Common sense reasoning against algorithmic misinformation using Knowledge Graph Embeddings

by

Aabir Abubaker Kar

July 2021

A thesis presented in partial fulfillment of requirements for the

Master of Arts in Computational Social Science

at

The University of Chicago

Faculty Advisor: James Evans

Preceptor: Sanja Miklin

Abstract

Knowledge Graph Embedding algorithms learn low-dimensional vector representations for facts in a Knowledge Graph (where each fact is a $relation$ edge between $head$ and $tail$ nodes). Learned embeddings are increasingly adapted for downstream tasks (like question-answering, statement classification, recommendation engines, and even auto-completion of incomplete Knowledge Graphs). In the real world, training datasets of facts (in domains like news events, public records, or user actions) can be inaccurate due to sampling issues, bias, malicious intent, or noise. This makes it important to design for robustness to corrupted data. This paper investigates how two popular Knowledge Graph Embedding algorithms (TransE, DistMult) respond to deliberate corruption of training data, using the Freebase15K-237 dataset. This paper uses a novel true/false triplet classification evaluation to demonstrate the utility of learned embeddings as features for truth prediction, showing surprising robustness to corrupted data. It uses negative sampling to operationalize different types of common-sense reasoning in the stages of learning (model training) and testing (model evaluation) - demonstrating the usefulness of naive negative sampling strategies, contrary to prevailing intuitions in the literature. It uses this to frame a social science interpretation for the Knowledge Graph Embedding training process, with implications for designing for robustness against inaccuracies in Knowledge Graph data.

1 Introduction

Knowledge Graphs are emerging as the de facto data structure for storing large numbers of $(head, relation, tail)$ facts at scale, with adoption by technology giants like Google. The following quote is from Google's blog [author, 2021]:

> Google's search results sometimes show information that comes from our Knowledge Graph, our database of billions of facts about people, places, and things. The Knowledge Graph allows us to answer factual questions such as "How tall is the Eiffel Tower?" or "Where were the 2016 Summer Olympics held." Our goal with the Knowledge Graph is for our systems to discover and surface publicly known, factual information when it's determined to be useful.

The accuracy of online information has been found to be increasingly unreliable, particularly with the rise of social media and digital information sources. As summarized by Tucker et al. [2018],

inaccurate information can take a number of different forms — such as 'fake news', rumors, propaganda, deliberately factually incorrect information (disinformation), inadvertently factually incorrect information (misinformation), politically slanted information, 'hyperpartisan' news, clickbait, and conspiracy theories. The effects of exposure to inaccurate information on human decision-making in social, economic, and political systems have been the subjects of many studies. For example, Bago et al. [2020] show that time newsreaders spend deliberating on headlines helps decreases their belief in fake ones, while leaving their belief in true ones relatively unaffected. However, there is currently a void in our understanding of the effects of disinformation or misinformation on massive, scalable, **algorithmic knowledge-representation systems**. This paper addresses that gap.

This paper interrogates the algorithmic response to corrupted information in the specific case of corrupted facts. A fact can be formalized as a relation connecting entities (such as $(head$: **Joe Biden**$, relation$: **Is President**$, tail$: **USA**$)$), with optional attributes for context (such as $\{inaugurationDate$: **January 20, 2021**$\}$). A Knowledge Graph (henceforth used interchangeably with 'KG') is a data structure for storing a set of such facts, where each fact is a $(head, relation, tail)$ or (h, r, t) triplet. Here $h, t \in \mathcal{E}$ are entities representing nodes, and $r \in \mathcal{R}$ are relations representing edges. See Figure 1 for an illustration.

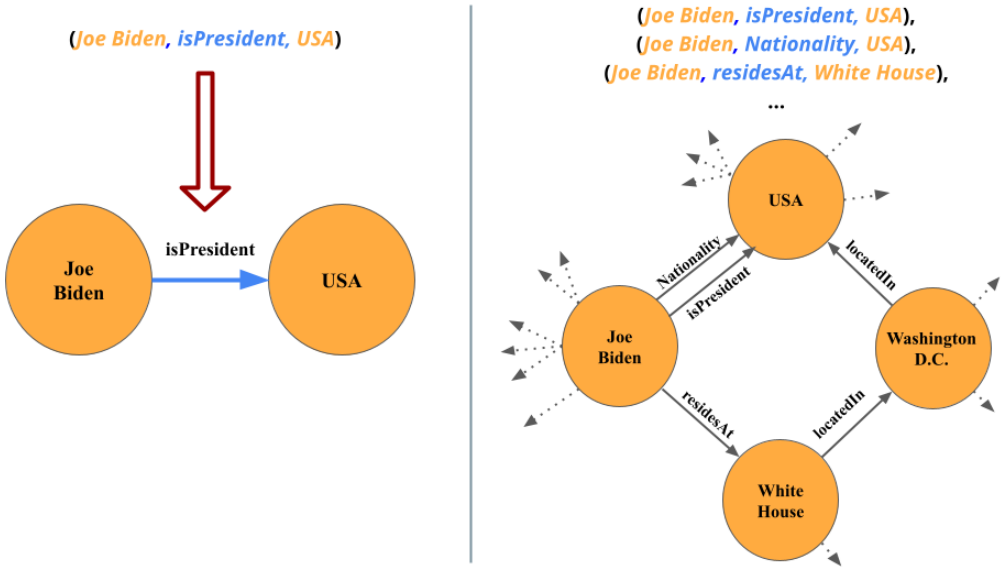

Figure 1: Illustration of a Knowledge Graph

Knowledge Graph Embeddings (henceforth used interchangeably with 'KGEs') are a family of algorithms devised to find low-dimensional vector representations for entities and relationships in a Knowledge Graph. Vector representations $\vec{h}$, $\vec{r}$ and $\vec{t}$ are learned to preserve different types of logic on the facts (h, r, t) in the training set. If additional attributes are available, they can be incorporated into the algorithm design to learn 'joint embeddings' (Toutanova et al. [2015], Peters et al. [2018]).

Because KGEs preserve context from the graph structure, the learned vectors are found to be useful features for downstream applications like question-answering, triplet classification, recommendation engines, and even auto-completion of incomplete Knowledge Graphs. The value of these embeddings in a multiplicity of use-cases is analogous to the utility of word vectors from Mikolov et al. [2013b]'s Word2Vec or Devlin et al. [2019]'s BERT in building a variety of downstream applications in natural language processing.

With representation learning for facts, a natural social concern is that of misinformation and its effect on KGEs. For example, millions of people could be exposed to misinformation if Google's Knowledge Graph accidentally learns spurious facts from poor sources or if it can be maliciously induced to learn spurious facts. Despite growing concerns regarding the volume, quality and effects of online misinformation among social scientists (Bago et al. [2020], Spohr [2017]), computational scientists (Jackson and Foucault Welles [2015], Chen et al. [2015]) as well as in the public sphere (Tucker et al. [2018]), there is currently no understanding of how KGEs respond to corrupted training data. This work is the first exploration in this regard, as far as the author can ascertain.

KGE models are trained on deliberately corrupted training data and evaluated on an adaptation of Socher et al. [2013]'s true/false triplet classification task. Contrary to past approaches to triplet classification, embeddings are fed into Logistic Regression and Multilayer Perceptron classifiers for true/false classification. This allows us to quantify utility of learned vectors as features for predicting the truth of unseen triples and perform feature analysis on the learned vector dimensions. This paper demonstrates a strong best performance of 93.02% on the true/false triplet classification evaluation task. This result demonstrates that relatively simple models can use trained embeddings to estimate the credibility of unseen fact triples — the latter task being a critical aspect of the knowledge base auto-completion task.

Surprisingly, adding spurious facts has little effect on the utility of learned embeddings as features for truth prediction. Even with up to 50% corrupted datasets, the test accuracies stay within 1% of the mean. This surprising result is discussed in detail in Section 9.

Safavi and Koutra [2020] characterize negative knowledge as "what is explicitly false or non-viable with respect to a specific goal". Negative sampling is the schema for generating negative knowledge — specifically, for corrupting a given true (h, r, t) triple. Qualitatively, different types of negative sampling can placed on a spectrum from 'naive' (generating nonsense) to 'intelligent' (generating plausible falsehood).

This paper evaluates different negative sampling strategies during learning (i.e. model training) and testing (i.e. model evaluation). Works like Cai and Wang [2018] and Safavi and Koutra [2020] anecdotally claim that negative sampling at train-time should be intelligent rather than naive — the

rationale being that nonsensical falsehoods are unhelpful during training. The results in this paper demonstrate a contrary finding — that the naive Uniform negative sampling strategy is valuable during training as it helps separate unrelated entities and relations in the embedding space. The strong and unexpected performance of 'unintelligent' sampling on the novel evaluation task exposes directions for future research. At evaluation-time, the paper also demonstrates an intuitively pleasing result — naive negative sampling generates more easily detectable falsehoods.

Finally, this paper uses the analysis to identify valuable use cases for this work, like Knowledge Base auto-completion, recommendation systems, and question-answering. It suggests the need to diversify and expand the suite of KGE evaluation tasks, and makes suggestions for designing KGEs that are robust to different types of false information (classified by intent, design, plausibility). The value of this technology to applications in human-in-the-loop and machine-in-the-loop fact-checking are also explored.

This work is accompanied by an open-source software package `weboftruth` for analyzing the effects of data corruption on any KG dataset and KGE algorithm.

2 Related Work

In this section, I describe KGEs in analogy with research on word embeddings, situating this paper as an exploration of the social implications of representation learning algorithms.

2.1 Representation learning for words and NLP

Representation learning has become a powerful and ubiquitous intermediate step to downstream applications in a host of fields, perhaps the most prominent of which is natural language processing. The most basic and powerful representations learned are for words. Simpler word embedding models like tf-idf, GloVe [Pennington et al., 2014], or Word2Vec [Mikolov et al., 2013a] ascribe numerical values to words based on logics of their counts, co-occurrences, or other features in a corpus. The most successful contemporary representation learning algorithms such as BERT [Devlin et al., 2019] are based on deep neural networks.

The utility of neural networks as universal function approximators has led to the advent of language models, which model the probability distributions of tokens, words, and entire sentences or documents - relying on the implicit construct of a 'word vector'. In deep neural language models like BERT [Devlin et al., 2019], word vectors are 'designed' by the model (by the choice of loss function and optimizer, during the training process) into being semantically useful features. 'Usefulness' here is defined by performance on tasks such as guessing a masked word in a sentence, guessing whether a

pair of sentences logically follows each other or not. These word vectors can then be then used to engineer more complex downstream applications such as question answering, sentence paraphrase generation, and so on.

learned word representations are able to capture analogical reasoning, as shown by Mikolov et al. [2013b]. They demonstrate that in their model, "the male/female relationship is automatically learned, and with the induced vector representations, 'King - Man + Woman' results in a vector very close to 'Queen.'" High-dimensional vector representations are thus demonstrably able to encode semantic meaning in their various dimensions.

