testing.

5.1.2 Task-specific Details

In our model, embeddings are trained to maximise the relative probability of an entity in the Gaussian distribution of its attributes. To adapt our models to triple classification task, we need to find the threshold for relative probability, θ_{p_r}, such that if $p_r(e; \mu_a, \Sigma_a) > \theta_{p_r}$, then the pair (e, a) is highly likely to be a positive pair. We use the training dataset to find the optimal θ_{p_r}.

Our model contains three other hyper-parameters: 1. learning rate, s, 2. dimension of embeddings, k, 3. and the arbitrary constant in equation (2), c. We search for the optimal s, k, and c using grid search. s is chosen from $\{0.1, 0.2, \ldots, 1.0\}$, k is chosen from $\{10, 15, 20, 25, 30\}$, c is chosen from $\{1, \sqrt{2}, 2, 3\}$. The optimal choice of hyper-parameters is $s = 0.3$, $k = 25$, and $c = \sqrt{2}$.

5.1.3 Results

The accuracy obtained by different models on triple classification task is shown in Table 5.

Table 5: Evaluation Results on Triple Classification

Model	Accuracy
NTN	68.5
TransE	79.5
TransH	80.2
TransR	83.9
TranSparse	84.2
ComplEx	87.2
Gaussian (ours)	**87.1**

From the results table, we can observe that our model achieves state-of-art performance in triple classification task, with an accuracy of 87.1%. It outperforms almost all the other models significantly. It is worth noting that as our model was trained on a commercial laptop with limited memory, computational power and processing speed, we have to limit the number of iterations and the number of distributions, n, that we can sample when estimating global influence (section **4.3**). Therefore, 87.1% may not be the best result this model is able to achieve.

5.2 Properties of Learnt Embeddings

We explore the properties of trained embeddings by

1. measuring the relative distance between means of attribute distributions to determine their relative positions
2. measuring the axis lengths of isocontours of attribute distributions to determine the spread of the distributions.

Property 1 We discover a special property about **attributes that are similar in types but distinct in terms of entities they describe**. The first requirement, "similar in types", can also be understood as two $a_i = (r_i, t_i)$ and $a_j = (r_j, t_j)$, where $r_i = r_j$. The second requirement, "Distinct in entities described" means that the set of entities with attribute a_i and the set of entities with attribute a_j are non-overlapping. $(gender, female)$ and $(gender, male)$ are such a pair of attributes, as a person cannot be both female and male. Note that the satisfaction of the first requirement does not entail the satisfaction of the second: a professor may also be a scientist. For attribute groups that satisfy both requirements, we notice that the Euclidean distance between them is generally large, especially when the attributes are frequently mentioned. This property undoubtedly enhances the accuracy of triple classification as entities would be classified as either only having one of these attributes or having none of these attributes at all, creating less contradictions. For instance, an entity would either be classified as female or male, or not describable by both attributes (i.e. not an animal); It is highly unlikely that an entity will be classified as both male and female (due to the large separation between "male" and "female").

Property 2 Less frequently mentioned attributes of the similar types and themes usually form a local cluster. For instance, the attributes $(profession, politicalscientist)$, $(profession$, $,businessman)$, and $(profession, philosophers)$ are closer to each other, with an average distance of 1.1 between pairs. As a comparison, the distance between $(gender, female)$ and $(gender, male)$ is 41.7. This may be because:

1. the less frequently appeared attributes are updated for fewer times, so they are still "in the corner" of the vector space.
2. entities that have these attributes usually are similar in terms of other attributes they possess (come from the same background). For instance, most businessmen, philosophers and political scientists (in FB24K dataset) have college degrees, are from USA, and are male. The overlapping between entities they describe cause these attributes to be close together.

The presence of Property 2, though expected, is a source of inaccuracy for our model. For instance, the model finds it hard to determine whether someone is a philosopher or political scientist when it only knows a person's degree, nationality, and some other general information, as shown by the relatively low classification accuracy of 61.2%. This is because the distributions are all situated at the same corner of the distributions of general attributes such as "degree" and "nationality", as shown in Figure 1 below, so the probabilities of this person's representation vector in any of these distributions are very similar. However, this difficulty is expected, as even humans cannot give the correct answer if there is insufficient information to reason with. To better evaluate our model in predicting infrequently mentioned attributes, we would need a larger and much more detailed dataset, which is currently unavailable.

