An Exploratory Study of the Impact of Graph Embeddings on CommonSense Conversational Generation with KGs

Bobby Yilun Hua / yh3228
Christopher Benka / cjb2261
Manuel Jenkin Jerome / mj2984

Matthew Toles / mt3639
Nikhil Balwani / nb3096

Abstract

Prior work demonstrated knowledge graphs' positive impact on natural language understanding and generation in dialog systems. The respective models of these works primarily apply a singular type of graph embedding. However, many methods for embedding graphs exist, with differing representational capabilities and computational efficiency. This work examines the effect of graph embedding choice on natural language generation performance. In particular, we retrain an open-domain, social conversation generation model, Commonsense knowledge-aware Conversational Model (CCM), using four different graph embedding techniques. To our knowledge, this is the first attempt to analyze how the choice of knowledge graph embedding schemes impact knowledge-aware generation performance in dialog systems. Our experiments show TransR, Rescal, and DistMult improve on TransE in terms of entity score, but at the expense of perplexity. We also find a benefit from augmenting the knowledge graph with symmetric and inverse relations under certain circumstances.

1 Introduction

Smoothly introducing new entities into the conversation can be a difficult task even for large language models. Much of human knowledge grounded in common sense is only assumed but not explicitly stated, making it difficult to for language models to learn without grounding. Additionally, conversational models can be prone to repetition, evasiveness, and generic answers in order to optimize for response likelihood, but at the expense of originality and engagingness (Li et al., 2015). Especially in socialbot models and unstructured dialogues, engagement is often curtailed by overly frequent or infrequent topical changes. Therefore significant work has been put into creating interesting conversation without non-sequiturs (Dinan et al., 2018; Gopalakrishnan et al., 2019), attempting strategies like modifying the model's loss function or training data, or incorporating human feedback.

Embedding	Symm	Anti	Inv	Comp
TransE	-	✓	✓	-
TransR	-	✓	✓	✓
RESCAL	✓	-	✓	✓
DistMult	✓	-	-	-

Table 1: Different KG embedding methods and types of relations they capture.

Conversational agents that optimize for maximally likely responses often tend to generate generic responses that, though likely, do not introduce new information or invitations for the user to contribute to the conversation (Kumar and Ali, 2020). Furthermore, without common sense grounding and background knowledge contextualizing conversational turns with topic changes, even training a model for the topical change dialogue subtask can be difficult. Therefore it makes sense to ground dialogue systems in commonsense knowledge, as done by Long et al. (2017) for example. By exploring common sense knowledge graphs, language models can improve their ability to transition smoothly between related topics, introduce the new subject matter, and avoid non-sequiturs.

Different embeddings of these knowledge graphs, however, have differing capabilities and may perform better or worse depending on the circumstances of the datasets and conversational tasks. Specifically, they may vary in their abilities to capture symmetric/antisymmetric relations, inversions, and compositions as shown in Table 1.

In this paper, we present results from our experiments that compared several embedding schemes and their impact on an archetypical dialogue generation model, the Commonsense Knowledge

Aware Conversational Model (CCM) by Zhou et al. (2018), in terms of perplexity and entity overlap. We also describe a method for augmenting knowledge graphs to improve their alignment with specific embedding schemes. We hope these results can be used to better inform future research on knowledge-grounded dialogue systems and expedite the selection of embedding schemes according to how their capabilities align with the task at hand. In short, our contributions include:

- Experiments comparing four promising knowledge graph embedding techniques for knowledge-aware conversations
- An augmentation method to improve knowledge graph representations for specific embedding techniques and its effect on entity scores.
- An analysis of the trade-off between entity scores and perplexity scores.
- An efficient re-implementation of the Commonsense Knowledge Aware Conversation Model (CCM) in PyTorch that runs faster and supports complex-valued graph embeddings.

