Online Learning of Temporal Knowledge Graphs

Paniz Bertsch
McGill University
Montreal, QC H3A 0G4
paniz.bertsch@mail.mcgill.ca

Nishant Mishra
McGill University
Montreal, QC H3A 0G4
nishant.mishra@mail.mcgill.ca

Albert M Orozco Camacho
Mila - Québec AI Institute
Montréal, QC, Canada H2S 3H1
alorozco53@mila.quebec

Abstract

For many computer science sub-fields, knowledge graphs (KG) remain a constant abstraction whose usefulness relies in their representation power. However, dynamic environments, such as the temporal streams of social media information, brings a greater necessity of incorporating additional structures to KG's. In this project, we intend to apply currently available solutions to address incremental knowledge graph embedding to several applications to test their efficiency. We also propose to build an embedding model agnostic framework to make these models incremental. Firstly, we propose a window-based incremental learning approach that discards least happening facts and performs link prediction on updated triples. Next, we present experiments on a GCN model-agnostic meta-learning based approach. Our best model is the Window-based KG Incremental Learning, where edge representations, are calculated from subtraction of embedding vectors of head and tail nodes.

1 Introduction

Knowledge bases store millions of facts about people, places, and events[1]. Many of them are also generated and deployed online, in order to be used for question answering and information retrieval by end users or third party companies, to provide service to their customers. With the exponential increase of new information, and dynamic nature of many knowledge bases, maintenance and accurate fact extraction is becoming a challenging task. Current knowledge bases are missing many links between existing facts; thus, there is a constant need of adding new facts and updating old relations [10].

Graphs are the main abstraction to represent facts in a well-formed sentence that involves a *subject*, a *predicate or relation*, and an *object* in knowledge bases. This fact allows them to benefit from many efficient graph algorithms to search, delete, and update facts. Recent efforts in applying machine learning methods to graphs, can help mitigate current challenges of **knowledge graphs** (KG) with dynamic graph structure, and improve application performance in an online environment. While there exist numerous models to learn KG embeddings [8], they are mostly limited to static KGs and cannot model an evolving KG, where new facts about the world are constantly added or updated. New facts provide additional information about existing entities (new edges) in the KG and also may provide added information about new entities (new nodes).

[1] Wikipedia is one of the main sources from which one can construct graph-like representations that encode facts about the world.

Preprint. Under review.

In this project, we apply different methods to a knowledge graph to learn accurate knowledge representations as an output of a set machine learning methods. Our main constraint deals with enforcing the proposed techniques to operate on dynamic graphs, that is, graphs whose set of vertices and/or edges change over a (*discrete*) time interval.We focus on providing methodologies to obtain representations (embeddings) for these new entities and also update the representations of existing (old) entities that have received new facts.

1.1 Task Definition

Though obtaining new entity representations is the focus of the work, the primary task is still to perform link prediction for KG completion. However, note that it is different from the conventional KG completion task where the graph is static, and new facts/links are predicted only for a fixed set of existing entities. Existing models cannot be used to obtain facts for entities that are not present in the graph. The typical naive solution is to train the KG embedding model from scratch every time a new set of facts arrive, which is computationally expensive with large graphs.

Hence, it is ideal to look forward to solutions where the old model can quickly adapt to new training data without forgetting the information learned in the past. This is the classical problem of incremental learning, which is also commonly referred to as continual learning and online learning [4]. We pose the problem of continually incorporating facts about new entities in the KG model as an incremental learning problem. Note that the parameters of the model are the embedding of entities themselves and their relations.

Consider a **directed graph** $G = (V, E)$ together with a set of labels $\mathcal{L}$ that serve to classify each directed edge $(s, o) \in E$. In a general sense, there is a relation $R \subseteq E \times \mathcal{L}$ that does such labeling. In the *knowledge graph* (KG) jargon, and for simplicity in notation, we prefer to observe these data as a set of related triplets `(subject, relation, object)` (or (s, r, o)) where s and o are graph vertices called *entities*, while r is one of the corresponding labels for edge (s, o). Furthermore, the set of triplets that characterize a KG is often called a *facts set* during application settings.

