A Further details on WD50K

In contrast with Freebase which is no longer supported nor updated, we choose Wikidata as the source KG for our dataset since it has an active community and has seen contributions from various companies that merge their knowledge with it. Additionally, many new NLP tasks (Xiong et al., 2020; Hayashi et al., 2019; Chakraborty et al., 2019), as well as datasets (Wang et al., 2019b; Mesquita et al., 2019; Dubey et al., 2019), are using Wikidata as a reference KG.

The combined statistics of our dataset are presented in Table 1. WD50k consists of 47,156 entities, and 532 relations, amongst which 5,460 entities and 45 relations are found only within qualifier (q_p, q_e) pairs. Fig. 5 illustrates how qualifiers are distributed among statements, i.e., 236,393 statements (99.9%) contain up to five qualifiers whereas remaining 114 statements in a long tail contain up to 20 qualifiers. Fig. 6 illustrates the in-degree distribution (with 50 bins, values higher than 1000 are omitted) of the WD50K graph structure where most of the nodes have in-degrees up to 200.

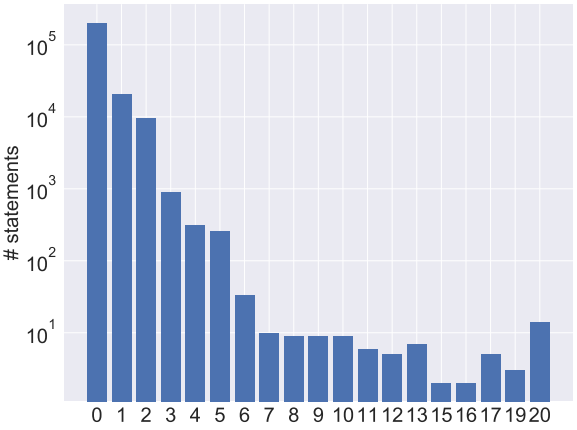

Figure 5: Number of qualifiers per statement

Recall that we augmented our dataset to reduce test set leakage by removing all instances from the train, and validation sets whose main triple (s, p, o) can be found in the test instances (Sec. 5). Another form of test leakage, as discovered in (Toutanova and Chen, 2015), may still persist in our dataset. To estimate this, we count the instances in the test set whose main triple's "direct" inverse (o, p, s), or "semantic" inverse (based on the relation P1696 in Wikidata, i.e., inverse of) is present in the train set. This amounts to less than 4% (1.6k out of 46k) instances in the test set.

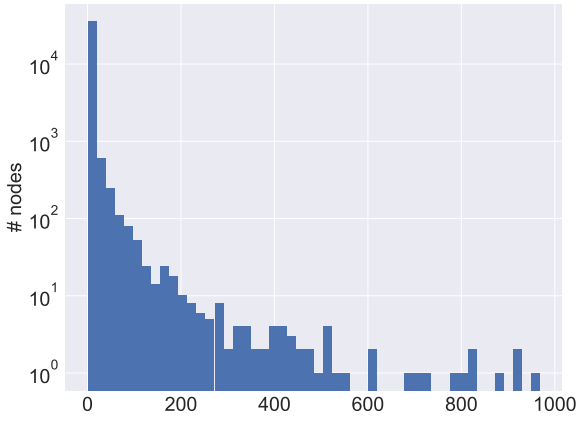

Figure 6: In-degree distribution

B Sparse Representation

Sparse Triple Representation

s	Q937	...	...
o	Q206702	...	...
r	P69	...	...
index	*k*	*k+1*	*k+2*

Sparse Qualifier Representation

index	*k*	*k*	...
qr	P812	P512	...
qv	Q413	Q849697	...

Figure 7: Sparse representation for hyper-relational facts. Each fact has a unique integer index k which is shared between two COO matrices, i.e., the first one is for main triples, the second one is for qualifiers. Qualifiers that belong to the same fact share the index k.

Storing full adjacency matrices of large KGs is impractical due to $O(|\mathcal{V}|^2)$ memory consumption. GNNs encourage using sparse matrix representations and adopting sparse matrices is shown (Cohen et al., 2020) to be scalable to graphs with millions of edges. As illustrated in Figure 7, we employ two sparse COO matrices to model hyper-relational KGs. The first COO matrix is of a standard format with rows containing indices of subjects, objects, and relations associated with the *main triple* of a hyper-relational fact.

