SimplE Embedding for Link Prediction in Knowledge Graphs

Seyed Mehran Kazemi [*1] **David Poole** [*1]

Abstract

The aim of knowledge graphs is to gather knowledge about the world and provide a structured representation of this knowledge. Current knowledge graphs are far from complete. To address the incompleteness of the knowledge graphs, *link prediction* approaches have been developed which make probabilistic predictions about new links in a knowledge graph given the existing links. *Tensor factorization* approaches have proven promising for such link prediction problems. In this paper, we develop a simple tensor factorization model called *SimplE*, through a slight modification of the *Polyadic Decomposition* model from 1927. The complexity of SimplE grows linearly with the size of embeddings. The embeddings learned through *SimplE* are interpretable, and certain types of expert knowledge in terms of logical rules can be incorporated into these embeddings through weight tying. We prove *SimplE* is fully-expressive and derive a bound on the size of its embeddings for full expressivity. We show empirically that, despite its simplicity, *SimplE* outperforms several state-of-the-art tensor factorization techniques.

1. Introduction

During the past two decades, several knowledge graphs (KGs) containing (perhaps probabilistic) facts about the world have been constructed (Carlson et al., 2010; Fader et al., 2011; Wu et al., 2012; Dong et al., 2014; Shin et al., 2015). These KGs have applications in several fields including natural language processing, search, automatic question answering, recommendation systems, etc.

Due to the enormous number of facts that could be asserted about our world and the difficulty in accessing and storing all these facts, KGs are incomplete. However, it is possible to predict new links in a KG based on the existing ones. *Link prediction* and several other related problems aiming at reasoning with entities and relationships are studied under the umbrella of *statistical relational learning (SRL)* (Getoor & Taskar, 2007; De Raedt et al., 2016). The problem of link prediction for KGs is also known as *knowledge graph completion*. A KG can be represented as a set of $(head, relation, tail)$ triples[1]. The problem of KG completion can be viewed as predicting new triples based on the existing ones.

Three of the main categories of SRL approaches to link prediction include weighted rule learning, graph random walk, and tensor factorization. Weighted rule models (De Raedt et al., 2007; Domingos et al., 2008; Kimmig et al., 2012; Kazemi et al., 2014) typically learn first-order logic rules that define the structure and the regularities in the data, and a weight for each rule representing the confidence of the model in that rule. Graph random walk approaches (Lao & Cohen, 2010; Lao et al., 2011; Wang et al., 2016; Das et al., 2017) typically define a strategy for starting from a node in the graph, (probabilistically) walking on the graph and reaching a desired node. Tensor factorization approaches (Nickel et al., 2012; Bordes et al., 2013b; Trouillon et al., 2016; Nguyen et al., 2016) learn embeddings for entities and relationships and define a function from the embeddings to whether a relation exists between a head and a tail entity or not.

Each of the three SRL categories mentioned above has its own advantages and disadvantages. Several recent surveys and comparative studies describe and compare the approaches in one or more of these categories on several tasks (Kimmig et al., 2015; Nickel et al., 2016a; Kazemi et al., 2017). It has been observed in several works that hybrid models (models based on a combination of the approaches in these categories) may result in higher predictive performance compared to models based on approaches in a single category (Lin et al., 2015a; Neelakantan et al., 2015; Demeester et al., 2016; Niepert, 2016). However, these hybrid approaches generally rely on powerful individual predictors from each category. Thus, developing better models in each category results in better hybrid models.

In this paper, we study the third category, namely *tensor*

[1]Department of Computer Science, University of British Columbia, Vancouver, Canada. Correspondence to: Seyed Mehran Kazemi <smkazemi@cs.ubc.ca>.

[1]Triples are complete for relations. They are sometimes written as $(subject, verb, object)$ or $(individual, property, value)$.

factorization approaches. We categorize these approaches into three main classes: *translational*, *deep learning*, and *multiplicative*. These approaches consider embeddings for each entity and each relation. To predict whether a relation holds between a head and a tail entity, they use a function which takes the embeddings for the two entities and the relation as input and outputs a number indicating the predicted probability. Translational approaches (Bordes et al., 2013b; Ji et al., 2015; Wang et al., 2014; Nguyen et al., 2016) apply an additive function over the embeddings. The function in deep learning approaches (Socher et al., 2013; Dong et al., 2014; Santoro et al., 2017) is typically a (deep) neural network. Multiplicative approaches (Hitchcock, 1927; Nickel et al., 2012; Yang et al., 2015; Nickel et al., 2016b; Trouillon et al., 2016) apply a product-based function over the input embeddings. Details and discussions of these approaches can be found in several recent surveys (Nguyen, 2017; Wang et al., 2017).

One of the first multiplicative approaches is the *polyadic decomposition* of Hitchcock (1927). This approach learns one embedding vector for each relation and two embedding vectors for each entity, one to be used when the entity is the *head* and one to be used when the entity is the *tail*. The two embedding vectors for entities are learned independently: observing some entity participates in a relation as the head does not affect the vector corresponding to the case where the entity is the tail and vice versa. This independence assumption has caused polyadic decomposition to perform poorly for KG completion (Trouillon et al., 2017). In this paper, we develop a multiplicative tensor factorization approach based on polyadic decomposition that addresses the independence among the two embedding vectors of the entities. Due to the simplicity of our model, we call it *SimplE*.

We show that *SimplE*: 1- learns interpretable embeddings, 2- is fully-expressive, 3- is capable of encoding expert knowledge into its embeddings through parameter sharing (aka weight tying), and 4- performs very well empirically despite its simplicity. We also discuss several disadvantages of other existing approaches. For instance, we prove that the existing translational approaches are not fully-expressive and we identify severe restrictions on what they can represent. We also show that the function used in *ComplEx* (Trouillon et al., 2016; 2017), one of state-of-the-art approaches, involves redundant computations.

2. Background and Notation

This section introduces our notation and provides the necessary background for readers to follow the rest of the paper.

Let $\mathcal{E}$ represent the set of entities and $\mathcal{R}$ represent the set of relations. A *triple* is represented as (h, r, t), where $h \in \mathcal{E}$ in the *head*, $r \in \mathcal{R}$ is the relation, and $t \in \mathcal{E}$ is the *tail* of the triple. Let ζ represent the set of all triples that are true in a world, and ζ' represent the ones that are false. An example of a triple in ζ can be $(Paris, isCapitalOf, France)$ and an example of a triple in ζ' can be $(Paris, isCapitalOf, Germany)$. A *knowledge graph* $\mathcal{KG}$ is a subset of ζ.

A relation r is *reflexive* on a set $\mathcal{E}$ of entities if $(e, r, e) \in \zeta$ for any entity $e \in \mathcal{E}$. A relation r is *symmetric* on a set $\mathcal{E}$ of entities if $(e_1, r, e_2) \in \zeta \iff (e_2, r, e_1) \in \zeta$ for any two entities $e_1, e_2 \in \mathcal{E}$, and is *anti-symmetric* if $(e_1, r, e_2) \in \zeta \iff (e_2, r, e_1) \in \zeta'$. A relation r is *transitive* on a set $\mathcal{E}$ of entities if $(e_1, r, e_2) \in \zeta \wedge (e_2, r, e_3) \in \zeta \Rightarrow (e_1, r, e_3) \in \zeta$ for any three entities $e_1, e_2, e_3 \in \mathcal{E}$.