2 Related Work

The impact of the effect of knowledge graph embedding schemes has been studied across other fields, though less so in the domain of conversational AI. For example, Chang et al. (2020) found that RotateE outperforms other common embedding schemes when performing link prediction on the SNOMED-CT dataset for clinical terms. Huang et al. (2019) showed insignificant differences in the accuracy of QA models using TransE, TransH, and TransR embeddings, but all three outperformed the no embedding model by at least 1.8% absolute points. In developing the STransE embedding method, Nguyen et al. (2016) compared 15 different methods on 12 different WN18 and FB15k tasks and found five of them received the top score in at least one category.

3 Experiments

3.1 Task Definition

We choose the Commonsense Knowledge Aware Conversational Model (CCM) by Zhou et al. (2018) to illustrate the effect of different embeddings on social dialogue. The goal of the model is to generate a response of length m, $Y = y_1 y_2 ... y_m$ given a post of length n, $X = x_1 x_2 ... x_n$ and a knowledge graphs $G = \{G_1, G_2, ..., G_n\}$, where graph G_t is associated to post token x_t. Each knowledge graph G_t consists of a set of head, relation, and tail triples collectively denoted as $k_{t,j}$ where j denotes the index of the triple record in the graph G_t of token x_t. For tokens not explicitly associated with any of the commonsense knowledge triples, the graph is marked as Not A Fact. The model generates the response by selecting the response tokens that maximizes $P(Y|X, G) = \prod_{t=1}^{m} P(y_t|y_{t-1}, X, G)$.

3.2 Model Architecture

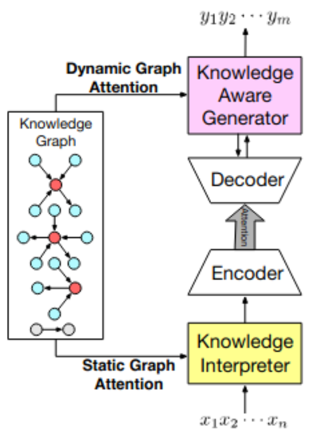

Figure 1: CCM Model Architecture

At the model's core, CCM is based on a general encoder-decoder GRU (Chung et al., 2014) text generation framework with additional attention components. As presented in Figure 1, the model consists of six key components, static graph attention, knowledge interpreter, encoder, decoder, dynamic graph attention, and knowledge aware generator Zhou et al. (2018).

First, the knowledge interpreter augments the meaning of a post token x_t with a corresponding graph vector g_t. Using the token x_t as a key, the knowledge interpreter retrieves a graph of triples G_t. Each G_t contains a set of (head, relation, tail) triples comprised of the token x_t and its immediate neighboring entities with the relations connecting them. As part of the knowledge interpreter, the static graph attention encodes the set of knowledge graph triple vectors in graph G_t for each post token into a summarized representation vector g_t. The word embedding $e(x_t)$ and g_t are concatenated as $([x_t; g_t])$ and provided as input to the GRU Encoder cell, GRU-encoder and is encoded into a state vector B_t.

The knowledge aware generator attends over all the encoder hidden states B_i, summarized graph vectors g_i and all triples in all graphs G_i to produce context vectors $C_{(t-1)(B)}$, $C_{(t-1)(g)}$ and $C_{(t-1)(G)}$ given a previous decoder state D_{t-1}. More formally, it employs three attention mechanisms as follows:

Attention over summarized graph vectors This block uses the the previous decoder state D_{t-1} and attends over all the summarized graph vectors g_1 to g_n to generate a context vector $C_{(t-1)(g)}$ using Bahdanau Attention Bahdanau et al. (2014).

Attention over encoder outputs This block uses the previous decoder state D_{t-1} and attends over all the encoder hidden states B_1 to B_n to generate a context vector $C_{(t-1)(B)}$ using Bahdanau Attention. For ease of notation, we mark the equations relative to time step t.

$$C_{(t)(g)} = \sum_{i=1}^{n} \alpha_{(t)(g)i} \mathbf{g_i}$$
$$\alpha_{(t)(g)i} = \frac{\exp(\beta_{(t)(g)i})}{\sum_{k=1}^{n} \exp \beta_{(t)(g)k}}$$
$$\beta_{(t)(g)i} = \mathbf{V_b^T} \tanh(\mathbf{W_b s_{(t)}} + \mathbf{U_b g_i})$$