2.2 Knowledge Bases and Knowledge Graphs

The Knowledge Base has been a useful a data structure for storing and structuring facts for several decades [Jarke et al., 1989]. A Knowledge Base consists of entities which may have attributes such as category, description, timestamp, etc.; and relations that can connect entities, and themselves have relation attributes. Attributes can be used to organize data within classes, hierarchies and subsets - as well as to describe relationships, events, and dynamic processes. The generalized Knowledge Base can be used to store unstructured information. A Knowledge Graph is a restricted version of the Knowledge Base that can store structured 'fact' triples of the form $(head, relation, tail)$. Figure 1 is an illustration of a knowledge graph.

Advances in the field of NLP, specifically subfields like Information Extraction, language modeling, Named Entity Recognition, and coreference resolution have made it possible to parse large volumes of unstructured text data into Knowledge Bases. Methods like AutoKG [Yu et al., 2021] which parses unstructured text into a Knowledge Graph, make these methods highly accessible. Knowledge Graphs can be parsed from datasets mined or scraped from the internet with relative ease, and KGE methods can then be applied to them. The well-studied abundance of inaccurate information on the internet [Tucker et al., 2018] becomes a pertinent concern for such automated Knowledge Graphs.

2.3 Representation learning for facts: KGEs

Knowledge Graph Embeddings, like word embeddings, are a type of representation learning algorithm. Analogously with word vectors, a KGE algorithm ascribes vector values to entities and relationships forming a Knowledge Graph, based on logics of their counts, co-occurrences or other features. Some popular KGE algorithms include TransE by Bordes et al. [2013], SimplE by Kazemi and Poole [2018], KG2Vec by Wang et al. [2021], ConvE by Dettmers et al. [2018], HolE by Nickel et al. [2015] and many others. KGE algorithms based on deep neural networks have become the standard, and

only these are discussed in this work. Models like Node2Vec [Grover and Leskovec, 2016], which learn embeddings for graph data without deep learning are not dealt with here.

KGEs are highly suited to problem formulations involving link prediction, node classification, question answering and triplet classification. For this reason, they have found applications in a variety of important tasks like question-answering, statement classification, recommendation engines, and even auto-completion of incomplete Knowledge Graphs [Dai et al., 2020, Nickel et al., 2016].

2.4 Social issues and representation learning

A growing body of work is investigating the problematic features of many types of machine learning algorithms, often finding issues of sample skewedness, inequity due to poor performance on data minorities, or biasedness in results. Such work is situated as computational social science, as the problems with machine learning algorithms typically have socioeconomic or even political consequences.

For example, in the case of representation learning for words, Caliskan et al. [2017] show how Word2Vec models trained on standard corpora learn human-like biases, specifically biases "toward race or gender". They note the somewhat tautological finding that observed biases were also "veridical reflecting the status quo distribution of gender with respect to careers or first names". Their 'Word Embeddings Association Test' uses word vectors and the analogy of their distributedness in 'high-dimensional space' to test bias in their 'closeness' to different target concepts. For example, gender bias manifests as male attribute words being statistically 'closer' to science- or math-related terms than female attributes. Machine learning applications that use these word representations as features are then prone to furthering the same biases by encoding them in their behaviors.

Particularly in the field of natural language processing, social biases (commonly gender bias) have been detected in word vectors from Word2Vec [Caliskan et al., 2017] and BERT (Bhardwaj et al. [2021], Rekabsaz et al. [2021]), as well as in downstream applications like machine translation [Stanovsky et al., 2019] and coreference resolution [Zhao et al., 2018].

2.5 Social issues and KGEs

As advanced NLP percolates into the field of KGEs, algorithms such as AutoKG [Yu et al., 2021] will potentially be parsing large swathes of data from the internet into dynamically updated knowledge graphs in domains such as social media, retail, public policy, governance, politics, and news. KGEs are the technology of choice for building applications and services on top of these KGs. Work that explores the social implications of KGEs is therefore crucial, but currently extremely nascent. At the time of writing, I was only able to find a single pre-print paper by Fisher et al. [2020] which

observes a gender bias similar to that reported by Caliskan et al. [2017] — observing that the model overwhelmingly predicts notable people to be male rather than female. However, this can be explained as simply veridical, with significantly more facts in the dataset describing males than females.

There is currently no work that directly adopts the trained embeddings as features for truth prediction. Socher et al. [2013] use the model's in-built scoring feature to identify a threshold such that triplets above the threshold are classified as true. However, this approach ignores the possible utility of the trained vectors themselves as features for truth prediction.

Work by Safavi and Koutra [2020] (still in pre-print) discusses the importance of 'negative commonsense knowledge' in the training process. They show that "obtaining meaningful negatives—i.e., knowledge of what is explicitly false or non-viable with respect to a specific goal — is nontrivial." They discuss how negative sampling (i.e. the schema for inferring false facts based on trusted true ones) can be designed to infer 'meaningful' rather than arbitrary or noisy negative facts. In this work, $sampler_{XYZ}$ refers to the negative sampler used for XYZ task.

3 Knowledge Graph Embedding algorithms

A KGE algorithm learns low-dimensional vector representations for each entity and each relation in the KG. The smooth embedding space in which these vectors lie offers several advantages. Computationally, a large and sparse graph holding n_{facts} triples can be reduced to the storage of $n_{entities} + n_{relations}$ d-dimensional vectors. Semantically, these d-dimensional vectors can be designed in ways that preserve useful information from the original graph, and allow for complex fine-tuning - analogous to the potential for fine-tuning language models.

The input to the algorithm is a training set of 'trusted' fact claims. The output is a set of low-dimensional vector representations, one for each entity $\in \mathcal{E}$ and each relation $\in \mathcal{R}$ in the set.

In general, a KGE model has five parts: an embedding look-up layer, a negative sampler, a scoring function f, a loss function $\mathcal{L}$, and an optimizer.

3.1 Embedding Look-up Layer

A batch of training facts consists of triples $(h, r, t)_i$. For each i, h and t are mapped to their respective entity indices, while r is mapped to its relationship index. The **embedding look-up layer** looks up embeddings for each of (h, r, t), retrieving vectors $(\vec{h}, \vec{r}, \vec{t})$. Initially, vectors are initialized randomly for every $h, t \in \mathcal{E}$ and $r \in \mathcal{R}$.

The embedding layer is parametrized by the weights of any neural layers it encodes, as well as the resulting embeddings for each entity and relation - we collectively refer to these as the model parameters Θ.

3.2 Negative sampler

For each 'true' training fact, some number of negative facts is sampled and learned as false. This process relies on the **local closed world assumption**, which states that the knowledge graph is locally complete, and the absence of a fact means that it is necessarily false. There are multiple ways to generate negative facts. Given a triple (h, r, t), we can corrupt the head to get (h', r, t), or the tail to get (h, r, t'), where h' and t' are sampled from other subjects and objects in the input knowledge graph. A similar exercise takes place in many popular embedding algorithms. For example, in the skipgram implementation of Word2Vec, a random word absent from the central word's neighborhood is picked as a negative example, effectively telling the model that those two words do *not* occur together.

In the case of KGEs, a variety of sampling strategies can be employed while replacing h or t - Uniform sampling or Bernoulli sampling are commonly used. Cai and Wang [2018] and Safavi and Koutra [2020] have demonstrated the strategic value of smarter negative sampling in improving performance. More complex negative sampling based on features of the entities or relationships can also be used. For example, negative samples can be generated to preserve entity type - replacing a country only with another country, or a person only with another person.

Negative sampling and its interpretation in this paper are discussed in detail in Section 7.2.

3.3 Scoring function

Let the set of true (h, r, t) triples be T^+ and the set of negatively sampled (hence, false) triples be T^-.

The scoring function now computes a score for each true triple in T^+, as well as each false triple in T^-. Each model offers a different justification for their definition of scoring function. The computed scores are then used to compute the loss.

3.4 Loss function

Loss has the same interpretation here as in typical neural networks - it is a measure of poorness of a prediction. The lower the loss on any datapoint, the better the model performs on that datapoint.

A commonly used loss in KGE algorithms is the pairwise margin-based hinge loss introduced in Bordes et al. [2013].

$$\mathcal{L}(\Theta) = \sum_{t^+ \in T^+} \sum_{t^- \in T^-} \max\left\{0, \gamma + f(t^-; \Theta) - f(t^+; \Theta)\right\}$$

3.5 Optimizer

The optimizer now backpropagates through the model to optimize it so as to maximize the score for true triples and minimize the score for synthetic false triples. Popular optimizers like SGD, Adam, Adagrad can be used here.

This process is repeated until convergence.

4 Types of KGE models

Different KGE models differ along various axes, such as scoring function, neural architectures, tensor representations, loss functions, and negative sampling. Here I briefly describe two categories of KGE models, and one example from each: translational embedding model (TransE), and a bilinear model (DistMult).

More advanced methods have also been explored — such as convolutional algorithms like ConvKB [Nguyen et al., 2018] which learns feature maps by applying filters on each dimension of $\vec{h}$, $\vec{r}$, and $\vec{t}$) or hyperbolic algorithms like HyperKG [Kolyvakis et al., 2020], which exploits hyperbolic geometries for embedding knowledge graphs with power-law degree distributions.

A complete exploration of all these categories is beyond the scope of this work.

4.1 Translational models

Translational models treat relation vectors as translations (i.e. spatial displacements) that direct the head vector towards the tail vector. Therefore, $\vec{r}$ (or some transformation on it) is expected to 'translate' $\vec{h}$ through space towards $\vec{t}$.