An *embedding* is a function from an entity or a relation to one or more vectors or matrices of numbers. We represent the embedding associated with an entity e by $\mathbf{E}(e)$ and the embedding associated with a relation r with $\mathbf{E}(r)$.

A *tensor factorization* model defines two things: 1- the embedding functions for entities and relations, 2- a function f taking $\mathbf{E}(h)$, $\mathbf{E}(r)$ and $\mathbf{E}(t)$ as input, and generating a prediction of whether (h, r, t) is in ζ or not. The values of the embeddings are learned using the triples in a $\mathcal{KG}$.

A tensor factorization model is *fully-expressive* if given any ground truth (full assignment of truth values to all triples), there exists an assignment of values to the embeddings of the entities and relations that accurately separates the correct triples from incorrect ones.

Let v, w and x be vectors of length k. We define $\langle v, w, x \rangle$ to be the sum of the element-wise product of the three vectors. That is, $\langle x, w, x \rangle = \sum_{j=1}^{k} v_j * w_j * x_j$, where v_j, w_j, and x_j represent the jth element of v, w and x respectively.

3. Existing Tensor Factorization Approaches

We describe three general categories of tensor factorization approaches and some selected models in each category.

3.1. Translational Approaches

Translational approaches define additive functions over embeddings. TransE (Bordes et al., 2013b) is one of the first such approaches. In TransE, the embedding for both entities and relations is a vector of size k. Let $v(e)$ and $v(r)$ represent the embedding vectors of an entity e and a relation r respectively. The dissimilarity function used in TransE for a triple (h, r, t) is $||v(h)+v(r)-v(t)||_i$, where $||v||_i$ represents norm i of vector v (i is typically 1 or 2). The smaller the value of the norm, the more probable the triple (h, r, t) being correct. An alternative way of looking at TransE is that it aims at learning its embedding vectors in a way that $v(h) + v(r) \approx v(t)$ for any triple (h, r, t) in the KG.

Realizing that TransE is quite limited in what it can model, several works have extended the idea in TransE and made it more flexible. Most of these works consider extra vectors/matrices in the embeddings of the relations, project the entity embeddings into relation-specific spaces using the extra vectors/matrices, and then apply TransE. As an example, *STransE* (Nguyen et al., 2016) considers $\mathbf{E}(r)$ to be a vector $v(r)$ plus two matrices $M_h(r)$ and $M_t(r)$. They define their dissimilarity function as $||M_h(r)v(h)+v(r)-M_t(r)v(t)||_i$. That is, they use the matrices to project entity vectors into a new space (projecting head and tail using different matrices), and then apply TransE. Attempts have been also made to relax the dissimilarity function. Instead of enforcing $v(h)+v(r)\approx v(t)$ as in TransE, Feng et al. (2016)'s *FTransE* enforces $v(h)+v(r)\approx \alpha v(t)$ for some real number α. According to FTransE, only the direction of the $v(h)+v(r)$ matters, not its size. Note that FTransE and STransE (or other related works) can be combined to create the function $M_h(r)v(h)+v(r)\approx \alpha M_t(r)v(t)$. To our knowledge, this combination is currently one of the most flexible translational models. In the rest of the paper, we call this model *FSTransE*.

3.2. Deep Learning Approaches

Deep learning approaches generally use neural networks to learn how the embeddings interact. E-MLP (Socher et al., 2013) considers the embeddings for entities to be vectors v of size k, and for relations to be a matrix M of size $2k*m$ and a vector v of size m. To make a prediction about a triple (h,r,t), E-MLP concatenates the vectors of the entities and create the vector $[v(h);v(t)]$, and feeds it into a two-layer neural network whose weights for the first layer are the matrix $M(r)$ and for the second layer are $v(r)$. E-MLP requires $(2k+1)*m$ parameters for each relation. ER-MLP (Dong et al., 2014), the approach used in Google Knowledge Vault, considers the embeddings for both entities and relations to be single vectors. To make a prediction about a triple (h,r,t), they concatenate the vectors of the entities and the relation creating the vector $[v(h);v(r);v(t)]$ and then feed it through a two layer neural network. In Santoro et al. (2017), once the entity vectors are provided by the convolutional neural network and the relation vector is provided by the long-short time memory network, for each triple the vectors are concatenated similar to ER-MLP and are fed into a four-layer neural network. Neural tensor networks (NTN) (Socher et al., 2013) combine E-MLP with a bilinear part.

3.3. Multiplicative Approaches

Multiplicative approaches define product-based functions over embeddings. DistMult (Yang et al., 2015), one of the simplest multiplicative approaches, considers the embeddings for each entity and each relation to be a single vector of size k ($v(e)$ and $v(r)$) and defines its similarity function as: $f(\mathbf{E}(h),\mathbf{E}(r),\mathbf{E}(t))=\langle v(h),v(r),v(t)\rangle$. Since DistMult does not distinguish between head and tail entities, it can only model symmetric relations. That is, by using DistMult, one has implicitly made the assumption that $(h,r,t)\in\zeta \iff (t,r,h)\in\zeta$.

To address the symmetry issue in DistMult, ComplEx (Trouillon et al., 2016) considers the embedding of each entity and each relation to be a vector of size k of complex (instead of real) numbers. For each entity e (and similarly for each relation r), $\mathbf{E}(e)$ can be represented as $re(e)+im(e)i$, where re and im are two vectors of size k and i is the square root of -1. According to ComplEx, $f(\mathbf{E}(h),\mathbf{E}(r),\mathbf{E}(t))=Real(\sum_{j=1}^{k}(re_j(h)+im_j(h)i)*(re_j(r)+im_j(r)i)*(re_j(t)-im_j(t)i)$, where $Real(\alpha+\beta i)=\alpha$. By changing the sign of im vector for the tail entity (corresponding to conjugate of a complex number), ComplEx distinguishes between head and tail entities and allows for modelling asymmetric relations. One can easily verify that the function used by ComplEx can be expanded and written as: $f(\mathbf{E}(h),\mathbf{E}(r),\mathbf{E}(t))=\langle re(h),re(r),re(t)\rangle+\langle re(h),im(r),im(t)\rangle+\langle im(h),re(r),im(t)\rangle-\langle im(h),im(r),re(t)\rangle$.

HolE (Nickel et al., 2016b) is another well-known multiplicative model, but we do not describe HolE in this paper as Hayashi & Shimbo (2017) show that HolE is isomorphic to ComplEx. RESCAL (Nickel et al., 2012) can be also considered as another multiplicative approach. RESCAL considers entity embeddings to be vectors of size k and relation embeddings to be vectors of size $k*k$. The similarity function used by RESCAL is the dot-product of the relation vector to the outer product of the entity vectors.

4. Properties of Tensor Factorization Models

We discuss fully-expressiveness, time complexity and parameter growth, interpretability, and computation redundancy of several tensor factorization approaches.

4.1. Fully-Expressiveness

Full-expressiveness is an important property of any model and has been recently the subject of study for several relational models (see e.g., (Buchman & Poole, 2017; Trouillon et al., 2017; Wang et al., 2018)).