We formulate the problem of *online learning* on KGs as follows. At time-step, $t + 1$, if $\mathcal{T}_t$ denotes the facts set at t and $\mathcal{Z}_t \in \mathbb{R}^{|\mathcal{E}_t|*d}$ denote the embeddings for the entity set $\mathcal{E}_t$ at t, then, the task is to obtain embeddings, $\mathcal{Z}_{t+1} \in \mathbb{R}^{|\mathcal{E}_{t+1}|*d}$ based on the new (updated) facts set, $\mathcal{T}_{t+1}$. Note that, $\mathcal{T}_{t+1} - \mathcal{T}_t$ corresponds to the new facts added and $\mathcal{T}_t - \mathcal{T}_{t+1}$ corresponds to the deleted facts. We restrict the scope of project to cases where $|\mathcal{E}_{t+1}| > |\mathcal{E}_t|$ and to only one future prediction, i.e $t \in \{0, 1\}$.

We organize our report as follows. First, in Section 2, we provide the background on TransE, a popular KG embedding model, and discuss a few other related works. Then, we formally pose the Incremental KG embedding problem and discuss the proposed methodologies to solve the problem. Then we provide the dataset and experiment details in Section 3 and 4 respectively and discuss the observed results in Section 5. Finally, we conclude our report discussing our findings in detail in Section 6, and note areas for future work.

2 Related Work

In this section we elaborate on some of the influential approaches taken recently to tackle the task defined on Section 1.1. All of them have significantly gotten benefit from deep neural networks. On the other hand, as the problem of online learning can be well formulated on alternative fields, such as reinforcement learning [1], automated reasoning [5], and recommendation systems [12].

2.1 TransE

Several models and paradigms have been proposed in the literature to learn embeddings for KGs [13] TransE [2] is used to learn embeddings for entities and relations of multi-relational KG data in low-dimensional vector spaces. TransE models the embeddings by regarding a relation as translation from head entity to tail entity - if (h, r, t) holds, then the embedding of the tail entity t should be close to the embedding of the head entity h plus some vector that depends on the relationship r. The scoring (energy) function for TransE is given by,

$$E(h, r, t) = ||h + r - t||$$

Given a training set S of triples (h, r, t), where $h, t \in E$ (E is the set of entities) and $r \in R$ (R is the set of relations), the embeddings are learnt by minimizing a margin-based ranking criterion over the training set.

$$L = \sum_{(h,r,t)\in S} \sum_{(h',r,t')\in S'_{(h',r,t')}} [\gamma + d(h + r, t) - d(h' + r, t')]_+$$

where $[x]_+$ denotes the positive part of x, $\gamma > 0$ is a margin hyper-parameter and $S'_{(h',r,t')}$ is the set of corrupted triplets.

$$S'_{(h',r,t')} = \{(h', r, t)|h' \in E\} \cup \{(h, r, t')|t' \in E\}$$

The set of corrupted triplets is composed by replacing either the head or tail entity with a random entity. The loss function favors lower energy values for training triplets than for the corrupted ones. This model requires reduced set of parameters as it learns only one low-dimensional vector for each entity and each relationship. We used OpenKE's [2] implementation for setting our model. For our task, we made changes to the TransE model, so that it can learn the representations of the new entities.

2.2 Dynamic Knowledge Graph Embeddings

In [14] a context-aware *Dynamic Knowledge Graph Embedding* (DKGE) method is proposed. Their contributions include two utilizing different embedding representations for entities and relations, separately. The proposed architecture will aim to learn *contextual subgraph embeddings* that will characterize a small neighborhood from a KG, via a deep *GCN*[3] together with an attention mechanism on vertex features. The appearance of such contextual embeddings is what effectively distinguishes this approach with others, such as puTransE [11].

During the process of updating a knowledge graph. One of the major contributions of [14] was to propose an incremental algorithm to learn better representations by only updating entities and relations that were involved in adding/deleting operations, as well as, emerging entities. Such updates are performed in proportion to the gradient of a *margin-based loss function*, defined as

$$\mathcal{L} \equiv \sum_{(h,r,t)\in S} \sum_{(h',r,t')\in S'} \max(0, f(h, r, t) + \gamma - f(h', r, t')) \tag{1}$$

where S and S' are the set of **correct** and **incorrect** triplets for an arbitrary KG, respectively; γ is the margin; and f is a scoring function[4].