Figure 2: Bahdanau Attention Zhou et al. (2018)

Graph Triple Attention This block uses the state D_{t-1} and the intermediate weights from the summarised graph vector attention and attends over every triple pair in the post to generate a context vector $C_{(t-1)(G)}$ using Luong Attention Luong et al. (2015).

$$C_{(t)(G)} = \sum_{i=1}^{n} \sum_{j=1}^{N_{g_i}} \alpha_{(t)(g)i} \alpha_{(t)(G)ij} \mathbf{k}_j$$
$$\alpha_{(t)(G)ij} = \frac{\exp(\beta_{tj}^{k})}{\sum_{k=1}^{N_{g_i}} \exp(\beta_{(t)(G)ik})}$$
$$\beta_{(t)(G)ij} = \mathbf{k_j^T W_c S_{(t)}}$$

Figure 3: Luong Attention Zhou et al. (2018)

At the next time-step t, the decoder takes in concatenation of all the previous state context vectors $([C_{(t-1)(B)}; C_{(t-1)(g)}; C_{(t-1)(G)}])$ and the previous word and knowledge triple predictions $([e(\hat{y}_{t-1}); \hat{k}_{t-1}])$ and produces the output D_t.

Finally, given the output state D_t of the decoder, and the overall context $[C_{(t)(B)}; C_{(t)(g)}; C_{(t)(G)}]$, the knowledge aware generator outputs two probabilities. One is a word token probability over the vocabulary and the other is a probability that is used to determine whether to output the predicted word token or an entity word from the knowledge graph. If the model opts to use a word entity token, the word entity token is inferred from the graph triple attention block.

3.3 Loss function

The model is directed to predict if a response token should be generated from the word vocabulary or is to be inferred from the knowledge graph triples and then predict the word or triple entity to be used. Given a post of length n, and a predicted response of length m, the first task is modeled as a binary classification problem, and is modeled using Binary-Cross Entropy loss and will be called match entity loss. For the second task, the prediction of words from the vocabulary denoted as the word prediction loss is modeled as a classification task, and is modeled using the Cross-Entropy loss. The prediction of Knowledge Graph triples is not explicitly modeled in the loss function and is inferred from the attention weights. Concretely, the word prediction loss is defined as:

$$L_{WP} = -\sum_{t=1}^{m} p_t \log(o_t)$$

where p_t is the probability of the true token from the reference distribution from the annotated response, o_t is the probability of the predicted token. The match entity loss is defined as:

$$L_{ME} = -\sum_{t=1}^{m} (q_t \log(\gamma_t) + (1 - q_t) \log(1 - \gamma_t))$$

where q_t is 1 if the ground truth response token is a knowledge graph entity and 0 otherwise, and γ_t is the predicted probability.

3.4 Dataset

The dataset consists of 10M Reddit posts, and response pairs. As not every post, response pair is connected by a commonsense knowledge graph triple, the authors filter out any post, response pair not connected by a commonsense knowledge graph triple. The original training dataset contains 3.38M training conversational pairs. Initially, we subsampled 10% (338K conversations) of the dataset due to resource constraints, however, we inadvertently trained this subsample without the Glove embeddings. Later, due to time constraints, we subsampled an even smaller proportion (1%) and retain 33.8K conversational post-response pairs. For validation and test sets, we maintain the same size as the authors, 10K for the validation set and 20K for the test set. A sample post, response pair as well as invoked commonsense knowledge triples are given in table 2.

3.5 Training Details

Due to budgetary constraints, we first validated the environment configuration and code on a Google Colab notebook. Upon validation, we trained the CCM model on single Nvidia T4 GPUs on the Google Cloud Platform and locally. The code was implemented in TensorFlow 1.3. We used the same hyperparameters as the original CCM model and a graph embedding dimension of 400. The bug-free version of CCM implementation can be found at https://github.com/ChrisBenka/ccm.