We mentioned earlier that DistMult is not fully-expressive as it forces relations to be symmetric. Trouillon et al. (2017) prove that ComplEx is fully-expressive. In particular, they prove that any ground truth can be modelled in ComplEx with embeddings of length at most $|\mathcal{E}|\cdot|\mathcal{R}|$. Models based on deep learning can be made fully-expressive with enough hidden units, as neural networks are universal. Wang et al. (2018) prove that *TransE* is not fully-

expressive. Here, we prove that not only TransE but also the more flexible translational approaches are not fully-expressive. We also identify some of the restrictions of these models.

Proposition 1. *FSTransE is not fully-expressive and has the following restrictions: 1- If a relation r is reflexive on a subset S of entities, r must also be symmetric on S, 2- If a relation r is reflexive on a subset S of entities, r must also be transitive on S, and 3- If an entity e_1 has relation r with every entity in a subset of entities S and another entity e_2 has relation r with one of the entities in S, then e_2 must have the relation r with every entity in S.*

Corollary 1. *Less flexible variants of translational approaches such as TransE, FTransE, STransE, TransH (Wang et al., 2014), TransR (Lin et al., 2015b), etc. are also not fully-expressive and have the restrictions mentioned in Proposition 1.*

Proposition 1 identifies severe restrictions on what relations translational approaches can represent. They cannot represent any relation not having the properties identified.

4.2. Complexity

As described in Bordes et al. (2013a), to scale to the size of the KGs we currently have and keep up with their growth, a relational model must have a linear time and memory complexity. Furthermore, one of the important challenges in designing tensor factorization models is the trade-off between expressivity and model complexity. Models with many parameters usually overfit and give poor performance.

The time complexity for TransE is $O(k)$ where k is the size of the embedding vectors. Adding the projections as in STransE increases the time complexity to $O(k^2)$. Besides time complexity, the number of parameters to be learned from data grows quadratically with k. A quadratic time complexity and parameter growth may arise two issues: 1- scalability problems, 2- overfitting. Same issues exist for models such as RESCAL and NTNs that have quadratic or higher time complexities and parameter growths. DistMult and ComplEx have linear time complexities and the number of their parameters grow linearly with k. Therefore, they scale better and are less prone to over-fitting.

4.3. Interpretability and Redundancy

In DistMult, each element of the embedding vector of the entities can be considered as a feature of the entity and the corresponding element of a relation can be considered as a measure of how important that feature is to the relation. Such interpretability allows the embeddings learned through DistMult for an entity (or relation) to be potentially transferred to other domains. It also allows for incorporating expert knowledge into the embeddings to some extent. For instance, if we observe a feature of some certain types of entities, we can fix one of the elements of the embedding vector of those entities to the observed value. For many other models discussed in this paper, it is difficult to provide an interpretable description.

For ComplEx, a portion of the computations performed to make predictions is redundant. Consider a ComplEx model with embedding vectors of size 1 (for ease of exposition). Suppose the embedding vectors for h, r and t are $[\alpha_1 + \beta_1 i]$, $[\alpha_2 + \beta_2 i]$, and $[\alpha_3 + \beta_3 i]$ respectively. Then the probability of (h, r, t) being correct according to ComplEx is proportional to the sum of the following four terms:

$$1)\alpha_1\alpha_2\alpha_3 \quad 2)\alpha_1\beta_2\beta_3 \quad 3)\beta_1\alpha_2\beta_3 \quad 4)-\beta_1\beta_2\alpha_3$$

It can be verified that for any assignment of (non-zero) values to α_is and β_is, at least one of the above terms is negative. This means for a correct triple, ComplEx uses three terms to overestimate its score and then uses a term to cancel the overestimation.

The following example shows how this complexity in ComplEx affects its interpretability:

Example 1. Consider a ComplEx model with embeddings of size 1. Consider entities e_1, e_2 and e_3 with embedding vectors $[1 + 4i]$, $[1 + 6i]$, and $[3 + 2i]$ respectively, and a relation r with embedding vector $[1 + i]$.

According to ComplEx, the score for (e_1, r, e_3) is positive suggesting e_1 probably has relation r with e_3 and for (e_2, r, e_3) is negative suggesting e_2 probably does not have relation r with e_3. Since the only difference between e_1 and e_2 is that the imaginary part changes from 4 to 6, it is difficult to associate a meaning to these numbers.

Such complexity in ComplEx led us towards designing a simpler model. In the next section, we develop simple interpretable models which remove redundant computations and have half or quarter the number of summations and multiplications of ComplEx. We show in our experiments that our models generally outperform ComplEx on standard benchmarks.

5. SimplE

In *polyadic decomposition* (Hitchcock, 1927), the embedding for each entity e has two vectors $h(e)$ and $t(e)$, and for each relation r has a single vector $v(r)$. The similarity function for a triple (e_1, r, e_2) is $\langle h(e_1), v(r), t(e_2)\rangle$. One major problem with polyadic decomposition is that the two embedding vectors learned for each entity are independent of each other: observing a correct triple (e_1, r, e_2) only updates $h(e_1)$ and $t(e_2)$, not $t(e_1)$ and $h(e_2)$.

Example 2. Consider a world with two relations: $likes(p, m)$ representing if a person p likes a movie m, and

$acted(m, a)$ representing who acted in which movie. Observations on *likes* only update the t vector of movies and observations on *acted* only update the h vector. Thus, what is learned about movies through observations on *acted* does not affect the predictions about *likes* and vice versa.

To account for the independence of the two vectors for each entity in polyadic decomposition, we introduce *SimplE*. To introduce SimplE, first we define the inverse of each relation. For any relation r, we define the inverse of r as r^{-1} such that for any two entities e_1 and e_2, $(e_1, r, e_2) \iff (e_2, r^{-1}, e_1)$. As an example, $(Paris, isCapitalOf, France)$ implies $(France, isCapitalOf^{-1}, Paris)$.

SimplE considers the embedding for entities to be similar to polyadic decomposition (i.e., two vectors for each entity), but considers the embedding for each relation r to be two (instead of one) vectors, one corresponding to r and one corresponding to r^{-1}. During training, for each correct triple (e_1, r, e_2), SimplE updates the embeddings such that $\langle h(e_1), v(r), t(e_2)\rangle$ and $\langle h(e_2), v(r^{-1}), t(e_1)\rangle$ both become larger. During testing, we consider two different functions as the similarity function for a triple (e_1, r, e_2) in our experiments: 1- we ignore $r^{-1}s$ and only consider $\langle h(e_1), v(r), t(e_2)\rangle$, 2- we take the average of the two values $\langle h(e_1), v(r), t(e_2)\rangle$ and $\langle h(e_2), v(r^{-1}), t(e_1)\rangle$. In our experiments, we call the first strategy *SimplE-ignr* and the second strategy *SimplE-avg*. The probability of two entities having a relation can be computed by taking the Sigmoid of the similarity function.

Note that in *SimplE-ignr*, to find the score of a triple only one multiplication is performed between three vectors. This number is 2 for *SimplE-avg* and 4 for *ComplEx*. Thus, with the same number of parameters, *SimplE-ignr* and *SimplE-avg* reduce the computations by a factor of 4 and 2 respectively compared to *ComplEx*. Thus, *ComplEx* is doing redundant computations.