Developed in 2013, TransE Bordes et al. [2013] is a translational model that attempts to learn d-dimensional entity and relationship embeddings such that for 'true' triples (h, r, t), their embeddings attempt to minimize the quantity:

$$|\vec{t} - (\vec{h} + \vec{r})|$$

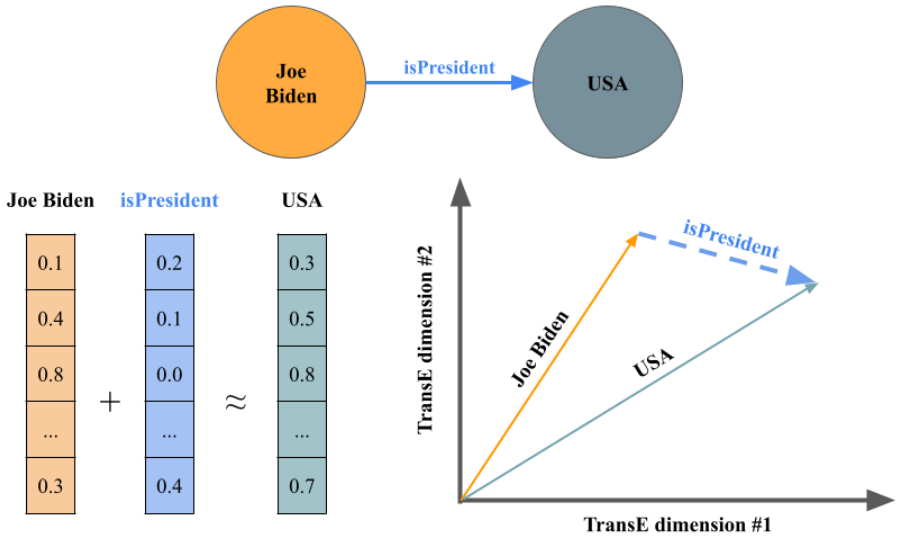

Figure 2: Schematic of TransE vector logic

Translational models are powerful because of their relative interpretability and convenient computation. However they struggle with heterogeneity (when some relations occur much more frequently than others, the model overfits the frequent relations and underfits the rare ones). Similarly, imbalances in the occurence of entities as head or tail can cause overfitting towards the more dominant position of occurrence.

4.2 Bilinear models

Bilinear models allow for more complex representations of relationships by applying matrix multiplications on head and tail vectors (i.e. bilinear transformations) for each relationship. The $relation$ vector $\vec{r}$ is expanded to a tensor R, implicitly ascribing meaning to the vector product $\vec{h}R\vec{t}$.

Developed in 2015, DistMult Yang et al. [2015] is a bilinear algorithm. Here, the relationship is represented as a tensor factorization problem

$$g_r^b(\vec{h}, \vec{t}) = \vec{h}^T \cdot M_r \cdot \vec{t}$$

where M_r is a diagonal $n \times n$ matrix.

As seen in Figure 3, the relationship 'vector' $\vec{r}$ is simply the diagonal of the relationship matrix M_{ri}, so that $v_{ri} = M_{rii} \forall i$.

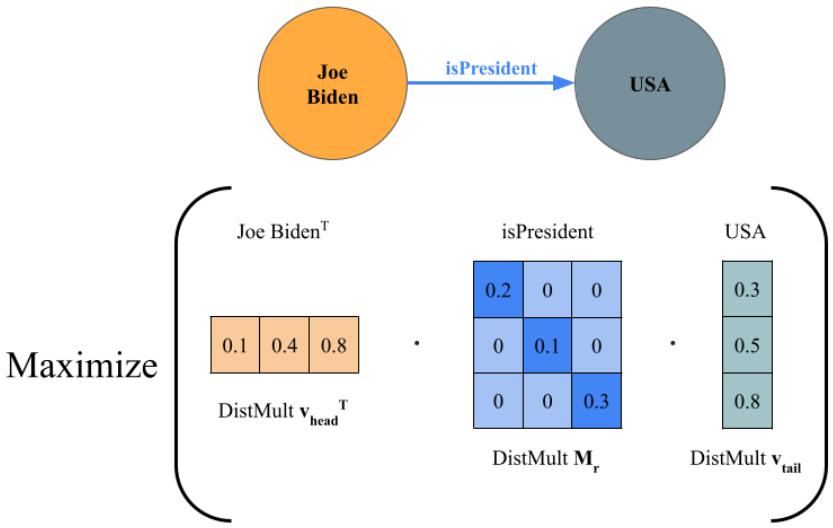

Figure 3: Schematic of DistMult vector/tensor logic

Bilinear relationships offer powerful and flexible representations for relationships by allowing for a variety of transformations via matrix multiplication. However, bilinear models are more expensive to train and less interpretable than translational models.

Other types of KGE algorithms, such as deep neural models (like Socher et al. [2013]'s NTN), convolutional models (like Dettmers et al. [2018]'s ConvE), or hyperbolic models (like Kolyvakis et al. [2020]'s HyperKG) are not discussed here. The reader is referred to Dai et al. [2020] for a comprehensive survey of the field.

In this paper, I experiment with TransE and DistMult. Both these models have the advantage of being widely used, well-understood, and computationally tractable. An understanding of their robustness against corrupted data will inform the investigation of other, more complex models.

5 Data

5.1 FB15k-237

In this paper, I make use of the FB15k-237 dataset [Toutanova et al., 2015], which is a Knowledge Graph of 14,541 entities and 237 relationship types, connected by over 300,000 edges (i.e. facts).

FB15k-237 is a subset of the original, FB15k dataset [Bollacker et al., 2008] which is composed of 14,951 entities, 1,345 relationship types (more by a factor of nearly 6) and nearly 600,000 edges. As noted by Toutanova et al. [2015] Toutanova and Chen (2015), the link prediction evaluation task becomes trivial with FB15k because the dataset contains many reversible relations. FB15k-237 is a

reduced version of this dataset that eliminates many of these reversible relations. It has emerged as the standard KG dataset in the literature.

Each entity and relation can be mapped to a meaningful text string. $2/237$ relation types and $446/14,951$ entities which could not be mapped are excluded from the data. The resulting complete dataset comprises $14,505$ entities and 235 relation types, with $N_{total} = 304,205$ facts.

The data are split into 3 sets: train set ($N_{facts} = 272,115 \approx 89\% \cdot N_{total}$), validation set ($N_{facts} = 17,087 \approx 5\% \cdot N_{total}$) and test set ($N_{facts} = 19,929 \approx 6\% \cdot N_{total}$). The split is designed by the data providers to preserve at least one occurrence of each h, r and t in the validation and test sets. This helps avoid out-of-sample problems during evaluation.

The dataset consists of simple facts about people, places and other entities. A large number of facts are focused on the entertainment industries, specifically on award nominations and winners, release dates, and creative roles such as 'director' or 'actor'. The 5 most frequently and least frequently occurring relations are shown in the following table:

Rank	Relation	Count
1	/award/award_nominee/award_nominations. /award/award_nomination/award_nominee	16331
2	/film/film/release_date_s. /film/film_regional_release_date/film_release_region	15567
3	/award/award_nominee/award_nominations. /award/award_nomination/award	13980
4	/people/person/profession	13366
5	/award/award_category/nominees. /award/award_nomination/nominated_for	11035
...	...	...
231	/film/film/film_art_direction_by	108
232	/base/saturdaynightlive/snl_cast_member/seasons. /base/saturdaynightlive/snl_season_tenure/cast_members	104
233	/tv/non_character_role/tv_regular_personal_appearances. /tv/tv_regular_personal_appearance/person	103
234	/sports/sports_position/players. /american_football/football_historical_roster_position/position_s	102
235	/film/person_or_entity_appearing_in_film/films. /film/personal_film_appearance/type_of_appearance	45

Table 1: 5 most frequent and least frequent $relation$s in FB15K-237 dataset

As seen above, the facts are focused on the entertainment and sports industries, with information about feature films, television shows, musical performances, and award nominations. Some of the frequently occurring relations from more general domains are listed below:

- `/people/person/profession`: $13,366$ facts connecting people and professions (seen in the table above)
- `/location/location/contains`: $5,834$ facts establishing geographical relationships between places

- `/people/person/gender`: $4,526$ facts attaching (binary) gender attributes to people
- `/education/educational_institution/students_graduates./education/education/student`: $3,227$ facts connecting people with educational instutions
- `/education/educational_institution/students_graduates./education/education/major_field_of_study`: $3,218$ facts connecting people with fields of study

Tables 2 and 3 show the 10 most frequent head and tail entities in the dataset.

Rank	Head entity	Count
1	United States of America	1517
2	guitar	754
3	president	723
4	forward	622
5	Bachelor of Arts	619
6	defender	591
7	midfielder	582
8	United Kingdom	553
9	marriage	517
10	piano	509

Table 2: 10 most frequent $head$ entities in FB15K-237 dataset

Rank	Tail entity	Count
1	United States of America	7116
2	United States dollar	4746
3	male organism	3563
4	marriage	3159
5	English	3130
6	actor	2764
7	DVD	1692
8	United Kingdom	1670
9	drama film	1238
10	midfielder	1203

Table 3: 10 most frequent $tail$ entities in FB15K-237 dataset

6 Evaluation Measures

In this paper, each KGE model is evaluated an adaptation of a common evaluation tasks in the literature: **triplet classification**. In this task, the embeddings of (h, r, t) triples are used to classify those triples into categories such as true/false for fact claims, positive/neutral/negative for sentiment analysis, and so on. This evaluation task is easily interpretable while experimenting with corrupted data. It lends itself to the natural question, “How does learning corrupted facts affect a model’s ability to classify new facts as true or false?"

The other popular evaluation task in the KGE literature **link prediction**, the problem of correctly predicting a the hidden entity in a masked triple e.g. the correct h' in $(?, r, t)$. While this evaluation task is highly pertinent for applications in question answering, it uses metrics like Mean Rank and Mean Reciprocal Rank that are not directly interpretable in the case of corrupted training data. For this reason, link prediction evaluation is excluded from this analysis.

6.1 Triplet Classification

Triplet classification (proposed by Socher et al. [2013]) is the binary classification problem of ascertaining whether an unseen triple (h', r', t') is true or false.

A KGE model trained on KG_{train} is presented an unseen validation set of true facts, $KG_{val|true}$. A negative sampler generates an equal number of false facts $KG_{val|false}$. The two sets are combined into KG_{val}, $|KG_{val}| = 2|KG_{val|true}|$.

A classifier model on the **labeled** KG_{val} for which every (h, r, t) is either true or false.

The accuracy of this classifier is then evaluated on another unseen Knowledge Graph KG_{test} - comprising an equal number of true facts from $KG_{test|true}$ and negatively sampled false facts $KG_{test|false}$.

The variable $sampler_{evaluation}$ identifies the type of negative sampling chosen for generating these triples.

The **binary classifier accuracy** is the accuracy with which this model correctly classifies facts from KG_{val} as true or false. A higher accuracy is better.

Triplet classification is conventionally done using a threshold-based method described in Socher et al. [2013]. This method makes use of the models in-built score function, setting a score threshold above which a triplet is classified as 'positive' or 'true'. While this is a useful approach, it ignores the potential value of the learned embeddings as features for an independent classifier — an idea that is validated by the success of convolutional methods on pre-trained vector representation in fields like natural language processing as well as KGEs (see ConvE by Dettmers et al. [2018]). It also relies on the model's in-built score function and the relatively naive binning of scores for classification using thresholds. This approach is unappealing for multiclass classification.

In this paper, I explore an alternate approach, using the embeddings as raw features on a two simple classifiers, a Logistic Regression (LR) and a Multilayer Perceptron (MLP). Ridge Regression is found to perform too poorly for consideration and is omitted. This choice allows us to explore the value of the trained vectors as independent features for triplet classification. It offers more interpretability,

allowing visualization and interrogation of the semantic meanings of learned vector dimensions, as discussed in Section 9.