3.6 Manual Annotation of Commonsense KG Relation Classes

CCM uses 120,850 triples from the commonsense knowledge base, ConceptNet. The triples are constructed from 21,471 entities and 44 relations. Manual annotation of the 44 relations revealed that 8 of the relations are symmetric, 31 are anti-symmetric, and 44 are invertible. We classified the relations according to the following definitions Ji et al. (2021),

$$\begin{aligned}\text{Symmetric} : \forall x, y, (x, r, y) &\rightarrow (y, r, x)\\ \text{Antisymmetric} : \forall x, y, (x, r, y) &\rightarrow \neg(y, r, x)\\ \text{Inversion} : \forall (x, y), (x, r_2, y) &\rightarrow (y, r_1, x)\end{aligned}$$

Of the 120k triples, 83% contain symmetric relations, 15% contain anti-symmetric relations, and 99% contain invertible relations, according to our analysis of the common English meanings of the relations and entities involved.

While we categorized 8 of the relations as symmetric, none satisfy the formal definition of symmetry in the ConceptNet dataset. For example, the relation RelatedTo is symmetric, but of the 90k occurrences of the relation in the knowledge graph, less than 6% of the entities involved in the relation satisfy the implication above. Likewise, for the symmetric relation HasContext, for the nearly 3k occurrences, 0 of the entities involved in the relation satisfy the implication. Similarly, while we identified almost every relation as invertible, none of the respective inverse relations exist in the dataset.

To address these concerns, we augmented the dataset using the following steps: for every triple involving a symmetric relation aRb, we added bRa to the triples and for every invertible relation aRb, we added $bR^{-1}a$ as a triple. After augmentation, the dataset contains 234,858 triples. Nearly 200k involve symmetric relations, 18,749 contain anti-symmetric relations, and nearly all are invertible.

The reason for augmentation is two-fold. One, we wanted to evaluate performance of embeddings for knowledge-guided response under circumstances where knowledge graphs are guaranteed to include symmetric and inverse relations, and secondly, to see if the model improves when equipped with embeddings generated from the augmented set.

We expect transE to perform worse in the augmented set as compared to the unaugmented set as the augmented set has symmetric relations that transE cannot model effectively. For other embeddings, we expect either an improvement or no correlation depending on how well the model conditions on the embeddings.

3.7 Training of Alternative Embeddings & Generation

We used the DGL-KE library (Zheng et al., 2020) to generate knowledge graph embeddings for the model. We separately trained embeddings of dimension 400 for both the original and the augmented versions of the dataset for each of the following embeddings. Below is a brief description of the embedding schemes we plan to analyze. We will refer to head as h, tail as t, and relation as r in this section.

1. TransE Bordes et al. (2013) - Represents h, r and t all as vectors in R^d space such that $h + r = t$. Cannot model many-to-one and one-to-many relationships since unique r has unique h and t pairs.

2. TransR Lin et al. (2015) - Represents h,t as vectors in R^k space and r in R^d space with $k > d$ such that $h.M_r + r$ = t.M_r, where $M_r \in \mathbb{R}^{k \times d}$. Needs to learn numerous parameters, and can be prone to over-fitting.

3. RESCAL Nickel et al. (2011) - Uses a binary 3-dimensional tensor formed by setting valid (h, r, t) triples to 1 and rest to 0, and performs a

Prompt	I'll do it later, gonna have a cigarette first.
Relevant Knowledge	(cigarette, IsA, smoking)
Response	Smoking is bad, you know.

Table 2: Sample prompt, gold response, and knowledge triples invoked in response

Embedding	Augemented/Unaugmented	PPL	Entity Score
TransE	Unaugmented	144.67 ($\pm$11.29)	1.19 ($\pm$0.01)
	Augmented	166.40 ($\pm$ 10.47)	1.14 ($\pm$ .025)
TransR	Unaugmented	159.10 ($\pm$ 9.13)	1.20 ($\pm$0.01)
	Augmented	152.22 ($\pm$ 10.43)	1.35 ($\pm$.022)
Rescal	Unaugmented	183.33 ($\pm$ 11.48)	1.25 ($\pm$.015)
	Augmented	195.40 ($\pm$11.98)	1.32 ($\pm$ .019)
DistMult	Unaugmented	167.10 ($\pm$96.53)	1.22 ($\pm$.015)
	Augmented	163.88 ($\pm$10.43)	1.20 ($\pm$ .011)

Table 3: Averaged perplexity and entity match scores with respective standard deviation on testset. Model trained on 1% of training data, or 33.8k posts.