5.1. SimplE is Fully-Expressive

We provide two proofs for fully-expressiveness of SimplE with different bounds on the size of the embedding vectors.

Proposition 2. *For any ground truth over entities $\mathcal{E}$ and relations $\mathcal{R}$, there exists a SimplE model with embedding vectors of size $|\mathcal{E}| \cdot |\mathcal{R}|$ that represents that ground truth.*

Proposition 3. *For any ground truth over entities $\mathcal{E}$ and relations $\mathcal{R}$ containing γ true facts, there exists a SimplE model with embedding vectors of size $\gamma + 1$ that represents that ground truth.*

Corollary 2. *For any ground truth over entities $\mathcal{E}$ and relations $\mathcal{R}$ containing γ true facts, there exists a SimplE model with embedding vectors of size $min(|\mathcal{E}| \cdot |\mathcal{R}|, \gamma + 1)$ that represents that ground truth.*

The bound in Corollary 2 is tighter than Trouillon et al. (2017)'s bound for ComplEx ($|\mathcal{E}| \cdot |\mathcal{R}|$), as for some datasets, $\gamma + 1$ may be much smaller than $|\mathcal{E}| \cdot |\mathcal{R}|$. Note that in practice, there are many regularities in the data and a much lower vector size suffices.

5.2. SimplE & Expert Knowledge

Similar to DistMult, the embeddings of SimplE are interpretable: the entity and relation embeddings can be interpreted the same way as DistMult. Also similar to DistMult, one can incorporate certain types of expert knowledge (e.g., an observed feature of some entities) into the embeddings. However, unlike DistMult and many other models, expert knowledge in terms of logical rules can be also incorporated into the embeddings of SimplE by tying the parameters. The following propositions represent three such logical rules that can be incorporated into SimplE embeddings.

Proposition 4. *Let r be a relation such that $(e_1, r, e_2) \in \zeta \iff (e_2, r, e_1) \in \zeta$ for any two entities e_1 and e_2 (i.e. r is symmetric). This property of r can be encoded into SimplE by tying the parameters $v(r^{-1})$ to $v(r)$.*

Proposition 5. *Let r be a relation such that $(e_1, r, e_2) \in \zeta \iff (e_2, r, e_1) \in \zeta'$ for any two entities e_1 and e_2 (i.e. r is anti-symmetric). This property of r can be encoded into SimplE by tying the parameters $v(r^{-1})$ to the negative of $v(r)$.*

Proposition 6. *Let r_1 and r_2 be two relations such that $(e_1, r_1, e_2) \in \zeta \iff (e_2, r_2, e_1) \in \zeta$ for any two entities e_1 and e_2. This property of r_1 and r_2 can be encoded into SimplE by tying the parameters $v(r_1^{-1})$ to $v(r_2)$ and $v(r_2^{-1})$ to $v(r_1)$.*

5.3. From DistMult to SimplE to ComplEx

Consider a dataset where some (but not all) certain types of entities participate in asymmetric relations. In such cases, DistMult may not be appropriate as it can only model symmetric relations. A complex-valued embedding vector may not as well be appropriate as it contains more parameters than necessary. Using SimplE for such a dataset, it is possible to model the asymmetric relations and not have more parameters than necessary by considering two (head and tail) vectors only for those entities that participate in asymmetric relations, and tie the parameters of head and tail vectors for the other entities.

6. Experiments and Results

We compare SimplE empirically with several other tensor factorization models. We show that despite the simplicity of SimplE, it performs very well. We also design an experiment which validates our proofs in Section 5.2 empirically.

Table 1. Statistics on WN18 and FB15k datasets.

Dataset	$\|\mathcal{E}\|$	$\|\mathcal{R}\|$	#train	#valid	#test
WN18	40,943	18	141,442	5,000	5,000
FB15k	14,951	1,345	483,142	50,000	59,071

6.1. Datasets

We conducted experiments on two standard benchmarks: WN18 a subset of *Wordnet* (Miller, 1995), and FB15k a subset of *Freebase* (Bollacker et al., 2008). We used the same train/valid/test sets as in Bordes et al. (2013b). The statistics on these datasets are available in Table 1.

6.2. Evaluation Metrics

To measure and compare the performances of different models, for each test triple (h, r, t) once we compute the score of (h', r, t) triples for all $h' \in \mathcal{E}$ and calculate the ranking $rank_h$ of the triple having h, and once we compute the score of (h, r, t') triples for all $t' \in \mathcal{E}$ and calculate the ranking $rank_t$ of the triple having t. Then we compute the *mean reciprocal rank (MRR)* of these rankings as the mean of the inverse of the rankings:

$$MRR = \frac{1}{2 * |tt|} \sum_{(h,r,t) \in tt} \frac{1}{rank_h} + \frac{1}{rank_t} \qquad (1)$$

where tt represents the test triples.

Bordes et al. (2013b) identified an issue with the above procedure for calculating the MRR (hereafter referred to as *raw MRR*). For a test triple (h, r, t), since there can be several entities $h' \in \mathcal{E}$ for which (h', r, t) holds, measuring the quality of a model based on its ranking for (h, r, t) may be flawed. That is because two models may rank the test triple (h, r, t) to be second, when the first model ranks a correct triple (e.g., from train or validation set) (h', r, t) to be first and the second model ranks an incorrect triple (h'', r, t) to be first. Both these models will get the same score for this test triple when the first model should get a higher score.

To address this issue, Bordes et al. (2013b) proposed a modification to raw MRR. For each test triple (h, r, t), instead of finding the rank of this triple among triples (h', r, t) for all $h' \in \mathcal{E}$ (or (h, r, t') for all $t' \in \mathcal{E}$), they proposed to calculate the rank among triples (h', r, t) only for $h' \in \mathcal{E}$ such that $(h', r, t) \notin train \cup valid \cup test$. Following Bordes et al. (2013b), we call this measure *filtered MRR*.

We also report $hit@k$ measures. The $hit@k$ for a model is computed as the percentage of test triples whose ranking (computed as described earlier) is less than or equal k.

6.3. Learning SimplE Models

To learn a SimplE model, we use stochastic gradient descent with mini-batches. In each learning iteration, we iteratively take in a batch of positive triples from the KG, then for each positive triple in the batch we generate n negative triples by corrupting the positive triple. We use Bordes et al. (2013b)'s procedure to corrupt positive triples. The procedure is as follows. For a positive triple (h, r, t), we randomly decide to corrupt the head or tail. If the head is selected, we replace h in the triple with an entity h' randomly selected from $\mathcal{E} - \{h\}$ and generate the corrupted triple (h', r, t). If the tail is selected, we replace t in the triple with an entity t' randomly selected from $\mathcal{E} - \{t\}$ and generate the corrupted triple (h, r, t'). We generate a labelled batch $\mathbf{LB}$ by labelling positive triples as $+1$ and negatives as -1.