In addition, we store index k that uniquely identifies each fact. The second COO matrix contains rows of qualifier relations qr and entities qe that are connected to their main triple (and the overall hyper-relational fact) through the index k, i.e., if a fact has several qualifiers those columns corresponding to the qualifiers of the fact will share the same index k. The overall memory consumption

is therefore $O(|\mathcal{E}| + |\mathcal{Q}|)$ and scales linearly to the total number of qualifiers $|\mathcal{Q}|$. Given that most open-domain KGs rarely qualify each fact, e.g., as of August 2019, out of 734M Wikidata statements approximately 128M (17.4%) have at least one qualifier, this sparse qualifier representation saves limited GPU memory.

C Hyperparameters

We tuned the model (STARE encoder with Transformer decoder) on the validation set using the hyperparameters reported in Table 5. Implementations of mult, ccorr, and rotate functions in ϕ_q and ϕ_r correspond to DistMult (Yang et al., 2015), circular correlation (Nickel et al., 2016), and RotatE (Sun et al., 2019), respectively.

Table 5: This table reports the major hyperparameters of our approach, and their corresponding bounds. Note that "Trf" refers to Transformers. Selected values are in **bold**.

Parameter	Value
STARE layers	{1, **2**}
Embedding dim	{100, **200**}
Batch size	{**128**, 256, 512}
Learning rate	{**0.0001**, 0.0005, 0.001}
ϕ_q	mult, ccorr, **rotate**
ϕ_r	mult, ccorr, **rotate**
γ	**weighted sum** concat, mul
Weighted sum α	$[0.0, 1.0]$ step 0.1
Quals aggregation	**sum**, mean
Trf layers	{1, **2**}
Trf hidden dim	{256, **512**, 768}
Trf heads	{2, **4**}
StarE dropout	{0.1, 0.2, **0.3**}
Trf dropout	{**0.1**, 0.2, 0.3}
Label smoothing	{0.0, **0.1**}

The selected hyperparameters include two STARE layers, embedding dimension of 200, batch size of 128, Adam optimizer with 0.0001 learning rate and 0.1 label smoothing. ϕ_r and ϕ_q are rotate functions, $\gamma(\cdot)$ is a weighted sum function with α of 0.8, qualifiers are aggregated using a simple summation, and 0.3 dropout rate. We use 2-layer Transformer block with the hidden dimension of 512, and 4 attention heads with 0.1 dropout rate as our decoder. For WD50K and JF17K datasets we set the maximum length of a hyper-relational fact to 15 (i.e., a statement can contain at most 6 qualifier pairs), and 7 for WikiPeople.

Infrastructure and Parameters. We train all models on one Tesla V100 GPU. Due to a large number of parameters, owing to large trainable embedding matrices, it is advisable to a GPU with at least 12GB of VRAM. Running STARE (H) + Transformer (H) models with the selected hyperparams on WD50K requires approximately 2 days to train and has 10.8M parameters[9]; on JF17k the model has 7.1M parameters and takes about 10 hours to train; on WikiPeople the model has 8.2M parameters which we run for 500 epochs and takes about 4 days.

StarE (H) + Transformer (H) models on reduced datasets: the model corresponding to WD50K (33) has 9M parameters and takes 20 hours to train while WD50K model has 6.8M parameters and takes about 9 hours to train. In case of WD50K (100), the model has 5M parameters and takes 5 hours to train.

D Decoders

As an additional experiment, we pair STARE with different decoders and evaluate them over WD50K datasets. Along with the main reported model denoted as **StarE + Trf**, we implemented two CNN-based decoders and another Transformer-based decoder. All models are trained with the same encoder hyperparameters as chosen in the main reported model.

StarE + ConvE relies on the ConvE (Dettmers et al., 2018)-like decoder but expanded for statements with qualifiers. Given a query *(s, r,* $\{(qr_i, qv_i)$, ... $\}$*)*, we stack entities and relations embeddings row-wise and reshape the tensor into an *image* of size $H \times W$. For instance, for a statement with 6 qualifier pairs, i.e., query length of 14, and an embedding size of 200, we obtain *images* of size 40×70. We then apply a 2D convolutional layer with a 7×7 kernel for each image, apply ReLU, flatten the resulting tensor, and pass it through a fully-connected layer. We used 200 filters and the learning rate was set to 0.001.