2.3 Other Related works

In another earlier experiment we used transfer learning for the task of incremental learning of Knowledge Graph embedding. It was done by finetuning the pre-trained embedding using new data(triplets) involving the new nodes by freezing of relation embedding and embedding of old nodes that are not the neighbour nodes of the new nodes instead of retraining from scratch. This training was done using OpenKE for training TransE embeddings. This selective freezing of weight vectors was done by changing the trainer model in OpenKE, setting a gradient mask that forces the gradients of weights associated with these entities to be zero at each epoch. The results obtained are tabulated in Table 1 along with results where no freezing of embedding was done. As can be observed from the results transfer learning results in much faster training with very small loss in performance when compared with the skyline performance i.e when the new knowledge graph is trained from scratch.

Another interesting approach to the presented task comes from the temporal setting, in which the dynamics of a KG can be seen as an *evolving* phenomenon. In [6], the authors model the temporally conditioned joint probability distribution for *event sequences* (KG updates) by presenting the novel Recurrent Event Network (RE-Net). This is a discriminative model that uses a recurrent neural network to approximate the probability $\mathbb{P}\{s_{t+1}, r_{t+1}, o_{t+1} \mid G_{:t}\}$ at current time step conditioned on the preceding events. Such approach employs a probabilistic framework to predict the forthcoming events based on previous observations.

[2] https://github.com/thunlp/OpenKE

[3] graph convolutional network

[4] An l_1-norm is often used in this setup.

Model	MRR	MR	Hits@10	Hits@3	Hits@1
Fine-tuning TransE	0.2993	1872	0.5449	**0.4801**	0.1026
Fine-tuning TransE - static relation embeddings	**0.3727**	1387	**0.5676**	**0.5148**	0.2152
Skyline model (Training from scratch)	0.3959	1244	0.5678	0.4783	0.2815

Table 1: Evaluation results on TransE with different settings.

[8] provides a comprehensive review of the advances in representation learning for dynamic graphs. The work points out that the literature on dynamic graphs is limited to only a few and that too only for temporal graphs where new facts or added or old facts dropped in a KG with static number of entities. The review also acknowledges the need for models that can learn representations for new entities.

The scarcity of models for incremental learning in KGs can be attributed to the complexity of multi-relational graphs. Only in the recent past, embedding models for simple dynamic graphs have surfaced. In the NLP side of research, [7] explored an incremental training strategy for training word embeddings using skip-gram models. We adopt ideas from this paper for setting up our incremental approach to dynamically upgrade the embedding dictionary and update the negative sampling probability.

3 Dataset

Triples in training	Triples in validation	Triples in testing	Triples in test_zeroshot	No. of Entities	No. of Relations
4,83,142	50,000	59,071	31,078	19,970	1,345

Table 2: Statistics of FB20K dataset.

For this experiment we use multi-relational FB20K dataset and only consider relation for each triplet (head, relation, tail) rather than node or edge attributes. Figure 1 shows a graph of 400K edges in FB20K dataset. Freebase provides general facts of the world, and is an extensively used dataset for knowledge completion tasks. We employed the FB20K dataset [15] for our task. In addition to containing all the entities and relations from the FB15K dataset, this dataset also contains new entities which was required for our setup. As a first step, we analyzed and visualized this dataset to gain insights. The FB20K dataset contains 1,345 relations and 19,970 entities in total, out of which 14,951 are present in the training set. The test_zeroshot file contains all the new entities that are not seen in the training data, while the test file contains the triples between the existing entities (e-e). The zeroshot setting contains a total of 9,217 entities, from which 5,019 are the new ones. The triplets in test_zeroshot can be split in three categories: (e-d), (d-e), and (d-d); where e stands for existing and d stands for the new entities. (e-d) means a triple with an existing entity as its head and a new entity as its tail, (e-d) implies that the tail is a new entity but the head is not, and (d-d) implies that both the head and the tail are new entities. The statistics on the zero-shot setting are given in Table 2.