Once we have a labelled batch, following Trouillon et al. (2016) we optimize the $L2$ regularized negative log-likelihood of the batch:

$$\min_{\theta} \sum_{((h,r,t),l) \in \mathbf{LB}} softplus(-l \cdot \phi(h, r, t)) + \lambda ||\theta||_2^2 \quad (2)$$

where θ represents the parameters of the model (the parameters in the embeddings), l represents the label of a triple, λ is the regularization hyper-parameter, and $softplus(x) = log(1 + \exp(x))$. While several previous works (e.g., TransE, TransR, STransE, HolE, etc.) consider a margin-based loss function, Trouillon & Nickel (2017) show that the margin-based loss function is more prone to overfitting compared to log-likelihood.

We implemented SimplE in TensorFlow (Abadi et al., 2016)[2]. We tuned our hyper-parameters over the validation set. We used the same search grid on embedding size and λ as Trouillon et al. (2016) to make our results directly comparable to their results. We fixed the maximum number of iterations to 1000 and the batch size to 100. We set the learning rate for WN18 to 0.1 and for FB15k to 0.05 and used *adagrad* to update the learning rate after each batch. Following Trouillon et al. (2016), we generated one negative example per positive example for WN18 and 10 negative examples per positive example in FB15k. We computed the filtered MRR of our model over the validation set every 50 iterations for WN18 and every 100 iterations for $FB15k$ and selected the iteration that resulted in the best validation filtered MRR. The best embedding size and λ values on WN18 for SimplE-ignr were 200 and 0.001 respectively, and for SimplE-avg were 200 and 0.03. The best embedding size and λ values on FB15k for SimplE-ignr were 200 and 0.03 respectively, and for SimplE-avg were 200 and 0.1.

[2]Code: https://github.com/Mehran-k/SimplE

Table 2. Results on WN18 and FB15k. Best results are in bold.

	WN18					FB15k				
	MRR		Hit@			MRR		Hit@		
Model	Filter	Raw	1	3	10	Filter	Raw	1	3	10
PD	0.075	0.058	0.049	0.080	0.125	0.326	0.152	0.219	0.376	0.532
TransE	0.454	0.335	0.089	0.823	0.934	0.380	0.221	0.231	0.472	0.641
TransR	0.605	0.427	33.5	87.6	94.0	0.346	0.198	21.8	40.4	58.2
DistMult	0.822	0.532	0.728	0.914	0.936	0.654	0.242	0.546	0.733	0.824
NTN	0.53	–	–	–	66.1	0.25	–	–	–	41.4
STransE	0.657	0.469	–	–	93.4	0.543	**0.252**	–	–	79.7
ER-MLP	0.712	0.528	0.626	0.775	0.863	0.288	0.155	0.173	0.317	0.501
ComplEx	0.941	0.587	0.936	**0.945**	**0.947**	0.692	0.242	0.599	0.759	**0.840**
SimplE-ignr	0.939	0.576	0.938	0.940	0.941	0.700	0.237	0.625	0.754	0.821
SimplE-avg	**0.942**	**0.588**	**0.939**	0.944	**0.947**	**0.727**	0.239	**0.660**	**0.773**	0.838

6.4. Baselines

We compare SimplE with several existing tensor factorization approaches. Our baselines include *polyadic decomposition (PD)*, TransE, TransR, DistMult, NTN, STransE, ER-MLP, and ComplEx. Given that we use the same data splits and objective function as ComplEx, we report the results of PD, TransE, DistMult, and ComplEx from Trouillon et al. (2016). We report the results of TransR and NTN from (Nguyen, 2017), and ER-MLP from (Nickel et al., 2016b) for further comparison.

6.5. Entity Prediction Results

Table 2 shows the results of our experiments. It can be viewed that both SimplE-ignr and SimplE-avg do a good job compared to the existing baselines on both datasets. On WN18, SimplE-ignr and SimplE-avg perform as good as ComplEx, a state-of-the-art tensor factorization model. On FB15k, SimplE-avg outperforms the existing baselines and gives state-of-the-art results among tensor factorization approaches. SimplE-avg (and SimplE-ignr) work especially well on this dataset in terms of filtered MRR (which is the criteria we optimizing for) and *hit@1*.

The table shows that models with many parameters (e.g., NTN and STransE) do not perform well on these datasets as they probably overfit. The table also shows that multiplicative approaches tend to have better performances compared to translational and deep learning approaches. Even DistMult, the simplest multiplicative approach, outperforms many translational and deep learning approaches, despite not being fully-expressive. We believe the simplicity of embeddings and the scoring function is a key property for tensor factorization approaches, and SimplE shows high performance due to its simplicity and fully-expressiveness.

6.6. Incorporating Expert Knowledge

When expert knowledge is available, we might expect that a knowledge graph might not include redundant information because it is implied by expert knowledge and so the methods that do not include the background knowledge can never learn it.

In section 5.2, we showed how expert knowledge in terms of rules can be incorporated into SimplE embeddings. To test this empirically, we conducted an experiment on WN18 in which we incorporated several rules (see Appendix B) into the embeddings as outlined in Propositions 4, 5, and 6. Then, for the relations for which we had a rule, we only kept the triples of that relation in the training data that were necessary for making predictions on test triples. This reduced the number of train triples from (approx.) $141K$ to (approx.) $36K$, almost 75% reduction in size, and reduced the number of entities from 40943 to 23880. Note that this experiment provides an upper-bound on how much expert knowledge can improve the performance of a SimplE model. The goal of this experiment is to validate empirically that expert knowledge can be incorporated into SimplE models to improve their performance.

We trained SimplE-ignr and SimplE-avg (with tied parameters according to the rules) on this new training dataset with the best hyper-parameters found in the previous experiment. We refer to these two models as *SimplE-ignr-exp* and *SimplE-avg-exp*. We also trained another SimplE-ignr and SimplE-avg models on this dataset, but without incorporating the rules into the embeddings. For sanity check, we also trained a ComplEx model over this new dataset. We allowed all these five models a fixed number of iterations to train, and then reported their performance on the test set. We found that the filtered MRR for SimplE-ignr, SimplE-avg, and ComplEx were respectively 0.219, 0.220, and 0.208. For SimplE-ignr-exp and SimplE-avg-

exp, the filtered MRRs were both 0.892. In terms of $hit@k$ measures, SimplE-ignr gave 0.217, 0.218 and 0.220 for $hit@1$, $hit@3$ and $hit@10$ respectively. These numbers were 0.217, 0.219, and 0.222 for SimplE-avg, and 0.201, 0.216 and 0.218 for ComplEx. For SimplE-ignr-exp, these numbers were 0.905, 0.906 and 0.906, and for SimplE-avg-exp they were 0.906, 0.907, 0.907. The obtained results validate that expert knowledge can be incorporated into SimplE embeddings efficiently to improve its performance.

7. Conclusion

We proposed a simple interpretable fully-expressive tensor factorization model for knowledge graph completion. We showed that our model, called SimplE, performs very well empirically and has several interesting properties. For instance, we showed that expert knowledge in terms of logical rules can be incorporated into SimplE by tying the embeddings. In future, SimplE can be improved or may help improve relational learning in several ways including: 1- building ensembles of SimplE models as (Kadlec et al., 2017) do it for DistMult, 2- adding SimplE to the relation-level ensembles of Wang et al. (2018), 3- explicitly modelling the analogical structures of relations as in Liu et al. (2018), 4- using Dettmers et al. (2018)'s 1-N scoring approach to generate all negative triples that can be generated for a positive triple by corrupting its head or tail (Trouillon et al. (2016) show that generating more negative examples per positive example improves accuracy), and 5- combining SimplE with (or use SimplE as a subcomponent in) techniques from other categories of relational learning as Rocktäschel & Riedel (2017) do it with ComplEx.