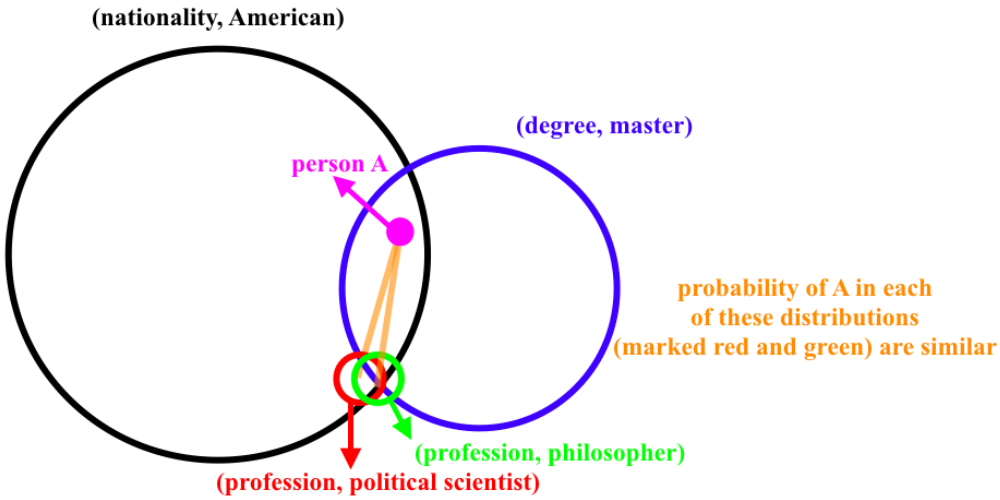

Figure 1: An Illustration of Property 2

Property 3 Gaussian distributions of frequently-mentioned attributes have wider spread than those of less frequently mentioned attributes. This is because:

1. More frequently mentioned attributes are updated for more times, thus the axis is likely to be extended longer.
2. More frequently mentioned attributes need to encompass more entities.

This property can both improve and deteriorate the performance of our model. It can improve the model because frequently-mentioned attributes are supposed to be possessed by many entities, so it should encompass a wide range in the vector space. However, this also means it can encompass many entities that do not have that attribute. We can address the second issue by **giving attribute embeddings more flexibility**, so they can "carve out" a more refined space. This is why k, the dimension of embedding vectors, have to be higher than a certain value. The lower the dimension, the less flexible the attribute embeddings, and it will encompass more false entities. In one-dimentional space, for instance, the attribute's distribution may extend to the entire range on the line where entity vectors (scalars in 1D) exist. Another way to address the second issue is to use non-diagonal covariance matrices for Gaussian embeddings to increase embeddings' flexibility, although it would increase the number of parameters by a factor of k.

5.3 Gaussian Embeddings in Reasoning

The model proposed in this paper can is essentially training a complex Venn diagram in low-dimensional vector space. Given the simple nature of this model, it is perhaps hard to understand why it can achieve such excellent performance. The answer to this question is two-fold.

The Nature of Relations There is a key difference between attributive and hyponymous relations and inter-relational relations. Entities are defined by attributes and hyponyms: an

apple is a fruit that grows on apple trees and is usually edible; but not inter-relations: an apple is not defined as something that was first ate by person X. Therefore, entities can be viewed as a rich collection of attributes. This idea can be translated to a Venn diagram (see Fig 2 below) in which entities locate at the intersections of attributes. Therefore, our model, by training a "complex and probabilistic Venn Diagram", adequately encapsulate the relationships between attributes and entities.

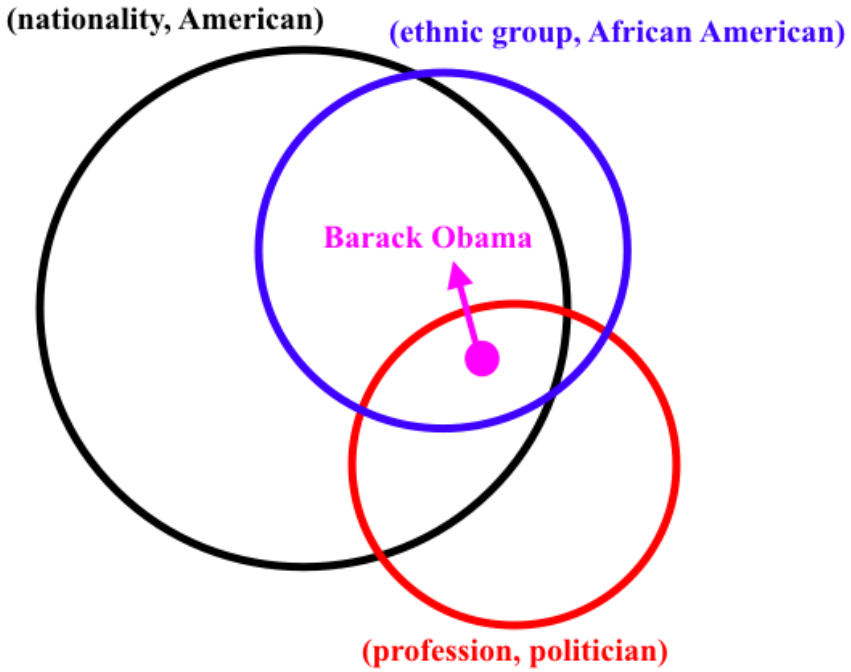

Figure 2: A Venn Diagram Representing Attributes, Entities, and their Relations

The Nature of Reasoning As pointed out by [4], this task is in fact testing model's ability to conduct common sense reasoning, including deduction and induction. An example of deduction is:

Premise 1 we know ants are a type of insects
Premise 2 we know all insects have six legs
Conclusion therefore we know ants should also have six legs.

Deductions can be easily conducted with Venn diagrams and our embeddings. If Premise 1 and Premise 2 are provided to our model as positive entity-relation pairs, the distributions trained on these pairs can be represented using a Venn diagram (see Fig 3 below). We can immediately see the "ants" will have the attribute "have six legs" as well.

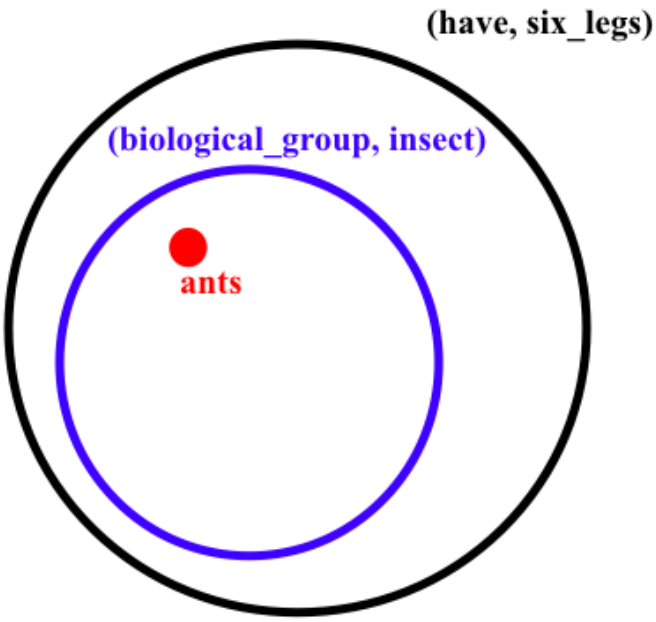

Figure 3: An Illustration of Deductive Reasoning in Gaussian Embeddings

An example of induction is:

Premise 1 we know a person A's religion is Buddhism

Premise 2 we know most Buddhists are Asians

Conclusion therefore we know that A is more likely to be an Asian rather than an African or a Caucasian

Premise 2, being a probabilistic statement, cannot be an entity-relation pair. Such probabilistic statements are usually learned by the model implicitly. For instance, most of the entities that are in positive relational pairs of the form $(entity, buddhist)$ are also in positive relational pairs of the form $(entity, Asian)$ rather than $(entity, Caucasian)$ or $(entity, African)$. When Premise 2 is learned , the attribute embeddings may distribute as shown in Fig 4 below. When Premise 1 is given, it is more likely that the vector that represents person A is located within the intersection of "Asian" and "Buddhist", which aligns with the inductive conclusion.

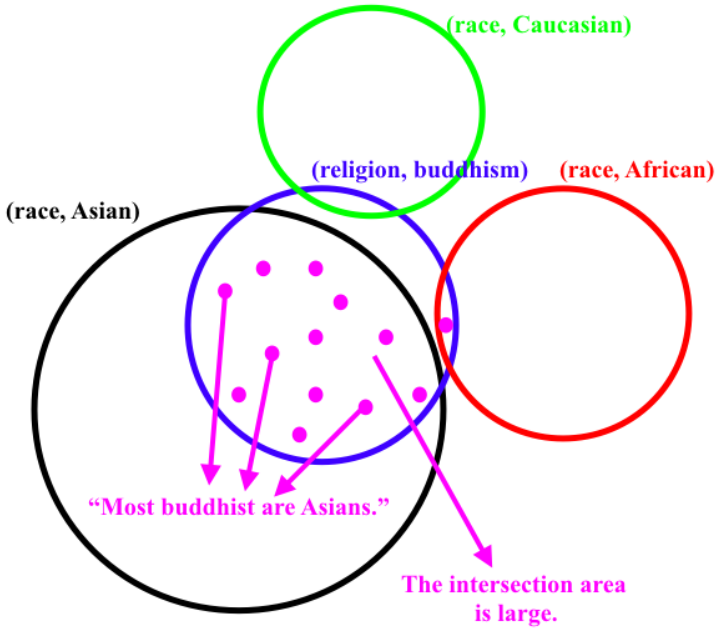

Figure 4: An Illustration of Inductive Reasoning in Gaussian Embeddings

During training, the model learns the premises for deduction and induction from the entity-attribute pairs passed into it. It performs deductive and inductive reasoning implicitly, and thus is able to predict the truthfulness of an unseen pair accurately based on existing knowledge.