Prompt	They're as **fake** and arbitrary as religion
Relevant Knowledge (Augmented)	(fake, relatedTo, real)
TransR (Augmented) Response	I don't think it's a **real** thing.
Relevant Knowledge (Unaugmented)	-
TransR (Unaugmented) Response	I don't know if it's a religion or not. I'm sure it's a cult of religion.

Table 4: Responses on a prompt for TransR (trained on the augmented knowledge graph) and TransR (trained on the original commonsense knowledge graph)

Prompt	It's still less **money** than just buying a ticket to get into wrigley.
Relevant Knowledge (Augmented)	(money, RelatedTo, worth), (money, RelatedTo, spend) (money, RelatedTo, price)
TransE (Augmented) Response	I'm not sure I wouldn't want to **spend** a deal with it. I'm just saying that the **price** of the game isn't **worth** it.
Relevant Knowledge (Unaugmented)	(money, RelatedTo, spend)
TransE (Unaugmented) Response	I don't think they're going to **pay** for it.

Table 5: Responses on a prompt for TransE (trained on the augmented knowledge graph) and TransE (trained on the original commonsense knowledge graph)

dyadic decomposition on them. Each slice X_k on this tensor is approximated to AR_kA^T where A is shared between all relational slices (and contains latent information about the relations in A), and R_k is a slice of the tensor M_k that the algorithm tries to find. It is computationally intensive and can have overfitting issues.

4. DistMult Yang et al. (2014) - Similar to RESCAL, but uses a computationally simpler model by restricting $\mathbf{M}_r$ to be a diagonal matrix and hence has less tendency to overfit as compared to RESCAL. Works well only for symmetric relationships since symmetric relationships can be diagonalized, and has a high error rate in other cases.

Due to differences in data formats generated by the DGL-KE library and CCM library, we designed a script to facilitate the data conversion with the proper entity and relation id mappings. The Github implementation is available here.

4 Case Studies

The following case studies analyze the effects of the modifications to the CCM model made in this paper:

4.1 Augmenting the KG with Relations

As described in section 3.6, we performed an augmentation of the commonsense knowledge graph. The following subsections analyze the effects of such annotations on our results.

4.1.1 Makes reasoning better

Table 4 illustrates the differences between the responses generated by the CCM model trained with TransR embeddings with the augmented knowledge graph and with the original knowledge graph.

The prompt has three main entities from the Commonsense Knowledge Graph - "fake", "arbitrary", and "religion". The unaugmented knowledge graph specifies the "RelatedTo" relation in a unidirectional manner. Therefore, the relation (real, RelatedTo, fake) occurs only as a unidirectional edge.

By contrast, it occurs as a bidirectional edge in our augmented version of the commonsense knowledge graph, as our graph considers all the symmetric relations to be bidirectional. More concretely, the relation (fake, RelatedTo, real) occurs explicitly in our augmented version of the commonsense knowledge graph.

The prompt uses a simile to say something is fake. The unaugmented version of our TransR model reasons something about the entity "religion" - which looks irrelevant based on the context. However, the augmented counterpart of this model not only reasons correctly about the prompt but also includes the word "real" based on our augmented edge.

4.1.2 Model includes more relevant entities in its results

It is clearly evident from the results that the CCM models trained with the augmented knowledge base include more entities than their unaugmented counterparts on average. It can also be seen that the perplexity scores are higher for these models. This could be due to the fact that including more entities in the graph makes it tougher for the decoder to generate sentences with those entities.