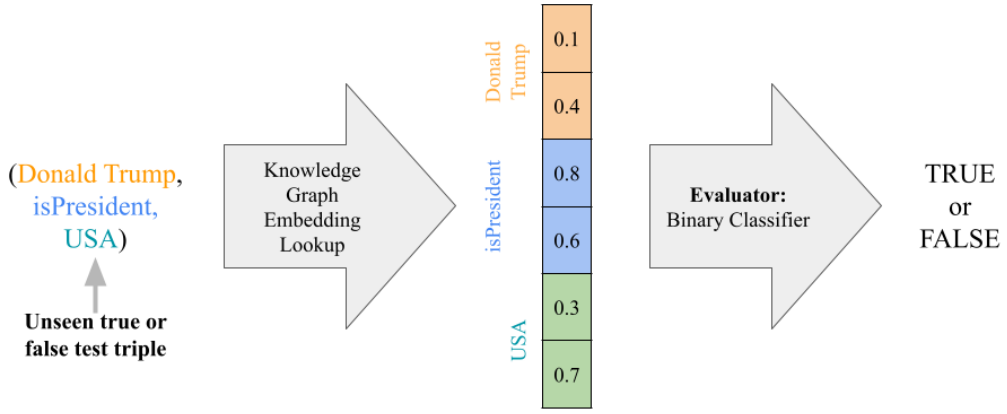

Figure 4: Schematic of Triplet Classification evaluation task

Each classifier is trained on binary classification of unseen facts from the validation set ($N = 17,087$) and then tested on unseen facts from the test set ($N = 19,929$). The MLP used has 100 hidden units and RELU activation, while the Logistic Regression uses a standard `newton-cg`solver.

Here I report both the train and test accuracy for each of the two classifiers, producing 4 metrics per evaluated KGE model.

Another common evaluation task with KGEs is link prediction, i.e. the task ot completing unseen facts (h', r', t') where one of h' or t' (chosen randomly) is masked. This task is not used here as it is more complex to evaluate the effects of corrupted training data on it.

7 Experimental Setup

The current work experiments with 2 Knowledge Graph Embedding models (TransE and DistMult), and a single dataset (FB15k-237). A total of 6 configurations of each model are trained, as described below.

7.1 Truth-share $ts \in \{100, 80, 50\}$

I define the **truth-share** ts of a corrupted training knowledge graph as the percentage of facts in it that exist in the original graph. A 50% truth-share training set contains an equal number of true and false fact claims. A 100% truth-share training graph contains no false fact claims.

This work experiments with truth-shares of 100%, 80% and 50%. Negative sampling is the method used to generate false fact claims, described below.

The motivating question here is: how do artificially generated false facts affect a KGE model's performance, ceteris paribus?

7.2 Negative sampling

There are 3 popular types of negative sampling.

Uniform Negative Sampling Given a triple h, r, t, either h or t is selected for replacement (with equal probability). It is then replaced by an entity chosen randomly from the set of entities. Each choice is equiprobable. For example, with Uniform sampling it is equally likely for the triple $(JoeBiden, isPresident, USA)$ to be substituted with $(JimCarrey, isPresident, USA)$ as with $(Chicago, isPresident, USA)$.

This is the most naive type of sampling which is rarely used in the literature. However, at train-time, it has interesting implications

Bernoulli Negative Sampling This method was proposed by Wang et al. [2014] in a paper suggesting TransH, a modification of TransE using hyperplanes to learn linearly independent vector subspaces for each relation. They describe it comprehensively as follows:

> Basically, we set different probabilities for replacing the head or tail entity when corrupting the triplet, which depends on the mapping property of the relation, i.e., one-to-many, many-to-one or many-to-many. We tend to give more chance to replacing the head entity if the relation is one-to-many and give more chance to replacing the tail entity if the relation is many- to-one. In this way, the chance of generating false negative labels is reduced. Specifically, among all the triplets of a relation r, we first get the following two statistics: (1) the average number of tail entities per head entity, denoted as tph; (2) the average number of head entities per tail entity, denoted as hpt. Then we define a Bernoulli distribution with parameter $\frac{tph}{tph+hpt}$ for sampling: given a golden triplet (h, r, t) of the relation r,

with probability $\frac{tph}{tph+hpt}$ we corrupt the triplet by replacing the head, and with probability $\frac{hpt}{tph+hpt}$ we corrupt the triplet by replacing the tail.

For example, given the triple $(DonaldTrump, wasPresident, USA)$, Bernoulli sampling would select the entity $DonaldTrump$ for replacement — because USA as a tail has more unique heads than $DonaldTrump$ as a head has unique tails. It would then replace this entity equiprobably with any other head that occurred with USA as the tail - such as $LeonardoDiCaprio$, $Chicago$, or $InternalRevenueService$.

Alternatively put, Bernoulli sampling is a way of identifying a true (h, r, t) and swapping out the r for a different r'. This r' must have has co-occurred with either h or t, one of which is chosen based on the computation described above. The key feature is that at least one true fact connecting the resulting h and t does exist in the training data.

Positional Negative Sampling

For any triple (h, r, t), the entity h or t to be replaced can only be replaced by another entity that has occurred in the training data on the same side of the relationship r. For example, given the triple $(DonaldTrump, wasPresident, USA)$, it would randomly select (say) the entity USA for replacement. It would now be constrained to replace it with other entities which have been recorded at the t position of the relation $wasPresident$ - such as $India$, NRA or $FIFA$.

7.3 Interpreting negative sampling: train-time, corruption-time and evaluation-time implications

At **corruption-time**, the negative sampler $sampler_{corrupt}$ determines how $(100 - ts)\%$ of facts in a Knowledge Graph are corrupted.

At **train-time**, the negative sampler $sampler_{train}$ determines how and how many facts are inferred to be false based on a given true fact. The train-time negative sampling is further parametrized by n, the number of negative facts inferred per true positive fact. In this paper, $n = 1$ — only 1 false fact is inferred per true one.

At **evaluation-time**, the negative sampler $sampler_{train}$ determines how to sample false facts intended to confuse the true/false discriminator.

Here I discuss the implications of the three types of sampling discussed in this paper in each step.

Uniform Negative Sampling sampling is the most naive way to sample negatives — and therefore amounts to the poorest common-sense reasoning of negatives. Uniform sampling is likely to generate nonsensical facts such as $(Chicago, wasPresident, USA)$. Here, a nonsensical fact can be qualita-

tively described as a false facts that is inconsistent with the true semantic meanings of its constituent (h, r, t).

Bernoulli Negative Sampling sampling is a slightly more context-driven way to sample negatives. Bernoulli sampling is likely to generate false facts such as $(Chicago, wasPresident, USA)$ or $(LeonardoDiCaprio, wasPresident, USA)$. Thus there is some chance that the fact is semantically coherent, but a higher chance of it being nonsensical. The h' and t' of the resulting (h', r', t') triple must necessarily have co-occurred in KG_{train} with a different r. However, it is possible (and even likely) that the new r' is a nonsensical connection between the entities.

Positional Negative Sampling is an intelligent way to infer negatives. It is strictest at preserving the semantic meaning of the replaced triple, selecting h' or t' from only those entities that have occurred in the same position with the anchoring relation r.

A Positional sampled negative fact is likely to be plausible rather than nonsensical. A 'plausible' falsehood can be qualitatively interpreted as a false fact that is consistent with the true semantic meanings of its constituent (h, r, t) - such as $(JoeBiden, isPresident, India)$.

In all types of sampling, there is an important caveat: a negatively sampled triple (h', r, t) or (h, r, t') could be true in fact, but simply excluded from the dataset. Such a triple will be treated as false under the closed world assumption. The implications of this, are discussed in Cortés-Calabuig et al. [2005].

7.4 Negative sampler configuration for KGE training

For each KGE model trained, the following configuring choices must be made:

1. How much of the training data is corrupted?
   Answer: Truth-share. The truth-share ts is the percentage of input training data that is false. These are false facts the model is told to treat as truths.
   This choice operationalizes the extent of data corruption.

2. How is the training data corrupted?
   Answer: choice of $sampler_{corrupt}$. The model learns the resulting corrupted training data as if it were all true. This choice operationalizes the sophistication of the data corruption process.

3. **At train-time i.e. during learning**, how do we choose negative facts to learn as false for each true positive fact learned to be true?
   This choice determines the strategy for 'deceptiveness' of the corrupted training data.
   Answer: choice of $sampler_{train}$. The model infers these sampled negatives as false,

explicitly adjusting its parameters in that direction.

This choice is an operationalization of the type of negative inference at train-time.

4. **At evaluation-time** (only for triplet classification), how do we choose negative facts to generate to confuse the model?

   Answer: choice of $sampler_{evaluation}$. The model is asked to predict true/false for a corpus of 50% true unseen facts, and 50% negatively sampled facts.

   This choice is an operationalization of 'deceptiveness' during evaluation.

The model is then trained on KG_{train}. Triples from KG_{val} and KG_{test} are later used for evaluation in Triplet Classification.

In this paper, we use only Positional Negative Sampling as $sampler_{corrupt}$. This is aligned with the goal of these experiments, which is to test the effects of **plausible** falsehoods in training data. In the real world, we expect these types of falsehoods to be both, more common, as well as more difficult to detect. Uniform and Bernoulli sampling are excluded because they mostly noisy and nonsensical false facts such as $(Chicago, wasPresident, USA)$. Such facts will likely confound the representations being learned for all three elements of the triple - as they would present these entities and relations in an incorrect, meaningless and nonsensical context.

Based on the choice of $sampler_{train}$, we identify 3 configurations for train-time:

1. $sampler_{train}$: Positional

   This sampler uses common sense reasoning to infer plausible negative examples at train-time.

2. $sampler_{train}$: Bernoulli

   This sampler uses relatively naive reasoning to infer noisy negative samples at train-time. Resulting triples may be nonsensical, semantically similar, or plausible (this case is least likely).

3. $sampler_{train}$: Uniform

   This sampler uses the most naive reasoning to infer negative samples. The swapped h' is picked equiprobably from all the entities $\mathcal{E}$.

Based on choice of $sampler_{evaluation}$, we also describe 2 choices for evaluation-time:

1. $sampler_{evaluation}$: Positional

   This is a more deceptive evaluator that tests discrimination between **plausible falsehoods** and true truths.

2. $sampler_{evaluation}$: Bernoulli

   This is a more naive evaluator that tests discrimination between **noisy falsehoods** and true truths.

7.5 Hyperparameters

A hyperparameter sweep is used to identify the following training hyperparameters for both TransE and DistMult: 250 epochs, Adam optimizer, learning rate of $5e-5$.

For $sampler_{train}$, we set $n = 1$. This leads to the smallest possible effect of the choice of train-time sampler.