References

Abadi, Martın, Agarwal, Ashish, Barham, Paul, Brevdo, Eugene, Chen, Zhifeng, Citro, Craig, Corrado, Greg S, Davis, Andy, Dean, Jeffrey, Devin, Matthieu, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

Bollacker, Kurt, Evans, Colin, Paritosh, Praveen, Sturge, Tim, and Taylor, Jamie. Freebase: a collaboratively created graph database for structuring human knowledge. In *ACM SIGMOD*, pp. 1247–1250. AcM, 2008.

Bordes, Antoine, Usunier, Nicolas, Garcia-Duran, Alberto, Weston, Jason, and Yakhnenko, Oksana. Irreflexive and hierarchical relations as translations. *arXiv preprint arXiv:1304.7158*, 2013a.

Bordes, Antoine, Usunier, Nicolas, Garcia-Duran, Alberto, Weston, Jason, and Yakhnenko, Oksana. Translating embeddings for modeling multi-relational data. In *NIPS*, pp. 2787–2795, 2013b.

Buchman, David and Poole, David. Why rules are complex: Real-valued probabilistic logic programs are not fully expressive. In *UAI*, 2017.

Carlson, Andrew, Betteridge, Justin, Kisiel, Bryan, Settles, Burr, Jr, Estevam R Hruschka, and Mitchell, Tom M. Toward an architecture for never-ending language learning. In *AAAI*, volume 5, pp. 3, 2010.

Das, Rajarshi, Dhuliawala, Shehzaad, Zaheer, Manzil, Vilnis, Luke, Durugkar, Ishan, Krishnamurthy, Akshay, Smola, Alex, and McCallum, Andrew. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. *NIPS Workshop on AKBC*, 2017.

De Raedt, Luc, Kimmig, Angelika, and Toivonen, Hannu. Problog: A probabilistic prolog and its application in link discovery. In *IJCAI*, volume 7, 2007.

De Raedt, Luc, Kersting, Kristian, Natarajan, Sriraam, and Poole, David. Statistical relational artificial intelligence: Logic, probability, and computation. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 10(2): 1–189, 2016.

Demeester, Thomas, Rocktäschel, Tim, and Riedel, Sebastian. Lifted rule injection for relation embeddings. *EMNLP*, 2016.

Dettmers, Tim, Minervini, Pasquale, Stenetorp, Pontus, and Riedel, Sebastian. Convolutional 2d knowledge graph embeddings. In *AAAI*, 2018.

Domingos, Pedro, Kok, Stanley, Lowd, Daniel, Poon, Hoifung, Richardson, Matthew, and Singla, Parag. Markov logic. In Raedt, L. De, Frasconi, P., Kersting, K., and Muggleton, S. (eds.), *Probabilistic Inductive Logic Programming*, pp. 92–117. Springer, New York, 2008.

Dong, Xin, Gabrilovich, Evgeniy, Heitz, Geremy, Horn, Wilko, Lao, Ni, Murphy, Kevin, Strohmann, Thomas, Sun, Shaohua, and Zhang, Wei. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *ACM SIGKDD*, pp. 601–610. ACM, 2014.

Fader, Anthony, Soderland, Stephen, and Etzioni, Oren. Identifying relations for open information extraction. In *EMNLP*, pp. 1535–1545, 2011.

Feng, Jun, Huang, Minlie, Wang, Mingdong, Zhou, Mantong, Hao, Yu, and Zhu, Xiaoyan. Knowledge graph embedding by flexible translation. In *KR*, pp. 557–560, 2016.

Getoor, Lise and Taskar, Ben. *Introduction to statistical relational learning*. MIT press, 2007.

Hayashi, Katsuhiko and Shimbo, Masashi. On the equivalence of holographic and complex embeddings for link prediction. *arXiv preprint arXiv:1702.05563*, 2017.

Hitchcock, Frank L. The expression of a tensor or a polyadic as a sum of products. *Studies in Applied Mathematics*, 6(1-4):164–189, 1927.

Ji, Guoliang, He, Shizhu, Xu, Liheng, Liu, Kang, and Zhao, Jun. Knowledge graph embedding via dynamic mapping matrix. In *ACL (1)*, pp. 687–696, 2015.

Kadlec, Rudolf, Bajgar, Ondrej, and Kleindienst, Jan. Knowledge base completion: Baselines strike back. *arXiv preprint arXiv:1705.10744*, 2017.

Kazemi, Seyed Mehran, Buchman, David, Kersting, Kristian, Natarajan, Sriraam, and Poole, David. Relational logistic regression. In *KR*, 2014.

Kazemi, Seyed Mehran, Fatemi, Bahare, Kim, Alexandra, Peng, Zilun, Tora, Moumita Roy, Zeng, Xing, Dirks, Matthew, and Poole, David. Comparing aggregators for relational probabilistic models. *arXiv preprint arXiv:1707.07785*, 2017.

Kimmig, Angelika, Bach, Stephen, Broecheler, Matthias, Huang, Bert, and Getoor, Lise. A short introduction to probabilistic soft logic. In *Proceedings of the NIPS Workshop on Probabilistic Programming: Foundations and Applications*, pp. 1–4, 2012.

Kimmig, Angelika, Mihalkova, Lilyana, and Getoor, Lise. Lifted graphical models: a survey. *Machine Learning*, 99(1):1–45, 2015.

Lao, Ni and Cohen, William W. Fast query execution for retrieval models based on path-constrained random walks. In *ACM SIGKDD*, pp. 881–888. ACM, 2010.

Lao, Ni, Mitchell, Tom, and Cohen, William W. Random walk inference and learning in a large scale knowledge base. In *EMNLP*, pp. 529–539, 2011.

Lin, Yankai, Liu, Zhiyuan, Luan, Huanbo, Sun, Maosong, Rao, Siwei, and Liu, Song. Modeling relation paths for representation learning of knowledge bases. *EMNLP*, 2015a.

Lin, Yankai, Liu, Zhiyuan, Sun, Maosong, Liu, Yang, and Zhu, Xuan. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, pp. 2181–2187, 2015b.

Liu, Hanxiao, Wu, Yuexin, and Yang, Yiming. Analogical inference for multi-relational embeddings. *AAAI*, 2018.

Miller, George A. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

Neelakantan, Arvind, Roth, Benjamin, and McCallum, Andrew. Compositional vector space models for knowledge base inference. In *AAAI spring symposium series*, 2015.

Nguyen, Dat Quoc. An overview of embedding models of entities and relationships for knowledge base completion. *arXiv preprint arXiv:1703.08098*, 2017.

Nguyen, Dat Quoc, Sirts, Kairit, Qu, Lizhen, and Johnson, Mark. Stranse: a novel embedding model of entities and relationships in knowledge bases. In *NAACL-HLT*, 2016.

Nickel, Maximilian, Tresp, Volker, and Kriegel, Hans-Peter. Factorizing yago: scalable machine learning for linked data. In *World Wide Web*, pp. 271–280. ACM, 2012.