Table 5 illustrates another cherrypicked example from the CCM evaluation dataset, where as a result of augmenting the knowledge graph with relevant backward edges, the model is able to discover more relevant entities. The prompt mentions getting a ticket for "Wrigley" is cheaper.

Similar to the previous example, augmenting the Knowledge Graph with backward relations introduces more relevant entities in the result. In the original version of the KG, the entities (worth, RelatedTo, money) and (price, RelatedTo, money) were present as unidirectional edges. From our augmentation process, two more (money, RelatedTo, worth) and (money, RelatedTo, price) were introduced to the (head, relation, tail) relation triples space. We reason that the addition of these relations induces the CCM model to include more entities and leverage more connections in the network.

5 Additional Efforts

We additionally came across another embedding RotatE Sun et al. (2019) which represents (h, r, t) in $\mathbb{C}^d$, with r being element-wise complex rotations (element r_j of the form $e^{i\theta}$) on h represented by $h \circ r$ and mapping to t. This embedding is novel in that it can learn composition of relations effectively, owing to it's complex space representation, and we think that this embedding can help the model learn reasoning better.

However, casting the complex values to real values, either by appending real and imaginary parts, or neglecting the imaginary parts is likely to void the representation capability of the embeddings. Hence we wanted to evaluate it on a modification to CCM to allow complex valued neural networks. Since Tensorflow 1.3 had limited support for complex valued operations and neural networks, we tried to port the model to TensorFlow 2.0. After significant time and effort, we realized that the model implementation used many features deprecated from TensorFlow 2.0 and redirected our efforts to re-implement the whole model and preprocessing pipeline from scratch.

We re-built the model using PyTorch as it had better support for complex-valued neural networks and the pre-processing pipeline in Julia for faster execution and training. This involved significant effort, and we have the full model training. Unfortunately, due to complexities surrounding complex-valued neural networks, the loss is not converging beyond a point. At present, the model predicts the most frequently occurring tokens, i.e. 'go', '.', 'the', and 'eos'. We are exploring optimization techniques to improve convergence.

The Model and Pre-processor, along with the training data used are available at https://github.com/mj2984/CCM_Complex

6 Results

We find that among unaugmented models, Rescal achieved the greatest entity score while TransE achieved the lowest perplexity. Among augmented models, TransR achieved both the greatest entity score and the lowest perplexity. Overall we observe a tradeoff between perplexity and entity score across models (Figure 6)

Augmentation appears to have a beneficial effect on entity score for TransR and Rescal, but not for TransE and DistMult. We attribute the increase in entity score to the models' ability to extract inverse relations from the knowledge graph that are likely to be used in responses. For example, given the existence of a triple (cigarette, IsA, smoking) but not the inverse relation, the graph encoder will be unable to introduce "cigarette" from a prompt including "smoking". On the other hand, augmentation likely had a deleterious effect on TransE due to TransE's degenerate behavior under the augmentation scheme. It is also possible that tuples added during augmentation increase the number of entities involved in reasoning, decreasing the signal-to-noise ratio and thereby raising perplexity.

These results, however, display relatively high perplexity, likely due overfitting on the relatively small training set. However, we observe similar trends in the appendix data which included 10x training data but did not use Glove embeddings of text, suggesting our findings would hold with better trained models.

We also find that in the model formulation for the loss function, there is a significant imbalance in the weights for predicting match entity rate and for predicting the vocabulary. Since there are only about 1.3 graph matches per sentence on average while having 26 tokens per sentence on average, we see that the loss weights are heavily skewed on the language prediction. We did not notice any regularization applied on the original CCM implementation, nor do we see any mention of it in the paper. We suspect this to be influencing a part of the trade-off we notice between match-entity rate and perplexity, and it would be interesting to see how the model performs with regularization applied.

7 Limitations

Due to time and budgetary constraints we had to limit the scope of our experimentation compared to what was done in the original CCM model. We reduced the training dataset for KG embeddings to 10% of its original size, so it may not cover all entities present in the validation set. Additionally, due to limited GPU resources available to us, we did not perform a hyperparameter search for the training of KG embeddings and the CCM models. Instead, we replicated the hyperparameters in the original paper. It is possible that an extensive search can find a better hyperparameter combination that would produce superior results.