4 Proposed Approach

4.1 Window-based Knowledge Graph Incremental Learning

In this method, an initial set of directed edges and nodes, are chosen as a base for training a model to capture structural information of a knowledge graph. Node and/or edge features created based on the underlying graph structure. Frequently, nodes and edges represented as vectors in a lower dimensional space using available algorithms or learned through a deep learning model [3]. These

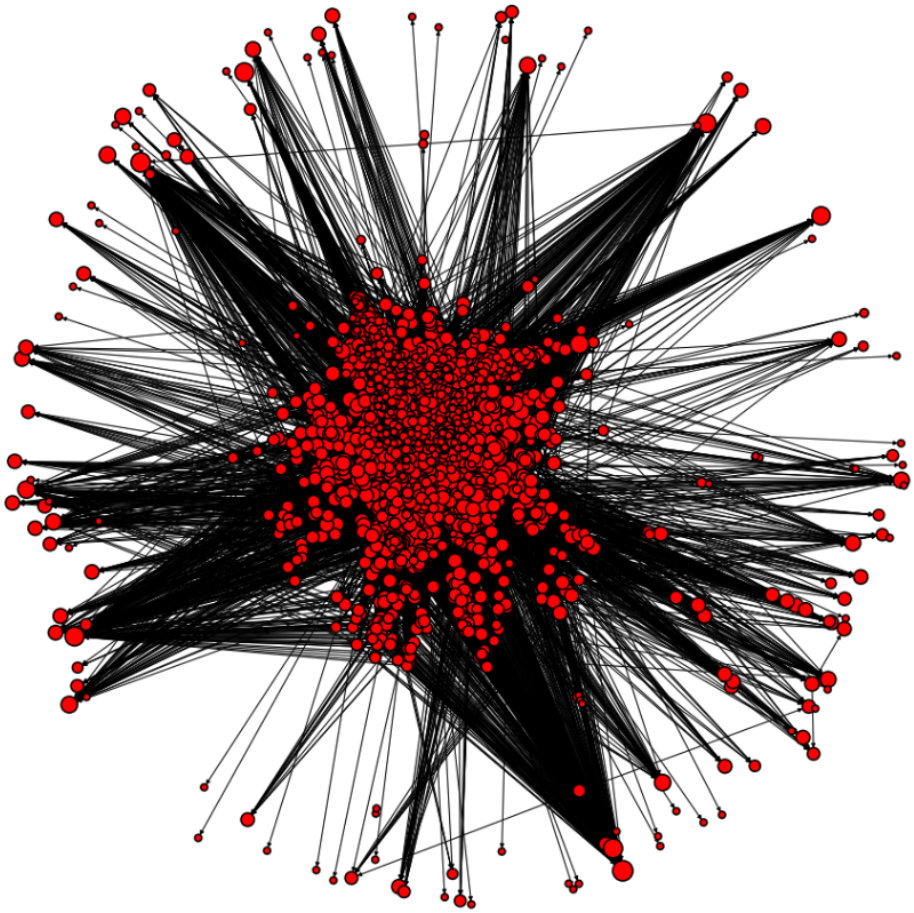

Figure 1: Visualization of the Freebase Graph

embeddings are used for training the base model to be used for prediction of new or missing facts. In this method, Node2Vec based on Word2Vec is used to generate node embeddings.

Next, current snapshot of the graph is updated by replacing least occurring facts (less central nodes & unconnected nodes). Embedding of new nodes are generated from this snapshot of the graph and passed to our base model, to perform link prediction for these new facts. Good performance is obtained when ratio of new to existing nodes is not greater than $\%15$ of previous graph snapshot. This is beneficial for online methods as there is no need to train a model from scratch with new data arrival.

To create edge embedding vectors, we experiment with two methods:

1. Concatenating head and tail's 128-dimensional Node2Vec embedding vectors to create 256-dimensional edge embedding
2. Subtracting head embedding from tail embedding vector to create 128-dimensional edge embedding vector

For this experiment, link prediction is converted to a binary classification with 0 and 1 representing link is present or absent respectively, with Random-Forest model for training and prediction. Also, dataset is divided to training set and nine test sets as incremental updates, to generate 9 snapshots of graph with each snapshot, adding new nodes and updating edges compare to previous graph snapshot. Negative samples generated for each set equal to the number of existing edges in each set, where shortest distance between nodes of each negative sample is greater than 2.