Nickel, Maximilian, Murphy, Kevin, Tresp, Volker, and Gabrilovich, Evgeniy. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, 2016a.

Nickel, Maximilian, Rosasco, Lorenzo, Poggio, Tomaso A, et al. Holographic embeddings of knowledge graphs. In *AAAI*, pp. 1955–1961, 2016b.

Niepert, Mathias. Discriminative gaifman models. In *NIPS*, pp. 3405–3413, 2016.

Rocktäschel, Tim and Riedel, Sebastian. End-to-end differentiable proving. In *NIPS*, pp. 3791–3803, 2017.

Santoro, Adam, Raposo, David, Barrett, David G, Malinowski, Mateusz, Pascanu, Razvan, Battaglia, Peter, and Lillicrap, Tim. A simple neural network module for relational reasoning. In *NIPS*, 2017.

Shin, Jaeho, Wu, Sen, Wang, Feiran, De Sa, Christopher, Zhang, Ce, and Ré, Christopher. Incremental knowledge base construction using deepdive. *Proceedings of the VLDB Endowment*, 8(11):1310–1321, 2015.

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

Trouillon, Théo and Nickel, Maximilian. Complex and holographic embeddings of knowledge graphs: a comparison. *arXiv preprint arXiv:1707.01475*, 2017.

Trouillon, Théo, Welbl, Johannes, Riedel, Sebastian, Gaussier, Éric, and Bouchard, Guillaume. Complex embeddings for simple link prediction. In *ICML*, pp. 2071–2080, 2016.

Trouillon, Théo, Dance, Christopher R, Welbl, Johannes, Riedel, Sebastian, Gaussier, Éric, and Bouchard, Guillaume. Knowledge graph completion via complex tensor factorization. *arXiv preprint arXiv:1702.06879*, 2017.

Wang, Quan, Liu, Jing, Luo, Yuanfei, Wang, Bin, and Lin, Chin-Yew. Knowledge base completion via coupled path ranking. In *ACL (1)*, 2016.

Wang, Quan, Mao, Zhendong, Wang, Bin, and Guo, Li. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, 2017.

Wang, Yanjie, Gemulla, Rainer, and Li, Hui. On multi-relational link prediction with bilinear models. *AAAI*, 2018.

Wang, Zhen, Zhang, Jianwen, Feng, Jianlin, and Chen, Zheng. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, pp. 1112–1119, 2014.

Wu, Wentao, Li, Hongsong, Wang, Haixun, and Zhu, Kenny Q. Probase: a probabilistic taxonomy for text understanding. In *ACM SIGMOD*, pp. 481–492. ACM, 2012.

Yang, Bishan, Yih, Wen-tau, He, Xiaodong, Gao, Jianfeng, and Deng, Li. Embedding entities and relations for learning and inference in knowledge bases. *ICLR*, 2015.

Appendix A: Proof of Propositions

Proposition 1. *FSTransE is not fully-expressive and has the following restrictions: 1) If a relation r is reflexive on a subset S of entities, r must also be symmetric on S, 2) If a relation r is reflexive on a subset S of entities, r must also be transitive on S, and 3) If an entity e_1 has relation r with every entity in a subset of entities S and another entity e_2 has relation r with one of the entities in S, then e_2 must have the relation r with every entity in S.*

Proof. For any entity e, let $h(e) = M_h(r)v(e)$ and $t(e) = M_t v(e)$. In the ideal situation, for a triple (e_1, r, e_2) to hold, we must have $h(e_1) + v(r) = \alpha t(e_2)$ for some α.

To prove (1), let e_1 and e_2 be two entities in S. A relation r being reflexive on S suggests: $h(e_1) + v(r) = \alpha_1 t(e_1)$ and $h(e_2) + v(r) = \alpha_2 t(e_2)$. Now we must prove $(e_1, r, e_2) \iff (e_2, r, e_1)$. Suppose (e_1, r, e_2) holds. Then we know: $h(e_1) + v(r) = \alpha_3 t(e_2)$. We can conclude:

$$h(e_2)+v(r) = \alpha_2 t(e_2) = \frac{\alpha_2}{\alpha_3}\alpha_3 t(e_2) = \frac{\alpha_2}{\alpha_3}(h(e_1)+v(r))$$

$$= \frac{\alpha_2}{\alpha_3}\alpha_1 t(e_1) = \alpha_4 t(e_1), (\alpha_4 = \frac{\alpha_2\alpha_1}{\alpha_3})$$

The above equality proves that (e_2, r, e_1) holds. Going from (e_2, r, e_1) to (e_1, r, e_2) is similar.

To prove (2), let e_1, e_2, and e_3 be three entities in S. A relation r being reflexive suggests: $h(e_1)+v(r) = \alpha_1 t(e_1)$, $h(e_2) + v(r) = \alpha_2 t(e_2)$, and $h(e_3) + v(r) = \alpha_3 t(e_3)$. Now we must prove $(e_1, r, e_2) \wedge (e_2, r, e_3) \Rightarrow (e_1, r, e_3)$. Suppose (e_1, r, e_2) and (e_2, r, e_3) hold. Then we know: $h(e_1) + v(r) = \alpha_4 t(e_2)$ and $h(e_2) + v(r) = \alpha_5 t(e_3)$. We can conclude:

$$h(e_1)+v(r) = \alpha_4 t(e_2) = \frac{\alpha_4}{\alpha_2}\alpha_2 t(e_2) = \frac{\alpha_4}{\alpha_2}(h(e_2)+v(r))$$

$$= \frac{\alpha_4}{\alpha_2}\alpha_5 t(e_3) = \alpha_6 t(e_3), (\alpha_6 = \frac{\alpha_4\alpha_5}{\alpha_2})$$

The above equality proves (e_1, r, e_3) holds.

To prove (3), let e_3 and e_4 be two entities in S, and let e_2 have relation r with e_3. We prove e_2 must have relation r with e_4 as well. We know: $h(e_1) + v(r) = \alpha_1 t(e_3)$, $h(e_1) + v(r) = \alpha_2 t(e_4)$, and $h(e_2) + v(r) = \alpha_3 t(e_3)$. We can conclude:

$$h(e_2) + v(r) = \alpha_3 t(e_3) = \frac{\alpha_3}{\alpha_1}\alpha_1 = \frac{\alpha_3}{\alpha_1}(h(e_1) + v(r))$$

$$= \frac{\alpha_3}{\alpha_1}\alpha_2 t(e_4) = \alpha_4 t(e_4), (\alpha_4 = \frac{\alpha_3\alpha_2}{\alpha_1})$$

The above equality proves (e_2, r, e_4) holds. □

Proposition 2. *For any ground truth over entities $\mathcal{E}$ and relations $\mathcal{R}$, there exists a SimplE model with embedding vectors of size $|\mathcal{E}| \cdot |\mathcal{R}|$ that represents that ground truth.*

Proof. With embedding vectors of size $|\mathcal{E}| * |\mathcal{R}|$, for each entity e_i we define $h_n(e_i) = 1$ if n *mod* $|\mathcal{E}| = i$ and 0 otherwise, and for each relation r_j we define $v_n(r_j) = 1$ if n *div* $|\mathcal{E}| = j$ and 0 otherwise (see Fig 1).