The dimensionality of embeddings learned is set as 250 for each model.

7.6 Code

All code used in these experiments is open-source Python software developed by the author, available for reuse and replication here https://github.com/bakerwho/weboftruth. The open-source package Python *torch-kge* Boschin [2020] has been heavily borrowed from in this work.

8 Results

The train and test accuracies on classification using Logistic Regression (LR) and Multilayer Perceptron (MLP) are interpreted as usual. This evaluation task tests the utility of the 250-dimensional entity and relation representations in a truth prediction task. For both TransE and DistMult, the MLP overfits, as can be expected on a 250-dimensional input to a binary classification problem. Adding dropout or other features can reduce overfitting, but the current case offers the opportunity to compare overfit models on unseen test facts.

Surprisingly, there is little to no reduction in accuracy for the LR and MLP as the data is corrupted with up to 50% **plausible** falsehoods. This implies that the embeddings learned from corrupted but semantically coherent data are still useful features for truth prediction, a counter-intuitive result. An explanation for this is offered in Section 9.

It should be noted here that these results demonstrate the smallest possible effect of $sampler_{train}$ at train-time, with $n = 1$ — only one negative fact is sampled and inferred false per true positive. Trouillon et al. [2016] report that increasing n to a value between 50 or 200 reduces the number of epoch taken for convergence, and improves overall performance on their link prediction evaluation task. Toutanova et al. [2015] report that "Performance improved substantially when the number of negative examples was increased and reached a plateau around 200". A significant difference in performance is recorded in this work even with the smallest possible n.

8.1 Triplet Classification using TransE embeddings

Tables 4 and 5 describe the accuracy of an LR and MLP trained on TransE embeddings for binary true/false prediction.

Model	sampler		truth_share	Logistic Regression	
	train	evaluation	ts	train accuracy	test accuracy
TransE_01	Positional	Bernoulli	100	81.50%	68.09%
TransE_01		Positional	100	83.90%	68.84%
TransE_02		Bernoulli	80	81.32%	68.29%
TransE_02		Positional	80	84.20%	68.26%
TransE_03		Bernoulli	50	81.60%	68.86%
TransE_03		Positional	50	83.88%	68.94%
TransE_04	Bernoulli	Bernoulli	100	85.51%	74.95%
TransE_04		Positional	100	86.83%	72.74%
TransE_05		Bernoulli	80	85.49%	73.97%
TransE_05		Positional	80	86.77%	72.50%
TransE_06		Bernoulli	50	85.52%	74.77%
TransE_06		Positional	50	86.89%	72.79%
TransE_07	Uniform	Bernoulli	100	85.28%	75.11%
TransE_07		Positional	100	87.04%	72.49%
TransE_08		Bernoulli	80	85.28%	75.11%
TransE_08		Positional	80	87.04%	72.49%
TransE_09		Bernoulli	50	85.28%	75.11%
TransE_09		Positional	50	87.04%	72.49%

Table 4: Logistic Regression Classifier results for TransE models of different configurations

Model	sampler		truth_share	Multilayer Perceptron	
	train	evaluation	ts	train accuracy	test accuracy
TransE_01	Positional	Bernoulli	100	100.00%	79.08%
TransE_01		Positional	100	100.00%	77.46%
TransE_02		Bernoulli	80	100.00%	78.67%
TransE_02		Positional	80	100.00%	76.55%
TransE_03		Bernoulli	50	100.00%	78.42%
TransE_03		Positional	50	100.00%	77.08%
TransE_04	Bernoulli	Bernoulli	100	100.00%	91.47%
TransE_04		Positional	100	100.00%	89.02%
TransE_05		Bernoulli	80	100.00%	91.12%
TransE_05		Positional	80	100.00%	89.07%
TransE_06		Bernoulli	50	100.00%	91.49%
TransE_06		Positional	50	100.00%	89.15%
TransE_07	Uniform	Bernoulli	100	100.00%	92.27%
TransE_07		Positional	100	100.00%	90.57%
TransE_08		Bernoulli	80	100.00%	92.01%
TransE_08		Positional	80	100.00%	90.53%
TransE_09		Bernoulli	50	100.00%	91.85%
TransE_09		Positional	50	100.00%	90.44%

Table 5: Multilayer Perceptron Classifier results for TransE models of different configurations

The salient features of these results are highlighted in the visualizations below. Note the stark difference due to the choice of train-time negative sampling strategy.

Figure 5: LR train and test accuracies for classifying Bernoulli- and Positional-sampled evaluation triplets as true/false using **TransE** embeddings. The three $sampler_{train}$ choices are shown in different colors for each evaluation metric. Results for $ts \in \{100, 80, 50\}$ are plotted separately. Positional sampling consistently produces the least useful features, with a 8-15% improvement observed by shifting to Bernoulli or Uniform. Uniform sampling does better than Bernoulli by a very small margin (under 1%).

Figure 6: MLP train and test accuracies for classifying Bernoulli- and Positional-sampled evaluation triplets as true/false using **TransE** embeddings. The three $sampler_{train}$ choices are shown in different colors for each evaluation metric. Results for $ts \in \{100, 80, 50\}$ are plotted separately. Training accuracy is omitted as it is uniformly over 99.9%. Similar trends are observed as in Figure 5 - decreasing truth-share does not affect performance, MLP does better than LR, Positional is the worst sampling for this task, and Uniform and Bernoulli sampling perform well, staying within 1% of each other on test accuracy.

Surprisingly, there is little to no dip in accuracy as truth-share is decreased. As truth-share goes from 100 to 50, the average test accuracy stays within 0.4% of 72% for LR, and within 0.4% of 86.6% for MLP. This unexpected result is addressed in Section 9. MLP is able to learn more complex functions than LR on the same inputs and overfits while training. It consistently performs better than LR on the test set despite this overfitting. MLP's average test accuracy is 14.26% higher than that of

LR for 9 results using Positional sampling. This difference is 14.67% for Bernoulli sampling. At evaluation-time, Positional sampling is consistently a more difficult evaluation task than Bernoulli. With LR, the average test accuracy is 1.41% higher for Bernoulli than Positional. With MLP, this difference is 1.83%.

8.2 Triplet Classification using DistMult embeddings

Tables 6 and 7 describe the accuracy of an LR and MLP trained on DistMult embeddings for binary true/false prediction.

Model	sampler		truth_share	Logistic Regression	
	train	evaluation	ts	train accuracy	test accuracy
DistMult_01	Positional	Bernoulli	100	84.58%	73.05%
DistMult_01		Positional	100	86.04%	72.59%
DistMult_02		Bernoulli	80	84.75%	72.60%
DistMult_02		Positional	80	85.83%	72.67%
DistMult_03		Bernoulli	50	84.58%	72.52%
DistMult_03		Positional	50	86.07%	72.15%
DistMult_04	Bernoulli	Bernoulli	100	89.55%	78.89%
DistMult_04		Positional	100	89.51%	76.62%
DistMult_05		Bernoulli	80	89.47%	78.95%
DistMult_05		Positional	80	89.43%	76.82%
DistMult_06		Bernoulli	50	89.46%	78.83%
DistMult_06		Positional	50	89.65%	76.94%
DistMult_07	Uniform	Bernoulli	100	87.88%	76.95%
DistMult_07		Positional	100	88.75%	75.73%
DistMult_08		Bernoulli	80	87.88%	76.95%
DistMult_08		Positional	80	88.75%	75.73%
DistMult_09		Bernoulli	50	87.88%	76.95%
DistMult_09		Positional	50	88.75%	75.73%

Table 6: Logistic Regression Classifier results for DistMult models of different configurations.

The salient features of these results are highlighted in visualizations exactly as they were for TransE in the previous section.

Model	sampler		truth_share	Multilayer Perceptron	
	train	evaluation	ts	train accuracy	test accuracy
DistMult_01	Positional	Bernoulli	100	99.99%	85.33%
DistMult_01		Positional	100	100.00%	81.36%
DistMult_02		Bernoulli	80	100.00%	84.77%
DistMult_02		Positional	80	99.92%	81.94%
DistMult_03		Bernoulli	50	100.00%	84.33%
DistMult_03		Positional	50	99.96%	80.67%
DistMult_04	Bernoulli	Bernoulli	100	100.00%	92.72%
DistMult_04		Positional	100	100.00%	90.44%
DistMult_05		Bernoulli	80	100.00%	92.24%
DistMult_05		Positional	80	100.00%	90.09%
DistMult_06		Bernoulli	50	99.97%	92.79%
DistMult_06		Positional	50	100.00%	90.26%
DistMult_07	Uniform	Bernoulli	100	99.99%	93.02%
DistMult_07		Positional	100	100.00%	91.50%
DistMult_08		Bernoulli	80	100.00%	92.65%
DistMult_08		Positional	80	100.00%	90.83%
DistMult_09		Bernoulli	50	100.00%	93.01%
DistMult_09		Positional	50	100.00%	90.41%

Table 7: Multilayer Perceptron Classifier results for DistMult models of different configurations

Figure 7: LR train and test accuracies for classifying Bernoulli- and Positional-sampled evaluation triplets as true/false using **DistMult** embeddings. The three $sampler_{train}$ choices are shown in different colors for each evaluation metric. Results for $ts \in \{100, 80, 50\}$ are plotted separately. Unlike TransE, the test accuracies decrease (but very marginally, within 1%) as the truth share decreases. Positional sampling at train-time once again produces the worst performing embeddings. Bernoulli sampling is the most useful train-time sampler, outperforming Uniform on test accuracy by an average of 1.93%.

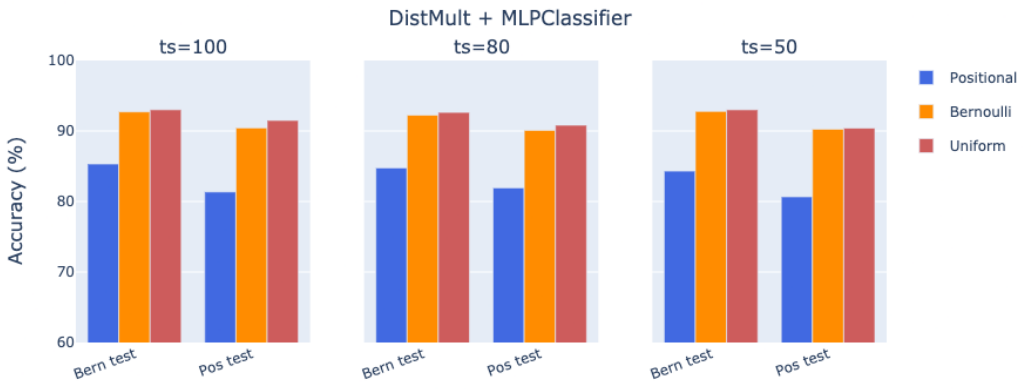

Figure 8: MLP train and test accuracies for classifying Bernoulli- and Positional-sampled evaluation triplets as true/false using **DistMult** embeddings. The three $sampler_{train}$ choices are shown in different colors for each evaluation metric. Results for $ts \in \{100, 80, 50\}$ are plotted separately. Training accuracy is omitted as it is uniformly over 99.9%. Once again, Positional sampling at train-time produces the least useful embeddings, and Uniform sampling dominates marginally over Bernoulli.