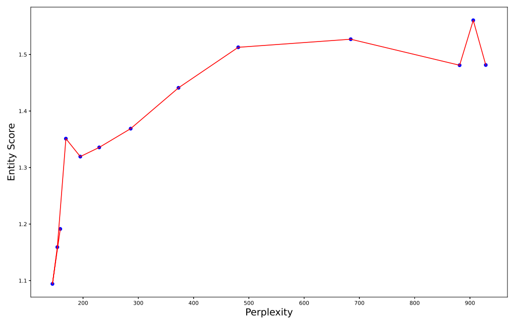

Figure 4: Relationship between Perplexity and Entity Score for CCM using TransE (unaugmented)

8 Future Work

CCM only uses single-hop relations to propose candidate topics. However, in reality, some coherent conversational topics may be two or more hops away. Future work could include searching larger subgraphs of the knowledge graph for coherent topical changes. This task would likely better highlight the expressiveness of more advanced KG embeddings and those that are able to model compositionality. CCM's GRU architecture and hierarchical, top-down dynamic graph attention may also be replaced by more recent transformer-based architecture and more powerful graph encoding techniques.

First, we will continue to explore the KG embeddings based on complex numbers (ComplEx, RotatE) with our new PyTorch implementation of CCM. These embeddings capture both symmetric and antisymmetric relations while maintaining a computational complexity similar to the basic TransE approach, at $O(d)$. We have implemented the model, in a performant way. We would focus on optimizing the loss function to generalize better, and train on the full dataset.

9 Conclusion

We present a study of common knowledge graph embedding techniques and their ability to enhance conversational models' ability to smoothly transition between related topics and introduce new subject matter. Our results show a trade-off between perplexity and entity score as well as an improvement in both metrics when augmenting the knowledge graph, except in the case of TransE, which is consistent with our theoretical predictions. We also propose a method for augmenting knowledge graphs to improve their alignment with specific embedding schemes.

10 Division of Labor

- Initial Graph Embedding Generation Exploration - Matt, Manuel, Bobby
- Generate knowledge graph embeddings and model files - Chris
- Conversion of CCM graph into DGLKE format for training scripts - Nikhil
- Resolving issues with CUDA and TF 1.3.0 and the first test version of the model - Nikhil
- Establish the training pipeline in TensorFlow - Nikhil, Matt, Bobby
- Set up an efficient and workable training infrastructure eventually used in all members' gcp instances, resolving CUDA issues and TensorFlow bugs - Bobby
- Dataset Analysis - Chris, Matt, Bobby
- Create the augmented dataset for knowledge graph embeddings - Manuel, Matt
- Tensorflow migration - Chris, Nikhil, Manuel
- Preprocessor Re-implementation - Manuel
- PyTorch Model Re-implementation - Manuel, Bobby
- PyTorch Performance Optimization - Manuel, Chris
- Finding Bugs in Implementation and Use - Matt, Manuel, Nikhil, Bobby, Chris.
- Model and architecture Analysis - Manuel, Chris
- Case Studies - Nikhil
- Model Training - Matt, Manuel, Nikhil, Bobby, Chris. (shared work-load)
- Floating - Matt, Bobby

11 Individual Contributions

My first contribution focused on training all embedding schemes for the model on the augmented and augmented datasets. After training the embeddings, I converted all the embeddings files into the required input formats for the model. I manually classified the relations in the dataset and computed the relevant statistics through python scripts. For the development of the model, my initial focus was on porting the code from TensorFlow 1 to TensorFlow 2 to allow for training on modern CUDA architectures and complex-value customizations. This was a rather arduous task, as many critical APIs used by the TensorFlow 1 version are not supported in TensorFlow 2. I spent significant time revising the model to use these new APIs. A decision was made to abandon the TensorFlow 2 migration and run the existing code on an older CUDA version on GCP. I trained the TransR augmented and unaugmented models and the Rescal unaugmented model. As Manuel was having more success in the PyTorch implementation, I spent the remainder of my time assisting him in the rewrite and optimizing the code to run the model on my 2080-TI so that we could make the complex-valued extensions to the model.