StarE + ConvKB is based on the ConvKB (Nguyen et al., 2018)-like decoder adjusted for statements with qualifiers. Given a query *(s, r,* $\{(qr_i, qv_i)$, ... $\}$*)*, we stack entities and relations embeddings row-wise and apply a 2D convolutional layer with a $L_Q \times 7$ kernel, e.g., for queries of length 14 the kernel size is 14×7. We then apply ReLU, flatten the resulting tensor, and

[9] According to a built-in PyTorch counter.

Table 6: Effect of different decoders on the link prediction task over WD50K, and its variations.

Dataset →	WD50K			WD50K (33)			WD50K (66)			WD50K (100)		
Method ↓	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10
STARE + Trf	**0.349**	**0.271**	**0.496**	**0.331**	**0.268**	0.451	**0.481**	**0.420**	0.594	**0.654**	**0.588**	**0.777**
STARE + ConvE	0.341	0.260	**0.496**	0.323	0.254	**0.456**	0.460	0.392	0.590	0.627	0.550	0.772
STARE + ConvKB	0.323	0.241	0.479	0.316	0.247	0.448	0.448	0.377	0.584	0.621	0.544	0.763
STARE + MskTrf	0.341	0.262	0.489	0.324	0.260	0.446	0.479	0.417	**0.595**	0.649	0.579	0.774

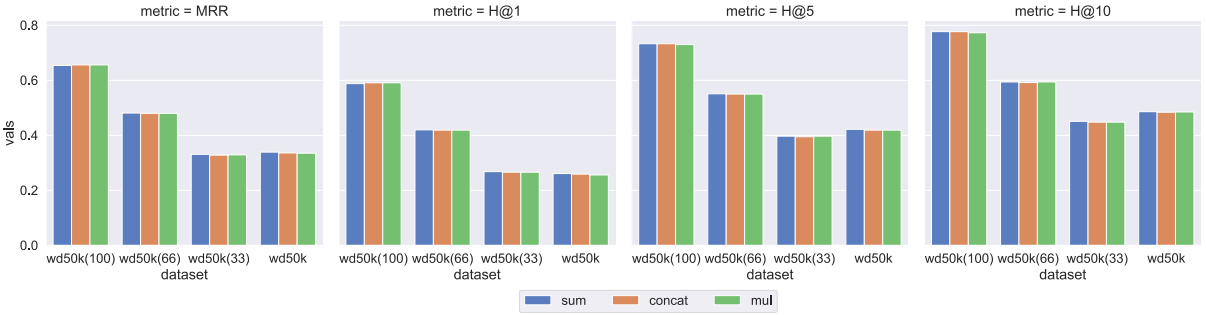

Figure 8: Gamma experiment.

pass it through a fully-connected layer. We used 200 filtersand the learning rate was set to 0.001.

StarE + MskTrf denotes a Transformer decoder with an explicit `[MASK]` token at the object position of each query. Given a query *(s, r,* $\{(qr_i, qv_i), \ldots\}$*)*, we extract relevant entities and relation embeddings and insert the `[MASK]` token, transforming it into *(s, r, [MASK],* $\{(qr_i, qv_i), \ldots\}$*)*. We then pass it through the Transformer layers and retrieve the representation of the `[MASK]` token. Finally, the token representation is passed through a fully-connected layer. We trained the model with 0.0001 as the learning rate.

Table 6 reports link prediction results on a variety of WD50K datasets with with different decoders. The default **StarE + Trf** decoder generally attains superior results with biggest gains along H@1 metric.

E Relation-Qualifiers Aggregation

In this experiment, we measure the impact of the choice of $\gamma(\cdot)$ function which is used for aggregating representations of a relation and its qualifiers (see Eq. 5). To evaluate its impact we use STARE (H) + Transformer (H) models, on four WD50K datasets using three functions, i.e., concatenation $[\mathbf{h}_r, \mathbf{h}_q]$, element-wise multiplication $\mathbf{h}_r \odot \mathbf{h}_q$, and weighted sum $\alpha \odot \mathbf{h}_r + (1 - \alpha) \odot \mathbf{h}_q$ where α is fixed to 0.8.

The results are presented in Fig.8. We find that all the three settings have similar performance indicating model's stability with respect to the choice of $\gamma(\cdot)$ function.