With DistMult embeddings as features for true/false classification, there is an almost negligible dip in accuracy as the truth-share ts decreases. As truth-share goes from 100 to 50, the average test accuracy stays within 0.1% of 75.7% for LR, and within 0.4% of 89% for MLP. This unexpected result is addressed in Section 9. MLP's average test accuracy is 12.5% higher than that of LR for 9 results using Positional sampling. This difference is 13.9% for Bernoulli sampling. At evaluation-time, Positional sampling is again the a more difficult task. With LR, the average test accuracy is 1.2% higher for Bernoulli than Positional. With MLP, this difference is 2.6%.

9 Discussion

KG Embeddings are useful features for truth prediction

This approach to triplet classification evaluation has not been explored before. Socher et al. [2013] use the model's in-built scoring feature to identify a threshold such that triplets above the threshold are classified as true. However, this approach ignores the potential utility of the trained vectors themselves as features for truth prediction. This paper confirms this potential with a relatively strong best performance of 93.02% (MLP on DistMult embeddings with Bernoulli evaluation) on the true/false triplet classification evaluation task.

Naive sampling at train-time is useful

Anecdotal accounts in Cai and Wang [2018] and Safavi and Koutra [2020] claim that negative sampling at train-time should be intelligent rather than naive — the rationale being that nonsensical falsehoods are unhelpful during training as they are too easy to dismiss as false. This intuitively makes sense for the goal of fine-tuned reasoning of negative knowledge. However, this paper demonstrates that 'unintelligent' or 'noisy' negative sampling techniques at train-time are of high utility for the evaluation task of truth prediction. In fact, the intelligent sampler i.e. Positional sampling does uniformly worst, with Uniform doing best in all configurations but one (i.e. DistMult + Logistic Regression). In general, naive sampling techniques appear to strengthen the utility of trained embeddings as features for truth prediction.

One possible explanation for this observation is the fact that naive sampling techniques give a strong signal to the KGE algorithm to separate semantically different objects in their respective vector spaces. Therefore, the most naive sampling stategy is in fact more likely to produce a set of embeddings where distinct entities have been given a strong negative signal. Conversely, the most intelligent sampling strategy is likely to produce a strong negative signal between semantically similar entities.

If this explanation holds, it is highly likely that a mixed strategy should lead to optimal performance — one that involves naive negative sampling at first, to spread objects out into distinct semantic zones and then a more intelligent sampling to fine-tune the differences between semantically similar objects.

The most naive train-time strategy i.e. Uniform sampling performs best in all configurations except DistMult + Logistic Regression. This anomaly, as well as the closeness in performance of Uniform and Bernoulli, should be investigated further. One way of doing this would be to increase n, as discussed later in this section.

Robustness to corrupt data

The drop in performance as the truth-share is decreased is almost negligible. This is a counter-intuitive result, as we expect the ability to distinguish between true and false to weaken as we train on more falsehoods. The explanation may lie in the power of even simple models like Logistic Regression to learn high-performing decision boundaries on semantically coherent featuresets.

The Logistic Regression is forced to learn a linear decision boundary between true and false in a 750-dimensional embedding space. Even with a truth-share as low as 50%, LR achieves 75.73% test accuracy using DistMult and 72.49% using TransE (with the Positional evaluation sampling and Uniform). MLP for the same configurations reports 90.44% and 90.41% on TransE and DistMult respectively.

Clearly, the utility of learned embeddings as features for truth prediction is unaffected by up to 50% Positional-sampled corrupt data. Positional sampling preserves the semantic roles of the elements of false triples. This implies that this preservation of meaning is sufficient to preserve the truth-predicting power of learned embeddings.

This result tells us that the Logistic Regression alone is able to learn powerful reasoning with just $750 + 1$ parameters. MLP, with 751×100 parameters, overfits to the training data, but still performs well on the test set. The strong performance validated the intuition that KGE vector dimensions can be combined into useful features for the final prediction. This assumption is aligned with the advancements in the fields of convolutional KGEs like ConvE [Dettmers et al., 2018], which learn convolutional feature maps on the embedding dimensions.

Evaluation sampling preserves intuitions of 'deceptiveness'

At evalution-time, train and test accuracies are lower for Positional sampling than Bernoulli for each KGE model, truth-share, as well as classifier choice. This validates the idea that Positional sampling is a more 'deceptive' evaluation-time sampler. The very same features (i.e. vector representations) can be more easily used for discrimination between sense and nonsense (i.e. noisy falsehoods generated by Bernoulli sampling) than between truth and plausible falsehood (i.e. strategic falsehoods generated by Positional sampling).

Feature analysis: identifying implicit semantic dimensions

One advantage of the Logistic Regression is that it allows for easy interrogation of feature importance, receive operating characteristics, and other interpretable metrics. Such work can help understand the semantic meaning implicitly encoded in learned embeddings, and better inform discussions of robustness to corrupted data. For example, the implicit 'gender' dimensions can be identified by the vector dimensions which are most useful to correctly predict the truth of the gender relation $r =$ **/people/person/gender**

Dataset dependence

This paper works with only a single dataset. Other datasets could vary along axes of structure and features, such as number and types of relationships, degree distribution, centrality, sampling and representation issues. Applying these methods to other datasets may help interrogate dataset-specific features such as 'robustness' to corruption. For example, consider relatively 'closed' dataset of geographical relations of only 2 types 'containedIn' and 'neighborOf'. We may expect such a dataset to be less robust to corruption than an extremely heterogeneous one like FB15K-237.

Increasing n may underscore the observed effect

In this paper, we use $n = 1$, i.e. only a single negative fact is inferred per positive fact at train-time. A stronger validation of the effects of train-time sampling will likely be found by running experiments with higher values of n such as 50, 100, 200 as recommended in the literature. Alternatively, the hybrid approach of naive sampling followed by intelligent sampling at later stages can be experimented with.

Local closed world assumption

10 Conclusions and Future Work

This paper demonstrates that knowledge graph embeddings are useful features for independent models to predict triplet validity. There are obvious and important use-cases for this in the fields of Knowledge Base auto-completion, recommendation systems, and question-answering. In the case of KB auto-completion, new triples can be evaluated not just by the model's in-built scoring function, but by an unbiased third-party classifier model which also allows for interpretation of the vector dimensions. In the case of recommendation systems and question-answering applications, the size and evaluation time for models can be dramatically decreased by training simple models such as Logistic Regression on trained embeddings. This is in stark contrast with the size and computational complexity of bulky convolutional KG embedding models. This space and time efficiency can allow for KGE applications to be deployed to smaller devices with less computing resources.

Another common KGE evaluation task is link prediction - the model is evaluated on its ability to complete unseen facts (h', r', t') where one of h' or t' (chosen randomly) is masked. In the case where h' is masked, the model evaluates the scores for all possible triplets $(h^+, r', t') \quad \forall \quad h^+ \in entities$ and ranks them. It is common for the KGEs to rank the correct entity in the top 10 with 70% and 90% probability. The findings in this paper indicate that the model's performance on link prediction can potentially be improved by training a simple independent model like Logistic Regression to identify the winner from amongst these 'finalists' — a task that can easily be formulated as true/false triplet classification.

The demonstration of the counterintuitive value of naive negative sampling exposes an important heuristic of KGE training — that it may be valuable to initially sample naively and then fine-tune with more intelligent sampling. This is intuitively pleasing, as we would like discriminating models to learn both types of differences — that between sense and nonsense, as well as that between true and false.

The strong performance on the feature-based truth prediction task defined in this work indicates a need to diversify and expand the suite of KGE evaluation tasks. Wang et al. [2018]'s GLUE benchmark for natural language understanding involves a suite of 9 tasks, each capturing some aspect of natural language reasoning, inference and understanding. On the leaderboard, it is common for some models to do better on some tasks while also underperforming on others, allowing a holistic evaluation of strengths and weaknesses. There is a need to develop a similarly comprehensive benchmark for KGEs. A focus on interpretability can be added by introducing visualizations and feature importance tests — these will be particularly important given the need for transparency and accountability regarding automated knowledge representations.

The ability to distinguish between unseen true and false triples, even when falsehoods are plausible, suggests applications to human-in-the-loop or machine-in-the-loop fact-checking. Fact-checking efforts by social media platforms like Facebook and Twitter are limited by restrictions on available human time and effort. Given an automatically parsed, ever-growing Knowledge Graph of events, facts, entities, and attributes, dynamic KGE models (easily retrained on a daily or weekly basis) can be adapted to this task. Truth prediction models tuned for low false positive rates (so the model rarely predicts a true fact is false) can be used to flag content for verification by human fact-checkers. On the other hand, models tuned for low false negative rates (so the model rarely predicts a false fact is true) can be used for user-facing applications that encourage users to be skeptical of low-credibility fact claims. This paper supports the use of KGEs as independent inputs to other models, allowing for a decoupled investigation of the semantic meaning of KGE dimensions and fine-tuning to the task. Specifically in the case of fact-checking, KGEs may assist the interpretability and explainability of the algorithmic process.

The fact that learned embeddings continue to be valuable truth predictors despite data corruption is both useful but also concerning. In effect, it demonstrates that a KGE can train on semantically correct but patently false information like $(BradPitt, isPresident, USA)$, and still produce vector embeddings that are useful truth predictors for other facts concerning these entities. Importantly, the vectors are fed into classifiers that are trained and tested on unseen fact combinations — effectively learning a model of 'fact claim plausibility' using entity and relation representations. This raises questions about the combined function approximation power of the KGE + Classifier pipeline used in this work. Further interrogating this process is a valuable direction for future research.

Promising directions for future work include expanding these experiments to other datasets and KGEs, developing a comprehensive set of benchmarks for KGEs, experimenting with dynamic and heterogeneous sampling strategies while training, and increasing and simultaneously investigating the utility of learned KG embeddings as input features for downstream models.

Acknowledgements

All the analysis, results, discussion presented here are the work of Aabir Abubaker Kar. That being said, this project would never have reached this stage without the contributions of my colleagues, friends and family.

The initial proof-of-concept of this project was established by Aabir Abubaker Kar and Adarsh Mathew in Spring 2020, as part of coursework for the course Advanced Machine Learning for Public Policy (CAPP 30255). Prof. Amitabh Chaudhary offered valuable feedback during that time.