6 Conclusion

In this paper, we introduce a model that trains Gaussian embeddings for entities and attributes in knowledge graphs. The model achieves good performance in triple classification tasks, indicating its potential in automatic knowledge graph completion. We explore the properties of the trained embeddings and elucidate some areas of improvement. Lastly, we explain the model's performance in triple classification tasks by analysing how it is able to conduct reasoning using trained embeddings.

It is worth noting that this paper by no means suggests that Gaussians are the most appropriate form for knowledge representations; instead, it stresses the novel idea of considering entities as rich aggregations of attributes and directly using attributes to represent entities, which can significantly increase the interpretability and transparency of the model, making it easier to understand and control.

Future work will focus on how to increase the flexibility of the model so that the model can maintain or even improve upon its current performance when the number of attributes and entities increases. Possible ways to increase the flexibility of the model includes: using non-diagonal covariance matrices for Gaussian embeddings, increase the dimension of embeddings,

using other distributive representations instead of Gaussians. Another possible path for future work is to explore how different types of reasoning can be formalised in vector space.

References

[1] Y. Lin, X. Han, R. Xie, Z. Liu, and M. Sun, "Knowledge Representation Learning: A Quantitative Review," *arXiv e-prints*, p. arXiv:1812.10901, Dec 2018.

[2] L. Xu, Q. Zhou, K. Gong, X. Liang, J. Tang, and L. Lin, "End-to-End Knowledge-Routed Relational Dialogue System for Automatic Diagnosis," *arXiv e-prints*, p. arXiv:1901.10623, Jan 2019.

[3] Y. Cao, X. Wang, X. He, Z. hu, and T.-S. Chua, "Unifying Knowledge Graph Learning and Recommendation: Towards a Better Understanding of User Preferences," *arXiv e-prints*, p. arXiv:1902.06236, Feb 2019.

[4] R. Socher, D. Chen, C. D. Manning, and A. Ng, "Reasoning with neural tensor networks for knowledge base completion," in *Advances in Neural Information Processing Systems 26* (C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, eds.), pp. 926–934, Curran Associates, Inc., 2013.

[5] A. Fader, S. Soderland, and O. Etzioni, "Identifying relations for open information extraction," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, (Stroudsburg, PA, USA), pp. 1535–1545, Association for Computational Linguistics, 2011.

[6] R. Snow, D. Jurafsky, and A. Y. Ng, "Learning syntactic patterns for automatic hypernym discovery," in *Advances in Neural Information Processing Systems 17* (L. K. Saul, Y. Weiss, and L. Bottou, eds.), pp. 1297–1304, MIT Press, 2005.

[7] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Advances in Neural Information Processing Systems 26* (C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, eds.), pp. 2787–2795, Curran Associates, Inc., 2013.

[8] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," in *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, pp. 1112–1119, AAAI Press, 2014.

[9] G. Ji, K. Liu, S. He, and J. Zhao, "Knowledge graph completion with adaptive sparse transfer matrix," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pp. 985–991, AAAI Press, 2016.

[10] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," 2015.

[11] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang, "Knowledge vault: A web-scale approach to probabilistic knowledge fusion," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, (New York, NY, USA), pp. 601–610, ACM, 2014.

[12] Q. Liu, H. Jiang, A. Evdokimov, Z.-H. Ling, X. Zhu, S. Wei, and Y. Hu, "Probabilistic Reasoning via Deep Learning: Neural Association Models," *arXiv e-prints*, p. arXiv:1603.07704, Mar 2016.

[13] T. Trouillon, J. Welbl, S. Riedel, E. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," in *Proceedings of The 33rd International Conference on Machine Learning* (M. F. Balcan and K. Q. Weinberger, eds.), vol. 48 of *Proceedings of Machine Learning Research*, (New York, New York, USA), pp. 2071–2080, PMLR, 20–22 Jun 2016.

[14] Y. Lin, Z. Liu, and M. Sun, "Knowledge representation learning with entities, attributes and relations," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pp. 2866–2872, AAAI Press, 2016.

[15] S. He, K. Liu, G. Ji, and J. Zhao, "Learning to represent knowledge graphs with gaussian embedding," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, (New York, NY, USA), pp. 623–632, ACM, 2015.

[16] Chuong B. Do, "The Multivariate Gaussian Embeddings", 2008.