4.2 GCN based

The second method we experimented with followed a model-agnostic meta-learning based approach with Graph Convolutional Networks(GCN). The idea here is to learn a GCN to predict the embeddings of new nodes given the old embeddings of its neighboring entities in the old graph and similarly obtain an updated representation of old entities based on the recently learned embedding of new entities. These two predictions are jointly iterated. This can be viewed as learning to learn problem (meta-learning).

To train this model, we prepared multiple sets of 2 time-stamped graphs and learned a GCN to learn embeddings for new entities when a new time-stamped graph set is provided. For data creation to feed into our GCN model, we took the training data and removed entities from them in an incremental way such that the entities removed do not affect the connectivity of the graph. We also ensured the number of relations remains the same and that we don't remove any entities which might be associated with any relation more than 50% of the total triplets involving that relation.

Now the graph with entities removed was considered our old graph and embeddings were learnt for this graph using triplets, and the graph with those entities as part of the data was considered as new graph and trained its embeddings by keeping embeddings of all entities from the old graph that are not in any way related to any of the new entities added in the new graph constant and the rest(neighbours of new entities) along with the new entities trainable.

We froze them by making their gradients zero at every epoch during training. We also froze the relationships as they remain the same. In this way we created 5 sets of data for training by removing random sets of 1000 entities from the original training data in FB20k. So 4 out of the 5 such pairs of old and new knowledge graphs without and with additional 1000 entities were to be used as training data for our GCN, the other set being the validation set while the training data and the training data appended with new entities in the test_zeroshot data in FB20k were to be used as test data.

The datasets prepared were in the form of an object file consisting of, the adjacency matrices of old and new knowledge graphs, the relation-entity sparse adjacency matrix,the indices of new entities, and the indices of neighbouring nodes of the new entity for each pair. This data was parsed and sent to the GCN model for training. Although it is meant ot be model agnostic, we used the TransE model here for our experiments.

5 Results

5.1 Window-based Knowledge Graph Incremental Learning

Precision, recall and F1 score are the commonly used metrics for binary classification. F1 score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0 [9]. For this method, area under ROC curve and F1-score used for selecting model parameters and evaluating of our model. Table 5.1 summarizes results of this experiment. We observe that, base model which is trained using initial set only, predicts links with high confidence for new sets (82% on average), however accuracy drops for snapshots with larger new nodes. Also, as expected, edge features generated from subtracting head embedding from tail embedding vector, performs better than concatenating head and tail embeddings. This could be explained by the way nodes are mapped to low dimensional space. Node2Vec uses Word2Vec to generate low-dimensional embedding vectors. When Word2Vec used for word embeddings, interesting results observed from arithmetic operation on embedding vectors such as King + Woman - Man = Queen. Because our classification problem attempts to predict existence of an edge from an embedding vector, head + edge = tail or edge = tail - head, can better preserve direction of edges and differentiate between node or edge entities. Figure 2 and 3 show Precision, Recall and ROC curve for one snapshot.