Further work was done as part of project coursework for Unsupervised Machine Learning (TTIC 31220) in Winter 2021, where Elena Orlova and Dennis Zheng made valuable contributions. Instructor Prof. Karen Livescu and TA Freda Shi were generous with their time and feedback.

Nak Won Rim assisted with a social-science expansion of this work as part of the course Thinking with Deep Learning (MACS 37000) under Prof. James Evans.

Regina Catipon — thank you for inspiring me to translate technologies of intelligence into accessible and helpful augmentations for the everyperson.

My dear friend, colleague, mentor, and constant inspiration, Bhargav Srinivasa Desikan, who inspired me to come to Chicago to study computational social science — thank you for venturing with me through the wildest of musings as young, gritty scientists unsatisfied with the way of things.

Teachers at the University of Chicago were unexpectedly successful at having me learn many new things, for which I am extremely thankful.

Prof. Michael Fisch from the Anthropology department helped me think through the technicity of the formidable yet potent technology that is Knowledge Graph Embeddings. Alexander Campolo was forged radical rethinkings of the connections between data and knowledge. Danielle DuMerer helped interpret algorithmic misinformation through the lens of public policy, motivating advocacy for real-world change.

Prof. David McAllister re-taught me deep learning in such a way that it will stay with me forever. Prof. Kevin Gimpel's coursework and feedback taught me advanced natural language processing skills that gave me the confidence to approach this massive problem from a deep learning angle. Prof. Chenhao Tan gave me much-needed feedback that helped disentangle my previously muddled approach to the topic.

As my faculty advisor, Prof. James Evans offered formative guidance and inspiration through multiple iterations of the experimental design and interpretation. His advice and feedback have been crucial

to shaping this research as an interdisciplinary project. Prof. Evans' enthusiasm and excellence in conducting exciting research are a continuing source of inspiration and motivation.

Sanja Miklin's generous time and support as a preceptor were tremendously appreciated. Every minor or major word of feedback, every assurance from her has counted in making this project what it is.

Finally, I bear deep gratitude for my family and friends, here in Chicago, back home in India, and everywhere else in the world from where their energy and support have fuelled my endeavors.

For love, curiosity, and the strength to carry on, I thank Aashya, Mum, Dad, Dadulay, Bhargav, Aastha, Ridima, Samhitha, Srinidhi, Regina.

Research contributions

- **Aabir Abubaker Kar**: project leader and research author - primary contributor for experiment design, literature review, software engineer for `weboftruth`package, machine learning engineer for all experiments in this work, model training, evaluation, results, and analysis
- **Adarsh Mathew**: early experimental design and discussion as part of coursework for CAPP 30255
- **Elena Orlova**: training models and running experiments using `weboftruth`package (none of these results were presented here) as part of coursework for TTIC 31220
- **Dennis Zheng**: contributions to Methods section as part of coursework for TTIC 31220
- **Nak Won Rim**: literature review, experimental design discussion (those experiments were not presented here) as part of coursework for MACS 37000

References

Unnamed author. How Google's Knowledge Graph works, 2021. URL `https://support.google.com/knowledgepanel/answer/9787176?hl=en#`.

Bence Bago, David G. Rand, and Gordon Pennycook. Fake news, fast and slow: Deliberation reduces belief in false (but not true) news headlines. *Journal of Experimental Psychology: General*, 149 (8):1608–1613, August 2020. ISSN 1939-2222, 0096-3445. doi: 10.1037/xge0000729. URL `http://doi.apa.org/getdoi.cfm?doi=10.1037/xge0000729`.

Rishabh Bhardwaj, Navonil Majumder, and Soujanya Poria. Investigating Gender Bias in BERT. *Cognitive Computation*, May 2021. ISSN 1866-9956, 1866-9964. doi: 10.1007/s12559-021-09881-2. URL `https://link.springer.com/10.1007/s12559-021-09881-2`.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, page 1247–1250, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605581026. doi: 10.1145/1376616.1376746. URL https://doi.org/10.1145/1376616.1376746.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 2787–2795, Red Hook, NY, USA, 2013. Curran Associates Inc.

Armand Boschin. TorchKGE: Knowledge Graph Embedding in Python and PyTorch. *arXiv:2009.02963 [cs]*, September 2020. URL http://arxiv.org/abs/2009.02963. arXiv: 2009.02963.

Liwei Cai and William Yang Wang. KBGAN: Adversarial Learning for Knowledge Graph Embeddings. *arXiv:1711.04071 [cs]*, April 2018. URL http://arxiv.org/abs/1711.04071. arXiv: 1711.04071.

Aylin Caliskan, Joanna J. Bryson, and Arvind Narayanan. Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334):183–186, April 2017. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.aal4230. URL https://www.sciencemag.org/lookup/doi/10.1126/science.aal4230.

Yimin Chen, Niall J. Conroy, and Victoria L. Rubin. Misleading Online Content: Recognizing Clickbait as "False News". In *Proceedings of the 2015 ACM on Workshop on Multimodal Deception Detection*, pages 15–19, Seattle Washington USA, November 2015. ACM. ISBN 978-1-4503-3987-2. doi: 10.1145/2823465.2823467. URL https://dl.acm.org/doi/10.1145/2823465.2823467.

Alvaro Cortés-Calabuig, Marc Denecker, Ofer Arieli, Bert Van Nuffelen, and Maurice Bruynooghe. On the local closed-world assumption of data-sources. In Chitta Baral, Gianluigi Greco, Nicola Leone, and Giorgio Terracina, editors, *Logic Programming and Nonmonotonic Reasoning*, pages 145–157, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. ISBN 978-3-540-31827-9.

Yuanfei Dai, Shiping Wang, Neal N. Xiong, and Wenzhong Guo. A Survey on Knowledge Graph Embedding: Approaches, Applications and Benchmarks. *Electronics*, 9(5):750, May 2020. ISSN 2079-9292. doi: 10.3390/electronics9050750. URL https://www.mdpi.com/2079-9292/9/5/750.

Tim Dettmers, Minervini Pasquale, Stenetorp Pontus, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *Proceedings of the 32th AAAI Conference on Artificial Intelligence*, pages 1811–1818, February 2018. URL https://arxiv.org/abs/1707.01476.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805 [cs]*, May 2019. URL http://arxiv.org/abs/1810.04805. arXiv: 1810.04805.

Joseph Fisher, Dave Palfrey, Christos Christodoulopoulos, and Arpit Mittal. Measuring Social Bias in Knowledge Graph Embeddings. *arXiv:1912.02761 [cs]*, May 2020. URL http://arxiv.org/abs/1912.02761. arXiv: 1912.02761.

Aditya Grover and Jure Leskovec. node2vec: Scalable Feature Learning for Networks. *arXiv:1607.00653 [cs, stat]*, July 2016. URL http://arxiv.org/abs/1607.00653. arXiv: 1607.00653.

Sarah J. Jackson and Brooke Foucault Welles. Hijacking #myNYPD: Social Media Dissent and Networked Counterpublics: Hijacking #myNYPD. *Journal of Communication*, 65(6):932–952, December 2015. ISSN 00219916. doi: 10.1111/jcom.12185. URL https://academic.oup.com/joc/article/65/6/932-952/4082320.

Matthias Jarke, Bernd Neumann, Yannis Vassiliou, and Wolfgang Wahlster. KBMS Requirements of Knowledge-Based Systems. In Michael L. Brodie, John Mylopoulos, Joachim W. Schmidt, Joachim W. Schmidt, and Constantino Thanos, editors, *Foundations of Knowledge Base Management*, pages 381–394. Springer Berlin Heidelberg, Berlin, Heidelberg, 1989. ISBN 978-3-642-83399-1 978-3-642-83397-7. doi: 10.1007/978-3-642-83397-7_17. URL http://link.springer.com/10.1007/978-3-642-83397-7_17. Series Title: Topics in Information Systems.

Rodolphe Jenatton, Nicolas Roux, Antoine Bordes, and Guillaume R Obozinski. A latent factor model for highly multi-relational data. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper/2012/file/0a1bf96b7165e962e90cb14648c9462d-Paper.pdf.

Seyed Mehran Kazemi and David Poole. Simple embedding for link prediction in knowledge graphs. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 4289–4300, Red Hook, NY, USA, 2018. Curran Associates Inc.

Prodromos Kolyvakis, Alexandros Kalousis, and Dimitris Kiritsis. Hyperbolic knowledge graph embeddings for knowledge base completion. In Andreas Harth, Sabrina Kirrane, Axel-Cyrille Ngonga Ngomo, Heiko Paulheim, Anisa Rula, Anna Lisa Gentile, Peter Haase, and Michael Cochez, editors, *The Semantic Web*, pages 199–214, Cham, 2020. Springer International Publishing. ISBN 978-3-030-49461-2.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 3111–3119, Red Hook, NY, USA, 2013a. Curran Associates Inc.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013b. URL https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf.

Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Phung. A Novel Embedding Model for Knowledge Base Completion Based on Convolutional Neural Network. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 327–333, 2018. doi: 10.18653/v1/N18-2053. URL http://arxiv.org/abs/1712.02121. arXiv: 1712.02121.

Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. Holographic Embeddings of Knowledge Graphs. *arXiv:1510.04935 [cs, stat]*, December 2015. URL http://arxiv.org/abs/1510.04935. arXiv: 1510.04935.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A Review of Relational Machine Learning for Knowledge Graphs. *Proceedings of the IEEE*, 104(1):11–33, January 2016. ISSN 0018-9219, 1558-2256. doi: 10.1109/JPROC.2015.2483592. URL http://arxiv.org/abs/1503.00759. arXiv: 1503.00759.

Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL https://aclanthology.org/D14-1162.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv:1802.05365 [cs]*, March 2018. URL http://arxiv.org/abs/1802.05365. arXiv: 1802.05365.

Navid Rekabsaz, Simone Kopeinik, and Markus Schedl. Societal Biases in Retrieved Contents: Measurement Framework and Adversarial Mitigation for BERT Rankers. *arXiv:2104.13640 [cs]*, May 2021. doi: 10.1145/3404835.3462949. URL http://arxiv.org/abs/2104.13640. arXiv: 2104.13640.

Tara Safavi and Danai Koutra. Generating Negative Commonsense Knowledge. *arXiv preprint arXiv:2011.07497*, page 10, November 2020.

Richard M. Shiffrin, Danielle S. Bassett, Nikolaus Kriegeskorte, and Joshua B. Tenenbaum. The brain produces mind by modeling. *Proceedings of the National Academy of Sciences*, 117(47): 29299–29301, November 2020. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.1912340117. URL http://www.pnas.org/lookup/doi/10.1073/pnas.1912340117.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning With Neural Tensor Networks for Knowledge Base Completion. *Advances in Neural Information Processing Systems*, page 10, January 2013.