Then for each e_i and r_j, the product of $h(e_i)$ and $v(r_j)$ is 0 everywhere except for $(j * |\mathcal{E}| + i)$-th element. So for each entity e_k, we set $t_{j*|\mathcal{E}|+i}(e_k)$ to be 1 if (e_i, r_j, r_k) holds and -1 otherwise. □

Proposition 3. *For any ground truth over entities $\mathcal{E}$ and relations $\mathcal{R}$ containing γ true facts, there exists a SimplE model with embedding vectors of size $\gamma + 1$ that represents that ground truth.*

Figure 1. $h(e)$s and $v(r)$s in Proposition 2.

$h(e_0)$	1	0	0	...	0	1	0	0	...	0	...	1	0	0	...	0
$h(e_1)$	0	1	0	...	0	0	1	0	...	0	...	0	1	0	...	0
$h(e_2)$	0	0	1	...	0	0	0	1	...	0	...	0	0	1	...	0
...																
$h(e_{\lvert\mathcal{E}\rvert-1})$	0	0	0	...	1	0	0	0	...	1	...	0	0	0	...	1
$v(r_0)$	1	1	1	...	1	0	0	0	...	0	...	0	0	0	...	0
$v(r_1)$	0	0	0	...	0	1	1	1	...	1	...	0	0	0	...	0
...																
$v(r_{\lvert\mathcal{R}\rvert-1})$	0	0	0	...	0	0	0	0	...	0	...	1	1	1	...	1

Table 3. Rules Used in Section 6.6.

Rule Number	Rule
1	$(e_1, hyponym, e_2) \in \zeta \iff (e_2, hypernym, e_1) \in \zeta$
2	$(e_1, member\_meronym, e_2) \in \zeta \iff (e_2, member\_holonym, e_1) \in \zeta$
3	$(e_1, instance\_hyponym, e_2) \in \zeta \iff (e_2, instance\_hypernym, e_1) \in \zeta$
4	$(e_1, has\_part, e_2) \in \zeta \iff (e_2, part\_of, e_1) \in \zeta$
5	$(e_1, member\_of\_domain\_topic, e_2) \in \zeta \iff (e_2, synset\_domain\_topic\_of, e_1) \in \zeta$
6	$(e_1, member\_of\_domain\_usage, e_2) \in \zeta \iff (e_2, synset\_domain\_usage\_of, e_1) \in \zeta$
7	$(e_1, member\_of\_domain\_region, e_2) \in \zeta \iff (e_2, synset\_domain\_region\_of, e_1) \in \zeta$
8	$(e_1, similar\_to, e_2) \in \zeta \iff (e_2, similar\_to, e_1) \in \zeta$

Proof. We use induction to prove this proposition. Let γ be zero (base of the induction). We can have embedding vectors of size 1 for each entity and relation, setting the value for entities to 1 and for relations to -1. Then $\langle h(e_i), v(r_j), t(e_k)\rangle$ is negative for every entities e_i and e_k and relation r_j. So there exists embedding vectors of size $\gamma + 1$ that represents this ground truth.

Let's assume for any ground truth where $\gamma = n - 1$ $(1 \leq n \leq |\mathcal{R}||\mathcal{E}|^2)$, there exists an assignment of values to embedding vectors of size n that represents that ground truth (assumption of the induction). We must prove for any ground truth where $\gamma = n$, there exists an assignment of values to embedding vectors of size $n + 1$ that represents this ground truth.

Let (e_i, r_j, e_k) be one of the n true facts. Consider a modified ground truth which is identical to the ground truth with n true facts, except that (e_i, r_j, e_k) is assigned false. The modified ground truth has $n-1$ true facts and based on the assumption of the induction, we can represent it using some embedding vectors of size n. Let $q = \langle h(e_i), v(r_j), t(e_k)\rangle$ where $h(e_i)$, $v(r_j)$ and $t(e_k)$ are the embedding vectors that represent the modified ground truth. We add an element to the end of all embedding vectors and set it to 0. This increases the vector sizes to $n+1$ but does not change any scores. Then we set $h(e_i)$ to 1, $v(r_j)$ to 1, and $t(e_k)$ to $q + 1$. This ensures that $\langle h(e_i), v(r_j), t(e_k)\rangle > 0$ for the new vectors, and no other score is affected. □

Proposition 4. *Let r be a relation such that $(e_1, r, e_2) \in \zeta \iff (e_2, r, e_1) \in \zeta$ for any two entities e_1 and e_2 (i.e. r is symmetric). This property of r can be encoded into SimplE by tying the parameters $v(r^{-1})$ to $v(r)$.*

Proof. If $(e_1, r, e_2) \in \zeta$, then a SimplE model makes $\langle h(e_1), v(r), t(e_2)\rangle$ and $\langle h(e_2), v(r^{-1}), t(e_1)\rangle$ positive. By tying the parameters $v(r^{-1})$ to $v(r)$, we can conclude that $\langle h(e_2), v(r), t(e_1)\rangle$ and $\langle h(e_1), v(r^{-1}), t(e_2)\rangle$ also become positive. Therefore, the SimplE model predicts $(e_2, r, e_1) \in \zeta$. □

Proposition 5. *Let r be a relation such that $(e_1, r, e_2) \in \zeta \iff (e_2, r, e_1) \in \zeta'$ for any two entities e_1 and e_2 (i.e. r is anti-symmetric). This property of r can be encoded into SimplE by tying the parameters $v(r^{-1})$ to the negative of $v(r)$.*

Proof. If $(e_1, r, e_2) \in \zeta$, then a SimplE model makes $\langle h(e_1), v(r), t(e_2)\rangle$ and $\langle h(e_2), v(r^{-1}), t(e_1)\rangle$ positive. By tying the parameters $v(r^{-1})$ to the negative of $-v(r)$, we can conclude that $\langle h(e_2), v(r), t(e_1)\rangle$ and $\langle h(e_1), v(r^{-1}), t(e_2)\rangle$ become negative. Therefore, the SimplE model predicts $(e_2, r, e_1) \in \zeta$. □

Proposition 6. *Let r_1 and r_2 be two relations such that $(e_1, r_1, e_2) \in \zeta \iff (e_2, r_2, e_1) \in \zeta$ for any two entities e_1 and e_2. This property of r_1 and r_2 can be encoded into SimplE by tying the parameters $v(r_1^{-1})$ to $v(r_2)$ and $v(r_2^{-1})$ to $v(r_1)$.*

Proof. If $(e_1, r_1, e_2) \in \zeta$, then a SimplE model makes $\langle h(e_1), v(r_1), t(e_2)\rangle$ and $\langle h(e_2), v(r_1^{-1}), t(e_1)\rangle$ positive. By tying the parameters $v(r_2^{-1})$ to $v(r_1)$ and $v(r_2)$ to $v(r_1^{-1})$, we can conclude that $\langle h(e_1), v(r_2^{-1}), t(e_2)\rangle$ and $\langle h(e_2), v(r_2), t(e_1)\rangle$ also become positive. Therefore, the SimplE model predicts $(e_2, r_2, e_1) \in \zeta$. □

Appendix B: Rules Used in Section 6.6

The rules used for the experiment in Section 6.6 can be found in Table 3.