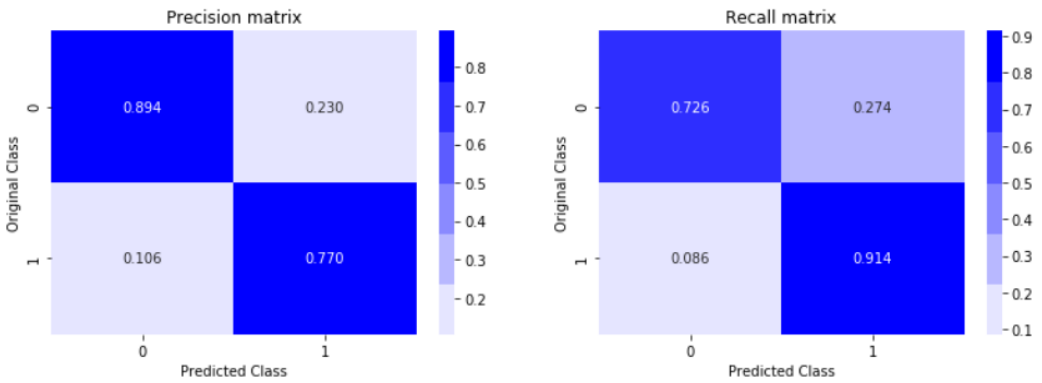

Figure 2: Precision & Recall Matrix for Snapshot2

	Edge size	# New nodes	Subtract-F1 Score	Concat-F1 Score
Train set	$290K$	$15K$	87 %	87 %
Snapshot 0	$50K$	1497	78 %	69 %
Snapshot 1	$10K$	108	84 %	69 %
Snapshot 2	$10K$	32	84 %	67 %
Snapshot 3	$10K$	118	85 %	68 %
Snapshot 4	$10K$	18	83 %	69 %
Snapshot 5	$10K$	960	72 %	69 %
Snapshot 6	11583	40	90 %	75 %
Snapshot 7	18750	54	73 %	69 %
Snapshot 8	151	25	87 %	73 %

Table 3: Window-based KG Incremental Learning

5.2 GCN based

We also trained a GCN based model to learn an embedding-model agnostic approach for incremental learning. The idea here is to learn a GCN to predict the embeddings of new nodes given the old embeddings of it is neighboring entities in the old graph and similarly obtain an updated representation of old entities based on the recently learned embeddings of new entities. These two predictions are jointly iterated. After training the two layer GCN model on the custom training dataset using Mean squared Error as the loss function, the model was evaluated using the output embeddings used in the downstream task of link prediction to calculate hits@10. The hits@10 for validation set was a decent 0.27,but it failed to generalise to the test set and gave unsatisfactory results. More precisely it failed at learning representations for new entities.

6 Discussion and Conclusions

Average performance of best model for Window-based KG Incremental Learning is 82% where edge representation, is calculated from subtraction of embedding vectors of head and tail nodes. For future experiment, we would like to extend this experiment by considering node/edge attributes and edge weights which could improve accuracy and extend application to other KG tasks. Also, using a much smaller training/update sets will be useful for a real-time online platform for link prediction and product suggestion.

The model agnostic GCN based method failed to generalise to the new entities introduced in the test set. The performance with validation set might be explained by the fact the even with random sampling of 1000 entities,the model during training had seen a large percentage of the new entities of the validation knowledge graph among the training data. Although we could not make the GCN approach work, it remains a potentially and conceptually robust approach that needs further experimentation, since it endeavors to be a model agnostic direct mapping based approach,which will

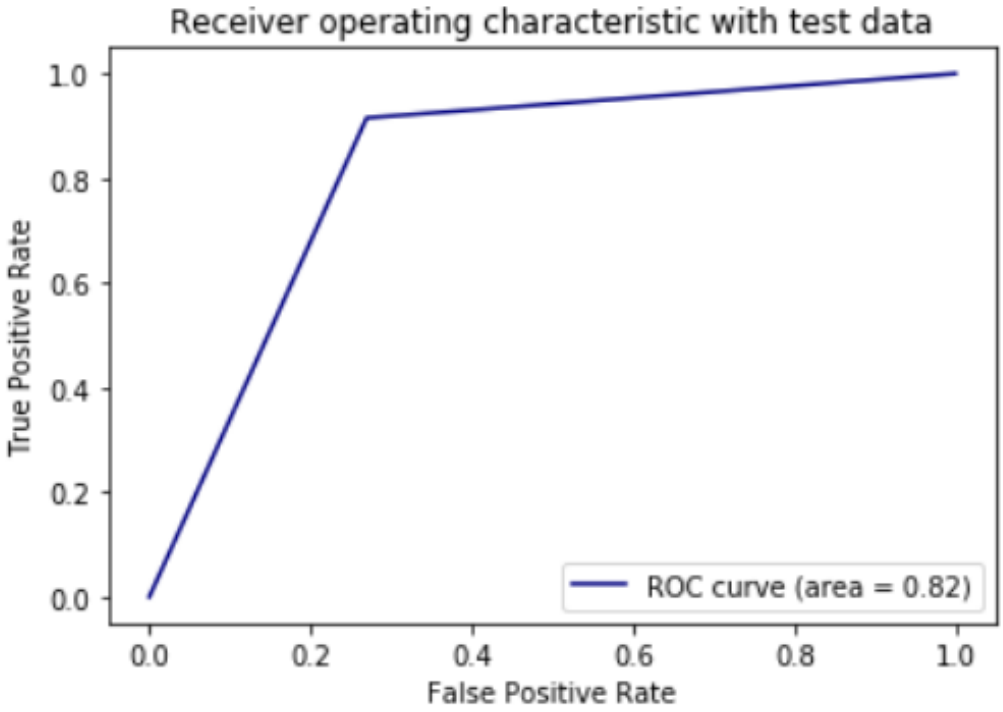

Figure 3: ROC Curve for Snapshot2

considerably reduce the time required to derive embeddings for dynamically evolving graphs.Since it is model agnostic,it will cater to the various different kinds of embedding.