As for the report, I wrote the abstract, task definition, dataset, and manual annotation of Commonsense KG relation classes sections. Manuel and I jointly analyzed and wrote the model and loss functions sections. I also wrote the training details section with Nikhil.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.

David Chang, Ivana Balažević, Carl Allen, Daniel Chawla, Cynthia Brandt, and Richard Andrew Taylor. 2020. Benchmark and best practices for biomedical knowledge graph embeddings. In *Proceedings of the conference. Association for Computational Linguistics. Meeting*, volume 2020, page 167. NIH Public Access.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2018. Wizard of wikipedia: Knowledge-powered conversational agents. *arXiv preprint arXiv:1811.01241*.

Karthik Gopalakrishnan, Behnam Hedayatnia, Qinglang Chen, Anna Gottardi, Sanjeev Kwatra, Anu Venkatesh, Raefer Gabriel, Dilek Hakkani-Tür, and Amazon Alexa AI. 2019. Topical-chat: Towards knowledge-grounded open-domain conversations. In *INTERSPEECH*, pages 1891–1895.

Xiao Huang, Jingyuan Zhang, Dingcheng Li, and Ping Li. 2019. Knowledge graph embedding based question answering. In *Proceedings of the twelfth ACM international conference on web search and data mining*, pages 105–113.

Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and S Yu Philip. 2021. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 33(2):494–514.

Ramakrishna Kumar and Maha Mahmoud Ali. 2020. A review on chatbot design and implementation techniques. *Int. J. Eng. Technol*, 7(11).

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2015. A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055*.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Twenty-ninth AAAI conference on artificial intelligence*.

Yinong Long, Jianan Wang, Zhen Xu, Zongsheng Wang, Baoxun Wang, and Zhuoran Wang. 2017. A knowledge enhanced generative conversational service agent. In *Proceedings of the 6th Dialog System Technology Challenges (DSTC6) Workshop*.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation.

Dat Quoc Nguyen, Kairit Sirts, Lizhen Qu, and Mark Johnson. 2016. Stranse: a novel embedding model of entities and relationships in knowledge bases. *arXiv preprint arXiv:1606.08140*.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Icml*.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*.

Da Zheng, Xiang Song, Chao Ma, Zeyuan Tan, Zihao Ye, Jin Dong, Hao Xiong, Zheng Zhang, and George Karypis. 2020. Dgl-ke: Training knowledge graph embeddings at scale. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 739–748.

Hao Zhou, Tom Young, Minlie Huang, Haizhou Zhao, Jingfang Xu, and Xiaoyan Zhu. 2018. Commonsense knowledge aware conversation generation with graph attention. In *IJCAI*, pages 4623–4629.

A Graph Embeddings Only

Embedding	Augemented/Unaugmented	PPL	Entity Score
TransE	Unaugmented	73.03 ($\pm$4.50)	1.23 ($\pm$.021)
	Augmented	79.27 ($\pm$ 6.27)	1.24 ($\pm$ .017)
TransR	Unaugmented	77.00 ($\pm$ 5.21)	1.31 ($\pm$.027)
	Augmented	75.81 ($\pm$ 4.65)	1.30 ($\pm$.025)
Rescal	Unaugmented	75.14 ($\pm$ 4.61)	1.33 ($\pm$.031)
	Augmented	81.07 ($\pm$5.14)	1.38 ($\pm$ 0.035)
DistMult	Unaugmented	80.95 ($\pm$1.59)	1.25 ($\pm$0.014)
	Augmented	82.97 ($\pm$5.09)	1.37 ($\pm$0.021)

Table 6: Averaged perplexity and entity match scores with respective standard deviation on testset. Model trained on 10% of training data, or 333.8k posts. Upon close inspection of the results, we realized the pretrained Glove word embeddings were not provided as inputs to the model. These results reflect only having the trained graph embeddings as input to the model.