Dominic Spohr. Fake news and ideological polarization: Filter bubbles and selective exposure on social media. *Business Information Review*, 34(3):150–160, September 2017. ISSN 0266-3821, 1741-6450. doi: 10.1177/0266382117722446. URL http://journals.sagepub.com/doi/10.1177/0266382117722446.

Gabriel Stanovsky, Noah A. Smith, and Luke Zettlemoyer. Evaluating Gender Bias in Machine Translation. *arXiv:1906.00591 [cs]*, June 2019. URL http://arxiv.org/abs/1906.00591. arXiv: 1906.00591.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018. ISBN 0262039249.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. Representing Text for Joint Embedding of Text and Knowledge Bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509, Lisbon, Portugal, 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1174. URL http://aclweb.org/anthology/D15-1174.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *International Conference on Machine Learning (ICML)*, volume 48, pages 2071–2080, 2016.

Joshua A Tucker, Andrew Guess, Pablo Barberá, Cristian Vaccari, Alexandra Siegel, Sergey Sanovich, Denis Stukal, and Brendan Nyhan. Social media, political polarization, and political disinformation: A review of the scientific literature. *Political polarization, and political disinformation: a review of the scientific literature (March 19, 2018)*, 2018.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium, 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5446. URL http://aclweb.org/anthology/W18-5446.

YueQun Wang, LiYan Dong, XiaoQuan Jiang, XinTao Ma, YongLi Li, and Hao Zhang. KG2Vec: A node2vec-based vectorization model for knowledge graph. *PLOS ONE*, 16(3):e0248552, March 2021. ISSN 1932-6203. doi: 10.1371/journal.pone.0248552. URL https://dx.plos.org/10.1371/journal.pone.0248552.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, page 1112–1119. AAAI Press, 2014.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. *arXiv:1412.6575 [cs]*, August 2015. URL http://arxiv.org/abs/1412.6575. arXiv: 1412.6575.

Seunghak Yu, Tianxing He, and James Glass. AutoKG: Constructing Virtual Knowledge Graphs from Unstructured Documents for Question Answering. *arXiv:2008.08995 [cs]*, March 2021. URL http://arxiv.org/abs/2008.08995. arXiv: 2008.08995.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. Gender Bias in Coreference Resolution: Evaluation and Debiasing Methods. *arXiv:1804.06876 [cs]*, April 2018. URL http://arxiv.org/abs/1804.06876. arXiv: 1804.06876.

Appendix

A Algorithmic and human 'world models'

Shiffrin et al. [2020] describe a growing body of research that views the human 'mind' as comprising a model formed by the brain.

> This ... (is) a model of the entire environment, including the self, the body, the physical environment, other agents, and the social environment. Furthermore, the model would be a best guess about the most likely state of this environment. It uses this model to learn, decide, attend, remember, perceive, predict, and produce action. This model develops as the brain matures, rapidly during infancy and more slowly later. It has structural components that remain stable over long times. It has labile elements that change at multiple time scales, adapting to the current environment and goals. The mind's formation through modeling of the world might be likened to the way scientists build models: through a combination of experiment (interaction with the world) and theory (thought).

Extending this understanding, as human beings interact in the world, they also parse information from their experiences, identifying key features from their sensory inputs. Parts of these features are used to determine 'facts' i.e. trusted descriptions of relationships between sets of entities. These facts could be directly transcribed (such as by reading a newspaper headline or hearing a statement from a friend), as well as inferred by human information processing (using logical reasoning, fallacy, bias, etc.).

The opening lines of Sutton and Barto [2018]'s authoritative text on reinforcement learning are as follows:

> The idea that we learn by interacting with our environment is probably the first to occur to us when we think about the nature of learning. When an infant plays, waves its arms, or looks about, it has no explicit teacher, but it does have a direct sensorimotor connection to its environment. Exercising this connection produces a wealth of information about cause and effect, about the consequences of actions, and about what to do in order to achieve goals. Throughout our lives, such interactions are undoubtedly a major source of knowledge about our environment and ourselves.

Thus, the human response to new parsed facts about the world can be described as constantly updating a 'world model'. This world model is able to respond to new facts, judge their plausibility and context, and update the world model in response to them.

An algorithmic analogue to the human's world model can be explored using Knowledge Graphs and Knowledge Graph Embeddings. A Knowledge Graph is analogous to perfect memory - it is able to store all information that was ever received. The cost of this is that retrieval and processing become computationally/biologically expensive, and therefore practically infeasible. A Knowledge Graph Embedding model is a low-dimensional approximation of this complete graph, designed by the variable guiding principle of 'retaining valuable information' - it is the difference between what is taught and what is remembered. In this way, the training process embodies the process of learning.

B Preliminary experiments

B.1 Preliminary Dataset: Subject-Verb-Object Wikipedia corpus

In early experiments, I used a Structured SVO corpus of triples (subject, verb, direct object) from Wikipedia Jenatton et al. [2012]. It consists of 1.3M triples, with 30K entities and 4.5K relation types. These results are presented for their value in developing guiding intuitions about negative sampling.

B.2 Preliminary Results: Triplet Classification SVO Wikipedia corpus

The results of preliminary experiments on the SVO corpus are presented in Table. 8. We utilized GloVe embeddings of dimensions 50 and 100 as a baseline.

A logistic regression was used for classification. The configuration used was **Train-time sampler**: Positional, **Corruption sampler**: Bernoulli.

Importantly, $sampler_{evaluation}$ here is the Bernoulli sampler. The relatively strong performance on this evaluation task was indicative of the need to modify evaluation sampling. It indicated that the evaluation task was too easy - it was effectively forcing the classifier to distinguish between plausible and nonsensical facts, rather than true and false facts. This is consistent with later results discussed above.

GloVe embeddings are merely noise for this classification task - as 50% is the accuracy expected with random choice or singleton predictions. GloVe relies on the distributional hypothesis - that words that are used in the same contexts have similar vectors. It lacks explicit information on relationships between words. For every $O(1)$ meaningful relationship, there are $O(n)$ bogus ones. We therefore

Model	ts (%)	Val acc. (%)	Test acc.(%)
GloVe baseline (d=50)	100	52.77%	52.82%
GloVe baseline (d=100)	100	53.66	53.49
TransE (d=250)	100	77.85	77.64
TransE (d=250)	80	75.53	75.50
TransE (d=250)	50	58.74	58.75
DistMult (d=250)	100	77.51	77.29
DistMult (d=250)	80	76.07	76.05
DistMult (d=250)	50	74.15	74.17
HolE (d=250)	100	77.90	77.95
HolE (d=250)	80	75.39	75.23
HolE (d=250)	50	73.27	72.61

Table 8: Results of Logistic Regression on binary (true/false) classification of validation and test facts for different KGE models, training truth-shares, and embedding dimensionalities

expect GloVe embeddings to lose any informational structure of a fact claim, explaining the poor result. GloVe's poor performance is intuitive and expected.

All three KGE models show promising performance. As seen, there is a marginal drop in performance for TransE as the training dataset is corrupted. DistMult and HolE are more robust to a lower truth-share compared to TransE. With TransE, the test accuracy drops from 77.6% to 58.8% as ts goes from 100% to 50%.

For HolE, the corresponding drop is less than 5 percentage points, from 78% to 72.6%. For DistMult, the drop is just 3.1 percentage points , going from 77.3% to 74.2%

C Triplet Classification: Ridge Regression results for TransE and DistMult

Model	sampler		truth_share	Ridge Regression	
	train	evaluation	ts	train accuracy	test accuracy
TransE_01	Positional	Bernoulli	100	55.04%	54.25%
TransE_01		Positional	100	56.76%	55.47%
TransE_02		Bernoulli	80	55.06%	54.87%
TransE_02		Positional	80	57.18%	55.67%
TransE_03		Bernoulli	50	54.82%	54.52%
TransE_03		Positional	50	56.96%	55.23%
TransE_04	Bernoulli	Bernoulli	100	56.10%	55.00%
TransE_04		Positional	100	58.26%	56.18%
TransE_05		Bernoulli	80	56.32%	55.18%
TransE_05		Positional	80	58.41%	56.25%
TransE_06		Bernoulli	50	56.46%	55.00%
TransE_06		Positional	50	58.35%	55.63%
TransE_07	Uniform	Bernoulli	100	56.12%	55.49%
TransE_07		Positional	100	58.37%	56.64%
TransE_08		Bernoulli	80	56.12%	55.49%
TransE_08		Positional	80	58.37%	56.64%
TransE_09		Bernoulli	50	56.12%	55.49%
TransE_09		Positional	50	58.37%	56.64%

Table 9: Ridge Classifier results for TransE models of different configurations

Model	sampler		truth_share	Ridge Regression	
	train	evaluation	ts	train accuracy	test accuracy
DistMult_01	Positional	Bernoulli	100	56.67%	56.15%
DistMult_01		Positional	100	58.76%	57.05%
DistMult_02		Bernoulli	80	56.83%	56.09%
DistMult_02		Positional	80	59.03%	56.98%
DistMult_03		Bernoulli	50	56.83%	56.14%
DistMult_03		Positional	50	59.01%	56.67%
DistMult_04	Bernoulli	Bernoulli	100	58.39%	56.92%
DistMult_04		Positional	100	60.00%	57.57%
DistMult_05		Bernoulli	80	58.61%	56.69%
DistMult_05		Positional	80	60.02%	57.52%
DistMult_06		Bernoulli	50	58.49%	56.98%
DistMult_06		Positional	50	59.86%	57.71%
DistMult_07	Uniform	Bernoulli	100	58.07%	56.21%
DistMult_07		Positional	100	59.52%	56.77%
DistMult_08		Bernoulli	80	58.07%	56.21%
DistMult_08		Positional	80	59.52%	56.77%
DistMult_09		Bernoulli	50	58.07%	56.21%
DistMult_09		Positional	50	59.52%	56.77%

Table 10: Ridge Classifier results for DistMult models of different configurations

D Datasets for future work

The following is a summary of other popularly used datasets in KGE research. Many of these are suitable for running the experiments described above, the results of which can inform a deeper understanding of the effects of corruption and common-sense reasoning on KGE performance.

- WN18

  WordNet 18 is a dataset of 41K 'word' entities, and 18 types of relationships (such as hypernym, hyponym and so on). This is a widely used benchmark dataset.

- WN18RR

  WordNet 18-RR is a reduced version of WN18 - consisting of 41K 'word' entities, but just 11 types of relationships. It was designed to mitigate the problem of reversible relations identified by Toutanova et al. [2015] - analogously to how FB15k was reduced to FB15k-237.

- YAGO3-10

  With 120K entities and 37 relation types, YAGO3 stores over 1M triples that associate mostly nouns based on relationships in the domains of profession, location, and much more.

- KINSHIP

  This dataset of 104 entities has 25 possible kinship relationships and more than 10K edges.