All in all, the incremental learning of dynamic knowledge graphs remains an active field of interest with a number of recent research being directed to address it. While it is imperative to work on modeling this dynamic nature of knowledge graph to ensure fast and accurate embeddings are generated, stress should also be given to find solutions that not only model new entities,but also relations. More and more avenues need to be explored to ensure better models,like say, the need to calculate embeddings of new nodes,but also update the old embeddings at higher level of hops from the new nodes,not just immediate neighbours.

It is also a crucial problem to model the temporal nature of the evolution of facts over multiple time steps. In our experiments we only dealt with entities and their relations, better performance can be achieved by using adjunct information such as attributes of entities,facts etc for training.

References

[1] Ashutosh Adhikari, Xingdi Yuan, Marc-Alexandre Côté, Mikuláš Zelinka, Marc-Antoine Rondeau, Romain Laroche, Pascal Poupart, Jian Tang, Adam Trischler, and William L. Hamilton. 2020. Learning Dynamic Knowledge Graphs to Generalize on Text-Based Games. arXiv:cs.CL/2002.09127

[2] 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*. 2787–2795.

[3] H. Cai, V. W. Zheng, and K. Chang. 2018. A Comprehensive Survey of Graph Embedding: Problems, Techniques, and Applications. *IEEE Transactions on Knowledge Data Engineering* 30, 09 (sep 2018), 1616–1637. `https://doi.org/10.1109/TKDE.2018.2807452`

[4] Zhiyuan Chen and Bing Liu. 2016. Lifelong machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 10, 3 (2016), 1–145.

[5] Marcel Hildebrandt, Jorge Andres Quintero Serna, Yunpu Ma, Martin Ringsquandl, Mitchell Joblin, and Volker Tresp. 2020. Reasoning on Knowledge Graphs with Debate Dynamics. arXiv:cs.LG/2001.00461

[6] Woojeong Jin, He Jiang, Meng Qu, Tong Chen, Changlin Zhang, Pedro Szekely, and Xiang Ren. 2019. Recurrent Event Network: Global Structure Inference over Temporal Knowledge Graph. *ICLR-RLGM* (2019).

[7] Nobuhiro Kaji et al. [n.d.]. Incremental Skip-gram Model with Negative Sampling. In *EMNLP 2017*. Association for Computational Linguistics. `https://doi.org/10.18653/v1/D17-1037`

[8] Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. 2019. Relational Representation Learning for Dynamic (Knowledge) Graphs: A Survey. *CoRR* abs/1905.11485 (2019). arXiv:1905.11485 `http://arxiv.org/abs/1905.11485`

[9] Zachary C. Lipton, Charles Elkan, and Balakrishnan Naryanaswamy. 2014. Optimal Thresholding of Classifiers to Maximize F1 Measure. In *Machine Learning and Knowledge Discovery in Databases*, Toon Calders, Floriana Esposito, Eyke Hüllermeier, and Rosa Meo (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 225–239.

[10] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2017. Modeling Relational Data with Graph Convolutional Networks. arXiv:stat.ML/1703.06103

[11] Yi Tay, Anh Luu, and Siu Cheung Hui. 2017. Non-Parametric Estimation of Multiple Embeddings for Link Prediction on Dynamic Knowledge Graphs. `https://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14524`

[12] Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. DKN: Deep Knowledge-Aware Network for News Recommendation. arXiv:stat.ML/1801.08284

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

[14] Tianxing Wu, Arijit Khan, Huan Gao, and Cheng Li. 2019. Efficiently Embedding Dynamic Knowledge Graphs. arXiv:cs.DB/1910.06708

[15] Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation Learning of Knowledge Graphs with Entity Descriptions. In *AAAI 2019*.